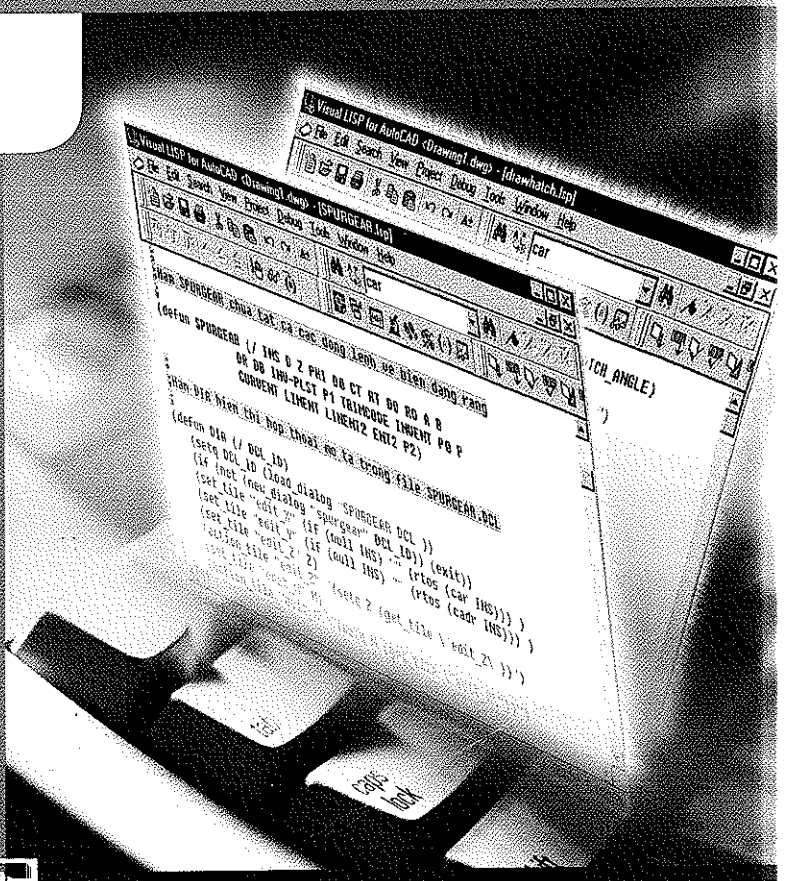
CAD

TS. NGUYỄN HỮU LỘC - NGUYỄN THANH TRUNG

Lập trình thiết kế với AutoLISP và Visual LISP

Tập 1



Nhà Xuất Bản Thành Phố Hồ Chí Minh

TS. NGUYỄN HỮU LỘC, NGUYỄN THANH TRUNG

Lập trình thiết kế với AutoLISP và Visual LISP

TẬP 1

In lần thứ 2

NHÀ XUẤT BẢN THÀNH PHỐ HỒ CHÍ MINH - 2003

LỜI NÓI ĐẦU

Ngày nay phần mềm **AutoCAD** được sử dụng rộng rãi trong công tác thiết kế. Nhu cầu về khai thác phần mềm này để ứng dụng chúng một cách hiệu quả hơn (như các phần mềm **Mechanical Desktop** và **AutoCAD Architectural Desktop**) trở thành bức thiết đối với các nhà thiết kế. Ngôn ngữ lập trình **AutoLISP**, là một bộ phần gắn liền với bộ phần mềm **AutoCAD**, đáp ứng được nhu cầu này.

AutoLISP, tập con ngôn ngữ **LISP**, là ngôn ngữ lập trình bậc cao thích hợp với các ứng dụng đồ họa. **AutoLISP** là một ngôn ngữ thông dịch, được viết theo cú pháp và thủ tục chặt chẽ như ngôn ngữ **LISP**. Tuy nhiên nó được bổ sung thêm các hàm để phù hợp với phần mềm **AutoCAD**.

Sử dụng **AutoLISP** ta có thể viết các chương trình macro để tạo các lệnh mới cho **AutoCAD** hoặc các chương trình tự động thiết kế các bản vẽ được sử dụng thường xuyên, thực hiện với các lệnh sẵn có của **AutoCAD** để góp phần tăng năng suất thiết kế. Bạn có thể sử dụng bất kỳ trình soạn thảo văn bản nào để tạo các chương trình **AutoLISP**. Để kiểm tra các chương trình này bạn gọi chúng vào trong **AutoCAD** và thực hiện. Môi trường **Visual LISP** được giới thiệu từ **AutoCAD 2000** giúp cho quá trình soạn thảo và gỡ rối chương trình thực hiện tốt hơn.

Sách được biên soạn theo các tài liệu kèm theo phần mềm của hãng **Autodesk**, giáo trình đào tạo tại các trung tâm **ATC** (Autodesk Training Center), một số ví dụ trong các tạp chí về **CAD**, một số luận văn tốt nghiệp sinh viên và theo kinh nghiệm sử dụng và giảng dạy của các tác giả.

Sách này có thể dùng làm tài liệu giảng dạy các khóa chuyên đề về lập trình trong thiết kế và giúp cho sinh viên và các cán bộ kỹ thuật tự học để tạo các phần mềm ứng dụng cho riêng mình. Sách được biên soạn

thành hai tập. Tập 1 bao gồm 12 chương, mỗi chương trình bày phần lý thuyết, các ví dụ mẫu, bài tập và các chương trình giải sẵn. Cùng với các sách đã xuất bản của cùng tác giả, chúng tôi mong muốn bạn đọc xa gần sử dụng hiệu quả hơn các phần mềm hiện có để góp phần tăng năng suất thiết kế, giảm giá thành sản phẩm, nâng cao tính canh tranh sản phẩm, nâng cao kiến thức xã hội và mang lại lợi ích cho cộng đồng.

Qua lần in thứ nhất **"Ngôn ngữ lập trình AutoLISP"** vào năm 2001 chứng tôi nhận được rất nhiều đóng góp bạn đọc xa gần, bởi vì trong tập 2 chúng tôi bổ sung chương về **Visual LISP** cho nên trong lần in thứ hai này chúng tôi đổi tên sách thành **"Lập trình thiết kế với AutoLISP và Visual LISP"**. Sử dụng tài liệu này có thể lập trình trên các phiên bản **AutoCAD 14, 2000, 2002** và **2004**.

Trong quá trình biên soạn để tái bản không thể tránh được thiếu sót. Chúng tôi xin thành thật cảm ơn các bạn có ý kiến đóng góp, phê bình những thiếu sót của sách để cho các lần xuất bản sau được hoàn thiện hơn. Mọi ý kiến đóng góp, phê bình và thắc mắc xin gửi đến địa chỉ:

Nguyễn Hữu Lộc, Bộ môn Thiết kế máy, Trường Đại học Bách Khoa TP. Hồ Chí Minh, 268 Lý Thường Kiệt, Quận 10.

hoặc liên hệ trực tiếp qua email: ***nhlcad@yahoo.com***

TP Hồ Chí Minh, 07 – 2003

NỘI DUNG

Chương 1

CĂN BẢN VỀ AutoLISP

Nội dung chương

1. Cấu trúc của biểu thức **AutoLISP**. Cách nhập biểu thức trong **AutoLISP**.
2. Các lỗi đơn giản gặp phải khi nhập biểu thức **AutoLISP**.
3. Kiểu số thực và số nguyên trong **AutoLISP**.
4. Sử dụng các hàm toán học của **AutoLISP** để biểu diễn các biểu thức đơn giản: cộng +, trừ -, nhân *, chia /.
5. Lồng các hàm **AutoLISP** vào nhau để biểu diễn các biểu thức phức tạp.
6. Các ký hiệu và các biến **AutoLISP**. Hàm **Setq**

LISP là một ngôn ngữ lập trình bậc cao thường được dùng cho việc nghiên cứu trí tuệ nhân tạo. Là một trong các ngôn ngữ lập trình cũ nhất vẫn được dùng, **LISP** viết tắt của **List Processing** đã được **John McCarthy** và các bạn đồng nghiệp tại Viện Kỹ thuật **Massachusets** biên soạn từ những năm đầu của thập niên 1960. **LISP** là một ngôn ngữ khai báo, lập trình viên chỉ cần soạn các danh sách biểu thị các mối quan hệ giữa các giá trị ký hiệu. Các bản danh sách là cấu trúc dữ liệu cơ bản của **LISP** và chương trình sẽ tiến hành các phép tính bằng các giá trị được diễn đạt trong các bản danh sách đó.

Một số phiên bản của **LISP** được tiêu chuẩn hoá, cấu hình đầy đủ và được chấp nhận rộng rãi hiện nay là **COMMON LISP**.

AutoLISP, tập con của ngôn ngữ **LISP**, là ngôn ngữ lập trình bậc cao thích hợp với các ứng dụng đồ họa. **AutoLISP** là một ngôn ngữ thông dịch, được viết theo các cu pháp và thủ tục chặt chẽ như ngôn ngữ **LISP**. Tuy nhiên, nó được bổ sung thêm các hàm để phù hợp với phần mềm **AutoCAD**.

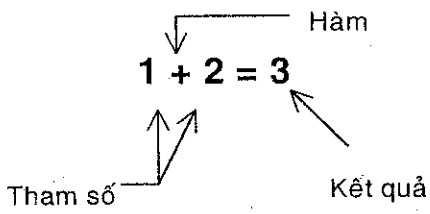
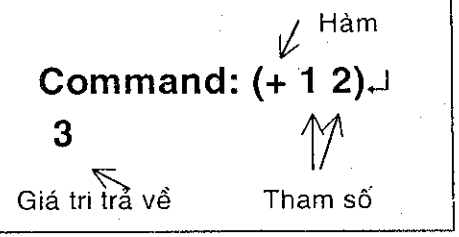
1.1 Xây dựng biểu thức AutoLISP

Khi bạn nhập dòng text tại dòng lệnh, **AutoCAD** dịch dòng text bằng cách so sánh các ký tự với danh sách các tên lệnh. Nếu các dòng text tương ứng với các lệnh của **AutoCAD** thì chúng sẽ đánh giá và thi hành các lệnh như mong muốn. Khi **AutoCAD** nhận các **AutoLISP** code thì chúng chuyển các code này đến bộ biên dịch **AutoLISP**.

Cấu trúc dữ liệu cơ bản của **AutoLISP** là *danh sách* (*list*). Danh sách là tập hợp các phần tử chứa trong các dấu ngoặc đơn. Các phần tử trong danh sách (còn gọi là các *atom*) phân cách nhau bởi các khoảng trắng.

Có hai loại danh sách: *biểu thức (expression)* và *danh sách dữ liệu (data list)*. Biểu thức là thành phần cơ bản trong tất cả các chương trình **AutoLISP**. Phần tử đầu tiên của biểu thức là một *hàm* (*function*). Hàm này sẽ được **AutoLISP** định giá trị và trả về kết quả.

Một biểu thức **AutoLISP** đơn giản tương tự như một đẳng thức toán học. Lấy ví dụ đẳng thức 1 + 2 = 3. Biểu thức **AutoLISP** tương ứng có dạng (+ 1 2). Ở đây, *hàm* là phép tính cộng. *Tham số* là giá trị cung cấp cho hàm để tính toán (1 và 2). *Giá trị trả về* là kết quả tính toán của hàm (3). Sự khác nhau chủ yếu giữa biểu thức **AutoLISP** và đẳng thức toán học là các phần tử biểu thức **AutoLISP** có thứ tự khác với đẳng thức và chúng phải được chứa trong các *dấu ngoặc đơn*.

ĐẲNG THỨC TOÁN HỌC	BIỂU THỨC AUTOLISP
	

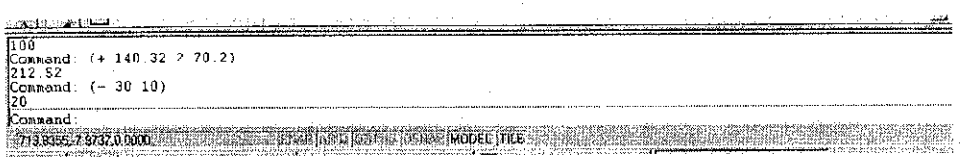
Tóm tắt:

1. Cấu trúc dữ liệu cơ bản của **AutoLISP** là danh sách (list).
2. Danh sách là tập hợp các phần tử chứa trong các dấu ngoặc đơn.
3. Phần tử đầu tiên của biểu thức luôn luôn là tên một hàm.
4. Một biểu thức bao gồm tên hàm và các tham số chứa trong các dấu ngoặc đơn.
5. Tham số là giá trị cung cấp cho hàm để tính toán.
6. **AutoLISP** trả về kết quả tính toán được từ biểu thức.

1.2 Cách nhập biểu thức AutoLISP

Ta có thể nhập và định giá trị biểu thức **AutoLISP** giống như nhập các lệnh **AutoCAD** (nhớ đặt biểu thức trong các dấu ngoặc đơn) bằng các phương pháp: nhập trực tiếp tại dòng nhắc lệnh của **AutoCAD**, gọi từ menu, hoặc tải từ file chương trình **AutoLISP** (xem chương 2). Sau đó, biểu thức sẽ được định giá trị và trả về kết quả.

Khi nhập biểu thức tại dòng nhắc lệnh, ta nên dùng cửa sổ *AutoCAD Text Window* (nhấn phím F2) hoặc mở rộng chiều cao cửa sổ lệnh để nhìn thấy được ít nhất 6 dòng lệnh (hình vẽ). Nhờ đó ta có thể thấy được giá trị trả về hoặc các thông báo lỗi nếu có.



ngay tại cửa sổ dòng nhắc lệnh. Nếu ta kết thúc biểu thức bằng phím ENTER, kết quả trả về ở dòng tiếp theo. Nếu ta kết thúc biểu thức bằng phím SPACEBAR, kết quả trả về trên cùng một dòng.

Ví dụ:

Command: **(+ 5 8)** *13* (kết thúc bằng SPACEBAR)
Command: **(+ 5 8)**↵ (kết thúc bằng ENTER)
13

Nếu biểu thức không bị lỗi, kết quả sẽ được trả về tại dòng nhắc lệnh. Nếu biểu thức bị lỗi, thông báo lỗi tương ứng sẽ xuất hiện kèm với biểu thức bị lỗi.

Các lỗi thường gặp phải:

Command: **(+ 5 8)**↵
13
Không bị lỗi. Kết quả trả về là 13.

Command: **(+6 8)**↵
Error: bad function
(6 8)
Cancel
Biểu thức này cần một khoảng trắng để ngăn cách tên hàm + với tham số 6. Phần tử đầu tiên trong biểu thức này là +6, sẽ được hiểu là dương 6. Vì đây không phải là tên hàm, nên thông báo lỗi được trả về, kèm theo biểu thức gây ra lỗi, sau đó thoát khỏi quá trình tính toán (*Cancel*)

Command: **(+ 8 .45)**↵
Error: invalid dotted pair
Cancel
Tham số .45 bị sai. Cần phải ghi đầy đủ 0.45. Thông báo lỗi được trả về, sau đó thoát khỏi quá trình tính toán (*cancel*)

Command: **(+ 6 8** ↵
1> **)**
14
Biểu thức chưa được đóng lại bằng dấu ngoặc đơn. Dòng kế tiếp biểu diễn số lượng dấu ngoặc cần dùng để đóng biểu thức (1 dấu ngoặc). Ta nhập vào dấu ngoặc bị thiếu tại dòng thông báo này. Biểu thức tiếp tục được đinh giá trị và kết quả 14 được trả về.

Command: **(+ 6 8** ↵
1>(Nhấp ESC) *error: input aborted*
Cancel
Thay vì nhập thêm dấu ngoặc như trong trường hợp trên, ta thoát khỏi quá trình tính toán bằng phím ESC.

1.3 Các hàm số học

Ở trên, chúng ta đã gặp một ví dụ về hàm + của **AutoLISP** (+ 1 2). Bốn hàm toán học căn bản của **AutoLISP** cũng có cú pháp tương tự. Tuy nhiên mỗi hàm đều có các quy định riêng khi sử dụng.

Hàm cộng +

Hàm + nhận vào nhiều tham số và trả về tổng các tham số này.

(+ [NUMBER NUMBER] . . .)

Ví dụ:

Biểu thức **AutoLISP**			Đẳng thức toán học
(+ 15 20)	trả về	35	15 + 20 = 35
(+ 10 25 30)	trả về	65	10 + 25 + 30 = 65
(+ 10 20 30 40 5)	trả về	1055	10 + 20 + 30 + 40 + 5 = 105
(+ 10 20 30 –20)	trả về	40	10 + 20 + 30 + (-20) = 40
(+ 300)	trả về	300	300 + 0 = 300

Như đã trình bày ở trên, thông tin cung cấp cho **AutoLISP** phải ở dạng một danh sách. Khi danh sách được chuyển cho **AutoLISP**, mỗi phần tử của danh sách sẽ được đinh giá trị. Nếu phần tử đầu tiên là tên một hàm, các phần tử còn lại sẽ được xem như là tham số cho hàm đó. Trong cú pháp trên, các tham số NUMBER được chuyển cho hàm **+**.

Các dữ liệu số được chia thành hai kiểu dữ liệu: *số nguyên* và *số thực*. Số nguyên là những số không có số thập phân (như 1, 4, -38 hoặc 72). Khi biểu diễn số nguyên, ta không dùng dấu chấm thập phân. Số thực là những số có dấu chấm thập phân (như 1.25, 15.0, -19.002, hoặc 357.14).

Kiểu dữ liệu của giá trị trả về phụ thuộc vào kiểu dữ liệu của các tham số:

- Tất cả các tham số là số nguyên. Giá trị trả về là số nguyên.
- Tất cả các tham số là số thực. Giá trị trả về là số thực.
- Các tham số là số nguyên lẫn số thực. Các số nguyên sẽ được chuyển đổi thành số thực. Giá trị trả về là số thực.

Quy định này áp dụng cho cả 4 hàm toán học: cộng, trừ, nhân chia.

Ví dụ:

(+ 20 30)	trả về	50
(+ 20 30.0)	trả về	50.0
(+ 20 30 50)	trả về	100
(+ 20.0 30 50.0)	trả về	100.0

(+ 120 –20 40 60 –100) trả về 100
(+ 140.32 20 70.2) trả về 230.52

Hàm trừ -

Hàm trừ - trả về hiệu số của tham số thứ nhất với tổng của các tham số còn lại. Nếu chỉ cung cấp một tham số, kết quả trả về là hiệu số của số không (0) với tham số này.

(- [NUMBER NUMBER] . . .)

Ví dụ:

Biểu thức **AutoLISP**			Đẳng thức toán học
(– 30 10)	trả về	20	30 – 10 = 20
(– 3.0 1)	trả về	2.0	3.0 – 1 = 2.0
(– 10 2 3)	trả về	5	10 – (2 + 3) = 5
(– 12)	trả về	-12	0 – 12 = -12
(– 100 25 25 50)	trả về	0	100 – (25 + 25 + 50) = 0
(– 40 20 –10)	trả về	30	40 – [20 + (-10)] = 30
(– 25 10.5 –15.5)	trả về	30.0	25 – [10.5 + (-15.5)] = 30.0

Hàm nhân *

Hàm nhân * trả về tích số của các tham số. Nếu chỉ có một tham số, tham số này sẽ được nhân với 1.

(* [NUMBER NUMBER] . . .)

Ví dụ:

Biểu thức **AutoLISP**			Đẳng thức toán học
(* 2 3)	trả về	6	2 x 3 = 6
(* 2)	trả về	2	2 x 1 = 2
(* 2 4.0 3)	trả về	24.0	2 x 4.0 x 3 = 24.0
(* 2 –4.3)	trả về	-8.6	2 x (-4.3) = -8.6
(* 2 –2.5 –2.0)	trả về	10.0	2 x (-2.5) x (-2.0)
(* 2.25 6.375)	trả về	14.3438	2.25 x 6.375 = 14.3438 (làm tròn)
(* 13.75 3.625)	trả về	49.8438	13.75 x 3.625 = 49.8438 (làm tròn)

Hai giá trị trả về sau cùng đã được làm tròn. Mặc dù, **AutoLISP** lưu trữ các số với ít nhất là 14 số thập phân, nhưng trên màn hình chỉ xuất hiện **6** chữ số có nghĩa (tính từ trái qua phải). Các số 0 ở cuối phần thập phân không được xem là các chữ số có nghĩa.

Ví dụ:

22,475.5
11.0123
278.91 (278.910 là không đúng)

Hàm chia /

Hàm chia / sẽ chia tham số thứ nhất cho tích các tham số còn lại và trả về kết quả phép chia. Nếu chỉ có một tham số, tham số nay sẽ được chia cho 1.

(/ [NUMBER NUMBER] . . .)

Ví dụ:

Biểu thức **AutoLISP**			Đẳng thức toán học
(/ 2)	trả về	2	2 – 1 = 2
(/ 4 2)	trả về	2	4 – 2 = 2
(/ 3 2)	trả về	1	3 – 2 = 1 (kết quả là số nguyên)
(/ 3 2.0)	trả về	1.5	3 – 2.0 = 1.5 (kết quả là số thực)
(/ 40 5 2)	trả về	4	40 – (5 x 2) = 4
(/ 100 60)	trả về	1	100 – 60 = 1
(/ 100 60.0)	trả về	1.66667	100 – 60.0 = 1.66667 (6 chữ số có nghĩa)
(/ 60 3 2 5)	trả về	2	60 – (3 x 2 x 5) = 2
(/ 60 3 2 –5.0)	trả về	-2.0	60 – [3 x 2 x (-5.0)] = -2.0

1.4 Các biểu thức số học phức tạp

Ta có thể lồng các hàm với nhau để tính các biểu thức phức tạp.

Ví dụ:

Biểu thức **AutoLISP**	Đẳng thức toán học	Giải thích
(- 40 (* 8 3))	40 – (8 x 3)	Trong ví dụ này, hàm trừ – có hai tham số. Tham số thứ nhất là 40, tham số thứ hai là một biểu thức. Các biểu thức lồng nhau được đinh giá trị theo thứ tự từ trong ra ngoài.

Ví dụ:

Biểu thức **AutoLISP**	Đẳng thức toán học	Giải thích
(/ 100 (- 40 (* 3 5)))	100 – (40 – (3 x 5))	Trong ví dụ này, biểu thức trong cùng (* 3 5) sẽ được định giá trị trước. Giá trị trả về là 15 được truyền cho biểu thức bên ngoài.
(/ 100 (- 40 15))	100 – (40 – 15)	Kế đến, biểu thức bên ngoài (- 40 15) được định giá trị, và kết quả được truyền cho biểu thức ngoài cùng:
(/ 100 25)	100 – 25	Biểu thức ngoài cùng được tính và trả về kết quả bằng 4.

Ví dụ: Chuyển biểu thức toán học sau đây thành biểu thức **AutoLISP**:

(720 ÷ 360) + (49 x 2)

Biểu thức **AutoLISP**	
(+ (/ 720 360) (* 49 2))	Trong biểu thức trên, phép chia và phép nhân cùng một mức, phép cộng ở mức ngoài cùng.
(+ 2 98)	Biểu thức (/ 720 360) và (* 49 2) được tính trước tiên, kết quả truyền cho biểu thức bên ngoài
100	Kết quả trả về của biểu thức này là 100.

Khi tạo biểu thức **AutoLISP**, trước tiên ta nên tạo các biểu thức ở mức sâu nhất, sau đó chuyển dần sang các biểu thức bên ngoài.

Ví dụ: 18 – [(3 + 6 + 9) – (9 – 6)] – 12

Bước 1: Mức sâu nhất	(+ 3 6 9) (- 9 6)
Bước 2: Mức kế tiếp	(/ (+ 3 6 9) (- 9 6))
Bước 3: Mức ngoài cùng	(- 18 (/ (+ 3 6 9) (- 9 6)))
Bước 4: Biểu thức kết quả	(- (- 18 (/ (+ 3 6 9) (- 9 6))) 12)

Tóm tắt:

1. Nếu phần tử đầu tiên của biểu thức **AutoLISP** là tên hàm, tất cả các phần tử còn lại được xem như là các tham số của hàm đó.
2. Bốn hàm toán học cơ bản có cùng dạng: (<FUNCTION> <ARGUMENT> <ARGUMENT> ...). Mỗi hàm có các đặc điểm riêng.
3. Các hàm số học **AutoLISP** trả về kết quả là số nguyên chỉ khi tất cả các tham số đều là số nguyên.
4. Phương pháp lồng các hàm với nhau dùng để tính các biểu thức phức tạp.
5. Khi chuyển các biểu thức toán học phức tạp thành các biểu thức **AutoLISP**, trước tiên ta nên tạo các biểu thức ở mức sâu nhất, sau đó chuyển dần sang các biểu thức bên ngoài.

Thay vì nhập trực tiếp các giá trị tại dòng nhắc lệnh của **AutoCAD**, ta có thể nhập vào các biểu thức **AutoLISP**.

Ví dụ:

Command: **Circle**↵
3P/2P/TTR/<Center point>: (Nhập tâm đường tròn)
Diameter/<Radius>: **D** ↵
Diameter: **(* 0.125 240)** ↵
Command:

Giá trị của biểu thức (* 0.125 240) là 30 sẽ được gán cho dòng nhắc *Diameter:* giống như ta nhập trực tiếp giá trị này từ bàn phím. Dùng biểu thức **AutoLISP** trong các trường hợp tương tự như trong ví dụ này thuận tiện và nhanh hơn dùng lệnh **'Cal** của **AutoCAD**.

1.5 Các biến và các ký hiệu trong AutoLISP

Các giá trị tĩnh (không đổi) như tên hàm **AutoLISP** (ví dụ, +, -, *, /), tên hàm tự tạo (xem chương 2), hoặc các hằng số (như *Pi*) gọi chung là các *ký hiệu* (*symbol*). Các dữ liệu thay đổi trong chương trình được gọi là các *biến* (*variable*). Dữ liệu chứa trong các biến thay đổi tùy theo các tham số cung cấp cho chương trình. Trong hầu hết các trường hợp, ký hiệu và biến được tạo ra và quản lý tương tự như nhau. Tên gọi phụ thuộc vào giá trị của chúng là tĩnh hay động.

Tên biến và ký hiệu (cũng như tên hàm) không phân biệt chữ hoa, chữ thường. Ví dụ, các tên aBcd và Abcd đều là tên của cùng một biến.

1.5.1 Gán giá trị cho các biến

Trong ngôn ngữ **AutoLISP**, ta gán giá trị cho các biến bằng hàm **Setq**. Hàm **Setq** gán giá trị cho biến và trả về kết quả là giá trị này. Các biến có thể chứa bất kỳ kiểu dữ liệu nào: số nguyên, số thực, chuỗi, danh sách, hoặc các kiểu khác.

Hàm **Setq** có cú pháp như sau:

(**Setq** SYMBOL1 VALUE1 [SYMBOL2 VALUE2] ...)

Hàm **Setq** có thể gán giá trị cho nhiều biến cùng lúc và trả về kết quả là giá trị được gán cho biến cuối cùng.

Ví dụ:

Command: **(setq X 3)** ↵ *3*	X được gán giá trị bằng 3. Kết quả trả về là 3.
Command: **(setq Y 4 Z 9)** ↵ 9	Y được gán bằng 4, Z được gán bằng 9. Kết quả trả về là 9.
Command: **(setq A Z)**↵ *9*	A được gán giá trị bằng Z. Giả sử Z bằng 9, khi đó A bằng 9. Kết quả trả về là 9.
Command: **(setq X (+ 1 2))**↵ *3*	Biểu thức (+ 1 2) được định giá trị, trả về kết quả là 3. Hàm **Setq** gán giá trị 3 cho A, và trả về là 3.
Command: **(setq X (+ Y Z))**↵ *13*	Giá trị của biểu thức (+ Y Z) được gán cho X.

1.5.2 Giá trị trả về của hàm AutoLISP

Giống như tất cả các hàm khác của **AutoLISP**, hàm **Setq** trả về một giá trị. Giá trị này có thể là "nil" (rỗng), "T" (True) hoặc các số, chuỗi, danh sách.

Ví dụ:

Biểu thức **AutoLISP**	Giá trị trả về	Giải thích
(setq a 14)	14	Biến a được gán bằng 14
(setq X Y)	nil	Vì Y bằng *nil* nên X cũng được gán bằng *nil*.
(setq b A)	14	Giá trị của A được gán cho b.
(setq z (* 3 4))	12	Z được gán giá trị bằng giá trị của biểu thức (*3 4).
(setq X (- B Z))	2	Vì B bằng 14, Z bằng 12, nên kết quả là 2 được gán cho X.
(setq M "XYZ")	"XYZ"	Chuỗi "XYZ" được gán cho M.
(setq aBcd 40)	40	Gán cho biến aBcd giá trị 40.
(setq X AbCD)	40	Vì không phân biệt chữ hoa chữ thường, nên AbCD cũng chính là biến aBcd ở trên. Do đó X được gán giá trị của biến aBcd là 40.

Để đổi dấu cho giá trị kiểu số chứa trong biến, ta dùng hàm trừ với một tham số duy nhất là biến này.

Ví dụ:

Command: **(setq A 15.0)**↵	A được gán bằng 15.0.
Command: **(setq B (- A))**↵	Biểu thức (- A) trả về giá trị là -15.0. Để ý giữa hàm trừ và biến A có khoảng trắng. B được gán bằng –15.0.
Command: **(setq C (+ (- B) (- A)))**↵	(- B) bằng 15.0, (- A) bằng -15. Do đó C được gán giá trị 0.0

Khi tham số của hàm là tên biến, **AutoLISP** sẽ định giá trị của biến và sử dụng giá trị này làm tham số.

Ví dụ:

(setq n 1)↵	n được gán bằng 1.
(+ 4 n)	Trả về giá trị là 5.
(setq n (+ 2 n))	Biểu thức trong cùng sẽ cộng 2 với n, trả về giá trị là 3. Biểu thức ngoài cùng sẽ gán cho biến n giá trị trả về của biểu thức bên trong. Do đó n được gán giá trị 3 và biểu thức **Setq** trả về giá trị 3.

Kiểu dữ liệu chứa trong biến phải phù hợp với kiểu dữ liệu mà hàm yêu cầu. Ví dụ như khi hàm yêu cầu tham số là số nguyên, ta phải dùng

biến chứa số nguyên. Các dữ liệu chuỗi dĩ nhiên không thể dùng làm tham số cho các hàm số học.

1.5.3 Sử dụng các biến tại dòng nhắc lệnh của AutoCAD

Giống như biểu thức **AutoLISP**, các biến có thể được định giá trị tại dòng nhắc lệnh của **AutoCAD**. Để lấy giá trị của một biến, ta đặt dấu chấm than ! vào trước tên biến.

✌ **Ví dụ:**

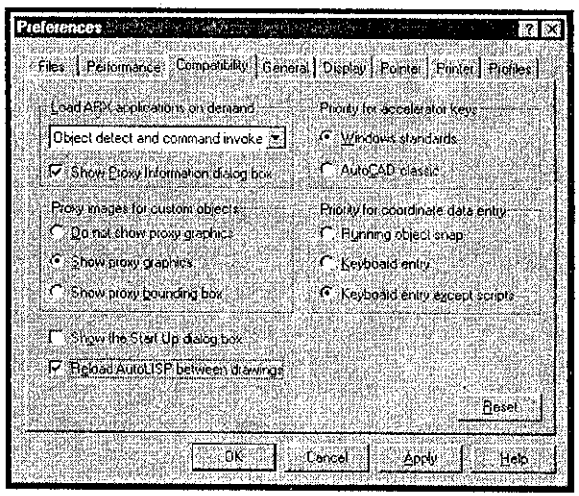
Command: **(setq X 5)**↵	Sử dụng hàm **Setq**
5	
Command: **!X** ↵	Trả về giá trị chứa trong biến X
5	
Command: **Circle** ↵	Lệnh vẽ đường tròn
3P/2P/TTR/<Center point>:	Nhập một điểm làm tâm đường tròn
Diameter/<Radius>: **!X** ↵	Gán bán kính đường tròn bằng giá trị chứa trong biến X là 5.

1.5.4 Các quy định về đặt tên biến

1. Tên biến có thể chứa bất kỳ ký tự nào, ngoại trừ các ký tự sau: ( ) . ' " và khoảng trắng.
2. **AutoLISP** chứa các hằng số tạo sẵn như *pi* bằng 3.1415926. Ta không nên gán giá trị khác cho các hằng số này, vì giá trị cũ sẽ bị mất đi, không lấy lại được.
3. Không đặt tên các biến trùng với tên các hàm **AutoLISP**.
4. Không nên đặt tên quá dài và phức tạp. Tên ngắn dễ nhập hơn, khi nhập ít bị lỗi hơn và được truy xuất nhanh hơn.

1.5.5 Phạm vi các biến AutoLISP

Trong các phiên bản **AutoCAD** trước, tất cả các biến **AutoLISP** được khởi tạo lại và nhận các giá trị mặc định ban đầu khi ta tạo mới hoặc mở bản vẽ khác. Do đó, các giá trị gán cho biến chỉ có tác dụng trong phạm vi một bản vẽ. Điều này có ích khi ta gán nhầm giá trị cho các tên biến và tên hàm tạo sẵn của **AutoLISP**. Khi đó ta chỉ cần dùng lệnh **Open** để mở lại bản vẽ ta đang làm việc hoặc tạo bản vẽ mới.



Bắt đầu từ **AutoCAD 14**, các biến **AutoLISP** chỉ được khởi tạo lại khi ta đóng cửa sổ chương trình **AutoCAD**. Giá trị gán cho các biến vẫn tồn tại khi ta tạo mới hoặc mở bản vẽ khác (nhưng không đóng cửa sổ chương trình **AutoCAD**). Nếu ta mở một lúc nhiều chương trình **AutoCAD**, các biến **AutoLISP** trong chương trình này không thể truy xuất được từ chương trình khác.

Để **AutoCAD 14** khởi tạo lại các biến **AutoLISP** khi tạo mới hoặc mở bản vẽ khác trong cùng một cửa sổ chương trình (giống như các phiên bản trước) ta gán giá trị biến hệ thống LISPINIT bằng 1, hoặc dùng hộp thoại **Preferences**, đánh dấu mục *Reload AutoLISP between drawings* của trang **Compatibility**. Nhưng thông thường ta không nên sửa đổi tính năng này của **AutoCAD**.

Tóm tắt:

1. Hàm **Setq** được dùng để gán giá trị cho các biến.
2. Các khái niệm ký hiệu và biến tương tự nhau, ngoại trừ ký hiệu dùng để chứa các giá trị tĩnh, biến dùng để chứa các giá trị động.
3. Có thể gán giá trị cho nhiều biến trong cùng một biểu thức.
4. Tên biến không được trung với tên hàm và hằng số của **AutoLISP**
5. **AutoLISP** có những hằng số tạo sẵn như *pi* (3.1415926).
6. Các biến **AutoLISP** được khởi tạo lại khi ta thoát khỏi **AutoCAD**, nhưng chúng vẫn giữ nguyên khi tạo mới hoặc mở bản vẽ khác mà không thoát khỏi **AutoCAD** (với điều kiện ta không tắt chức năng này).

1.6 Bài tập

1. Hãy chuyển các biểu thức toán học thành các biểu thức **AutoLISP** và nhập vào tại dòng nhắc lệnh của **AutoCAD** để tính kết quả.

44 + 27	6 x 9
6 + 33 + 12	4 x 5 x 3
4.2 +53 + 2.8	6.1 x 5
5.25 + 19.375	3.5 x 2.875
9 + 3 + 1.4 + 9	4 x 0.5
14 – 4	42 ÷ 7
12 – 7.3	14 ÷ 4
– 3.02	14.0 ÷ 4
9.418 – (2.5 + 6.39)	28.5 ÷ (8.5 x 1.25)
43 – (7 + 4 + 77)	9 ÷ (0.3 x 9)

2. Hãy liệt kê các thông báo lỗi (nếu có) khi nhập các biểu thức **AutoLISP** ở câu 1 tại dòng nhắc lệnh. Các lỗi mắc phải đã được sửa lại như thế nào?

3. Chuyển các biểu thức toán học sau đây thành các biểu thức **AutoLISP**:

15 + (6 x 4)	(45 ÷ 5) x (13 – 4)
100 – [22 – (10 + 1)]	(231 ÷ 12) x 14
16 x 4 x (81 ÷ 9)	28 ÷ [14 x (6 x 0.3)]
45 ÷ (3 x 5)	53 + (22 x 4)
106 – (2 + 2 + 2 + 2)	1 + 2 – (3 x 0.75)

4. Hãy chuyển các biểu thức toán học sau đây thành các biểu thức **AutoLISP**:

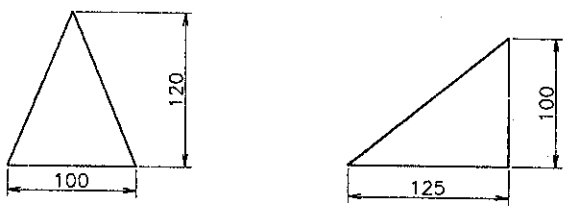
[21 ÷ (7 x 3)] + (4 x 5)
(27 – 7) – [(2 x 3) ÷ (3 x 2)] – (3 x 6)
[(25 – 4 – 6) ÷ (3 x 5)] + (4 x 3)
[256 ÷ (16 x 16)] + [255 ÷ (15 x 15)]
(7 x 6) ÷ {142 – [10 x (80 – 8)]}
[697 – (47 + 50)] ÷ 6
[(25 ÷ 5) x 36] – (4 x 9 x 5)
27216 – (7 x 6 x 18 x 12.0)
22 – {12 ÷ [-5 x (44.0 ÷ -22)]}
[31 + 15 + (-10)] ÷ (4.5 x 2)

5. Hãy biểu diễn bằng biểu thức **AutoLISP** các công việc sau:

a. Gán tổng các số 6, 22 và 14 cho biến X.

b. Nhân hai số 4 và 3. Chia số 144 cho tích số này. Kết quả gán cho biến Z.
c. Lấy Y trừ X và gán kết quả cho biến Z.
d. Gán giá trị 2 x (A + B) cho biến E.
e. Gán giá trị (D – 4) + B – A cho biến F.
f. Gán giá trị Z – A + B cho biến G.
g. Gán giá trị [(4.0 x B) ÷ C] x A cho biến H.
h. Chuyển biểu thức toán học sau đây thành biểu thức **AutoLISP**:
i. {(A – B) ÷ [(C + D) – (E x F)]} ÷ 2
j. Chuyển biểu thức toán học sau đây thành biểu thức **AutoLISP**:
k. (F – B) ÷ (G – H) – [(3 + C) x 1.75]

6. Viết các biểu thức **AutoLISP** tính diện tích các tam giác sau:



7. Viết biểu thức tổng quát tính diện tích tam giác:

- Dùng hàm **Setq** gán chiều dài cạnh đáy cho biến **b** và chiều cao cho biến **h**.
- Nhập biểu thức tính diện tích tam giác dựa trên hai biến này.

1.7 Lời giải

1.

(+ 44 27)	(+ 6 9)
(+ 6 33 12)	(* 4 5 3)
(+ 4.2 53 2.8)	(* 6.1 5)
(+ 5.25 19.375)	(* 3.5 2.875)
(+9 3 1.4 9)	(* 4 0.5)
(- 14 4)	(/ 42 7)
(- 12 7.3)	(/ 14 4)
(- 3.02)	(/ 14.0 4)
(- 9.418 2.5 6.39)	(/ 28.5 8.5 1.25)
(- 43 7 4 77)	(/ 9 0.3 9)

3.
```
(+ 15 (* 6 4))                    (* (/ 45 5) (- 13 4))
(/ 100 (/ 22 (+ 10 1)))           (* (/ 231 12) 14)
(* 16 4 (/ 81 9))                 (/ 28 (* 14 (* 6 0.3)))
(/ 45 (* 3 5))                    (+ 53 (* 22 4))
(- 106 (+ 2 2 2 2))               (+1 2 (- (* 3 0.75)))
```

4.
```
(+ (/ 21 (* 7 3)) (* 4 5))
(+ (/ 21 7 3) (* 4  5))
(- (- 27 7) (/ (* 2 3) (* 3 2)) (* 3 6))
(+ (/ (- 25 4 6) (* 3 5)) (* 4 3))
(+ (/ 256 16 16) (/ 255 (* 15 15)))
(/ (* 7 6) (- 142 (* 10 (/ 80 8))))
(/ (- 697 (+ 47 50)) 6)
(/ (* (/ 25 5) 36) (* 4 9 5))
(/ 27216 7 6 18 12.0)
(- 22 (/ 12 (* -5 (/ 44.0 –22))))
(/ (+ 31 15 –10) (* 4.5 2))
```

5.
```
(setq X (+ 6 22 14)
(setq Z (/ 144 (* 4 3)))
(setq  Z (- X Y)
(setq E (* 2 (+ A B)))
(setq F (- (+ (/ D 4) B) A))
(setq G (+ (- Z A) B))
(setq H (* (/ (* 4.0 B) C) A))
(/ (/ (- A B) (- (+ C D) (* E F))) 2)
(- (/ (- F B) (/ G H)) (* (+ 3 C) 1.75))
```

6.
```
(/ (* 100 120) 2)
(/ (* 125 100) 2)
```

7.
```
(setq b 100)
(setq h 120)
(/ (* b h) 2)

(setq b 125)
(setq h 100)
(/ (* b h) 2)
```

Chương 2

FILE CHƯƠNG TRÌNH AutoLISP

Nội dung chương

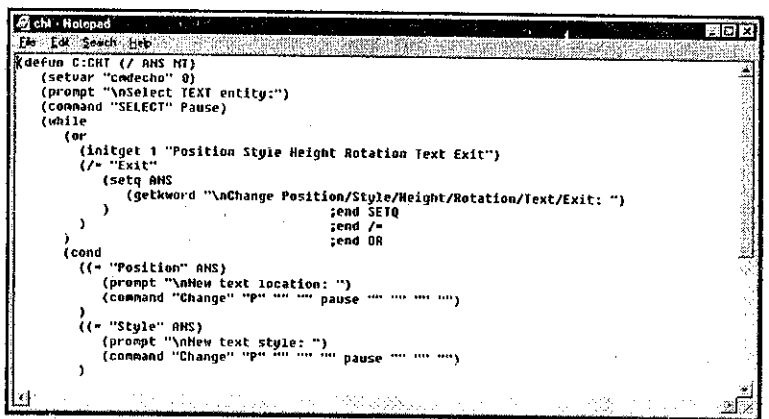
1. Cấu trúc file chương trình **AutoLISP**.
2. Tạo và sử dụng file chương trình **AutoLISP**: hàm **Load**, lệnh **Appload**.
3. Nhập điểm bằng hàm **Getpoint**, hiện thông báo và dòng nhắc bằng hàm **Promt**.
4. Sử dụng các lệnh **AutoCAD** trong **AutoLISP**.
5. Xây dựng các hàm tự tạo. Hàm **Defun**.
6. Tạo lệnh mới cho **AutoCAD**.

2.1.1 Tên file AutoLISP

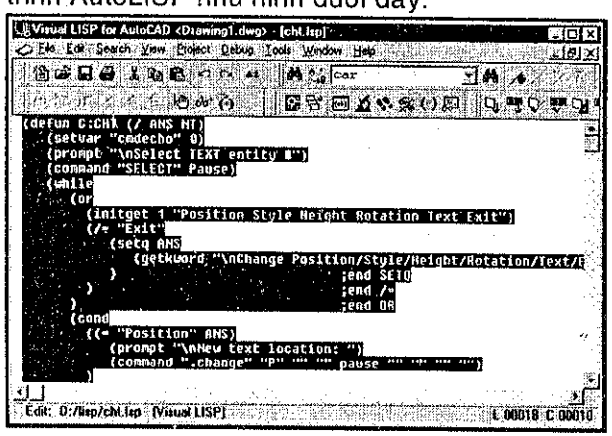
Tên file **AutoLISP** phụ thuộc vào hệ điều hành. Khi dùng **Windows 95, 98, 2000, Windows NT** (hoặc các phiên bản mới hơn) ta có thể đặt tên file dài đến 256 ký tự. Phần mở rộng (mặc định) của tên file là **.LSP**. Trong một số trường hợp, ta có thể dùng tên mở rộng khác. Ví dụ các file menu tự tạo bằng **AutoLISP** của **AutoCAD** có phần mở rộng là .MNL (MeNu Lisp).

2.1.2 Tạo file chương trình

File chương trình **AutoLISP** chỉ chứa các ký tự mã ASCII chuẩn. Ta có thể dùng các phần mềm soạn thảo văn bản như **Notepad, Microsoft Word** để tạo file và lưu chúng ở dạng *simple text*, ví dụ như chương trình CHT.LSP trên **Notepad** như hình trên.



Từ **AutoCAD 2000** ta có thể sử dụng lệnh **Vlide** hoặc từ **Tools** menu ta chọn *AutoLISP>Visual LISP Editor* sẽ xuất hiện màn hình soạn thảo chương trình AutoLISP như hình dưới đây.



Chú ý

- Một biểu thức có thể viết trên nhiều dòng.
- Ta có thể dùng các khoảng trắng để chương trình dễ đọc.
- Trong phần lớn các trường hợp, các biểu thức không phân biệt dạng chữ hoa chữ thường.

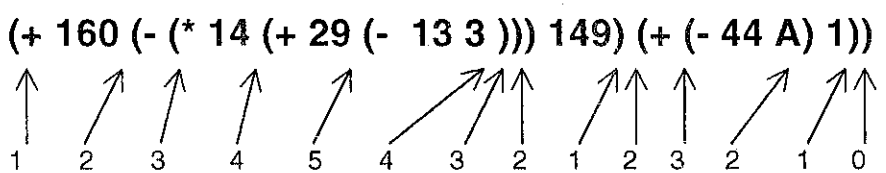
2.1.3 Các dấu ngoặc đơn

Mỗi biểu thức **AutoLISP** phải được đặt trong cặp dấu ngoặc đơn.

Ví dụ:

```
(setq A 43.9823)
(setq A (* 14 pi))
```

Khi các biểu thức lồng nhau, các dấu ngoặc mở và đóng của từng biểu thức phải được đặt đúng vị trí. Có một phương pháp nhanh để kiểm tra các dấu ngoặc là đếm các dấu ngoặc từ trái sang phải. Bắt đầu từ 0, khi gặp dấu ngoặc mở "(" thì cộng thêm 1, khi gặp dấu ngoặc đóng ")" thì trừ đi 1 (xem hình). Nếu kết quả khác 0, biểu thức ta viết đã bị lỗi. Tuy nhiên, phương pháp này chỉ giúp phát hiện việc thừa thiếu dấu ngoặc chứ không phát hiện được các dấu ngoặc đặt sai vị trí.



2.1.4 Dấu nháy chuỗi

Các dữ liệu kiểu chuỗi phải đặt trong cặp dấu nháy chuỗi. Nếu chuỗi dữ liệu không đặt trong dấu nháy chuỗi, **AutoLISP** sẽ xem đó là tên ham.

Ví dụ:

Command: **(setq A "OK")** ↵	Gán chuỗi "OK" cho biến A và trả về "OK".
"OK"	
Command: **(setq A "OK)** ↵	Chuỗi gán cho A chưa được đóng lại bằng dấu nháy, do đó dấu ngoặc đóng lại ")" bị xem như là một phần của chuỗi "OK)". Dòng thứ hai và ba yêu cầu nhập vào dấu nháy chuỗi để đóng chuỗi và dấu ngoặc đơn để đóng biểu thức. Chuỗi trả về chứa dấu ngoặc và ký tự xuống dòng \n.
1>"↵	
1>) ↵	
"OK)\n"	

Ngoài ký tự xuống dòng \n, **AutoLISP** còn sử dụng các ký tự điều khiển khác. Chúng tôi sẽ được trình bày trong các phần sau.

Tóm tắt:

1. Khi gặp các biểu thức phức tạp, ta nên tạo file chương trình chứ không nên nhập trực tiếp tại dòng nhắc lệnh.
2. File chương trình **AutoLISP** chứa một hoặc nhiều biểu thức **AutoLISP** và được lưu ở dạng file văn bản ASCII.
3. Khi viết chương trình ta nên cẩn thận sử dụng đúng các dấu ngoặc và các dấu nháy chuỗi để tránh gây ra lỗi.

Ví dụ: Tạo file **ABC.LSP** có nội dung sau:

```
;Tên file: ABC.LSP
;Người viết: NHL
(setq A (/ 100 4))
(setq B (+ A 75))
(setq C (/ B A))
;Kết thúc chương trình
```

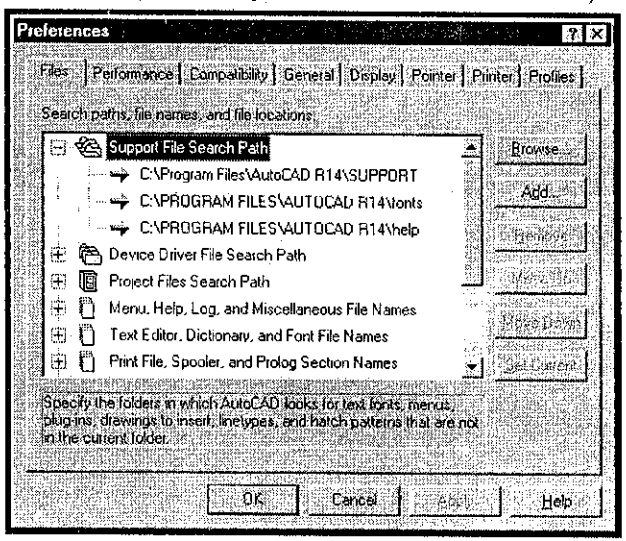
2.1.5 Hàm LOAD - gọi file chương trình AutoLISP

Hàm **Load** dùng để đọc file chương trình, kiểm tra các lỗi cú pháp, định giá trị các biểu thức và trả về giá trị của biểu thức cuối cùng.

(**load** FILENAME [ONFAILURE])

FILENAME

Tên file chương trình. Tên mở rộng mặc định là .LSP Đường dẫn thư mục mặc định chứa trong đường dẫn tìm kiếm thư viện của **AutoCAD** (mục **Support File Search Path** trên trang **Files** của hộp thoại **Preferences** trên **AutoCAD 14** hoặc hộp thoại **Options** trên **AutoCAD 2000**).



Khi cần thiết ta phải cung cấp đầy đủ tên file mở rộng và đường dẫn thư mục. Trong đường dẫn thư mục, ta dùng dấu / hoặc \\ thay cho dấu \.

Ví dụ 1: Ta gọi file CHT.LSP trên thư mục AutoLISP ổ đĩa E như sau:
Command:**(load E:/AutoLISP/CHT.LSP)**↵

hoặc
Command:**(load E:\\AutoLISP\\CHT.LSP)**↵

Ví dụ 2:

Command: **(load "ABC")**↵	Đọc file ABC.LSP tạo ra trong ví dụ trên, định giá trị các biểu thức và trả về giá trị của biểu thức cuối cùng.
4	
Command: **!A** ↵	Trong file ABC.LSP có hai biểu thức setq gán giá trị cho hai biến A và B. Tuy chương trình không hiển thị giá trị của các biểu thức trung gian, nhưng các biểu thức này vẫn được định giá trị. Biến A được gán giá trị 25.
25	

ONFAILURE

Tham số tùy chọn. Nếu **AutoLISP** gặp phải lỗi trong khi thi hành hàm **Load**, nó sẽ trả về kết quả là giá trị của tham số này.

Ví dụ:

Command: **(load "XYZ" "Error: File not loaded")**↵
"Error: File not loaded"

Hàm **Load** bị lỗi vì không tìm thấy file XYZ.LSP nên trả về chuỗi chứa trong tham số ONFAILURE.

Có nhiều nguyên nhân làm cho hàm **Load** bị lỗi. Tham số ONFAILURE có thể chứa biểu thức tạo ra các hành vi tương ứng với nguyên nhân gây ra lỗi: hoặc hiển thị thông báo lỗi, hoặc cung cấp đường dẫn thư mục khác để tìm kiếm file...

2.1.6 Các dòng chú thích

Trong chương trình, ta nên cung cấp các dòng chú thích để chương trình dễ hiểu, dễ theo dõi và dễ sửa lỗi.

Tất cả các ký tự đứng phía bên phải dấu chấm phẩy (;) cho đến hết một dòng đều được xem là chú thích. Các chú thích có thể bắt đầu từ vị trí đầu dòng hoặc đứng phía sau biểu thức.

Ví dụ:

```
; Tên file: SAMPLE.LSP
; Người viết: Nguyễn Thanh Trung
; Chương trình này gán cho biến n giá trị bằng 1
; sau đó nhân cho 2.
(setq n 1)              ; Gán n bằng 1
(setq n (* n 2))        ; Gán  n bằng 2n
;Kết thúc file
```

Các chú thích có thể đứng giữa một biểu thức, bắt đầu bằng ký hiệu **;|** và kết thúc bằng ký hiệu **|;**.

Ví dụ:

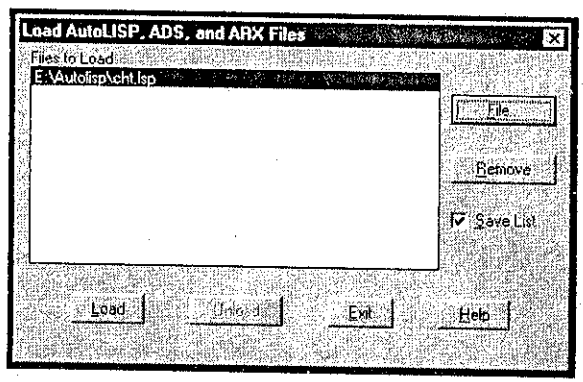
```
(setq R1    ;| Gán giá trị cho R1 |;  10.25)

(setq R2  230.5)  ;| Gán giá trị cho R2
Đây là lực tác dụng lên khung giàn
gây ra biến dạng |;

(setq R3  ;| Gán  giá trị cho R3  |;  5.75)
```

2.1.7 Gọi thực thi chương trình .LSP bằng lệnh Appload

Ngoài hàm **Load** ta còn sử dụng lệnh **Appload** để tải các ứng dụng **AutoLISP**, **ADS**, và **ARX**. Nếu bạn muốn sử dụng các ứng dụng mà không cần tải tự động khi thực hiện chương trình **AutoCAD** thì bạn phải tải chúng vào bằng cách chọn file cần thiết và chọn nút *Load* trên hộp thoại **Load AutoLISP, ADS, and ARX Files**.



Tóm tắt:

1. Hàm **Load** dùng để thi hành file chương trình. Tham số FILENAME là một chuỗi chứa trong cặp dấu nháy chuỗi " ".
2. **AutoLISP** chỉ hiển thị giá trị của biểu thức cuối cùng trong file chương trình.
3. Mọi ký tự nằm phía sau dấu chấm phẩy (;) đều là các chú thích.
4. Chương trình nên được chú thích đầy đủ để dễ kiểm tra lỗi và sửa chữa.
5. Các chú thích nên bổ sung ngay sau khi chương trình được viết xong.

2.2 Các hàm tự tạo

AutoLISP cho phép ta tự tạo ra các hàm mới, nhờ đó ta có thể kết hợp nhiều hàm **AutoLISP** thành một hàm duy nhất. Các hàm tự tạo có thể thực hiện các chức năng như yêu cầu người sử dụng nhập giá trị cho các

tham số, in thông tin ra màn hình, tạo hoặc hiệu chỉnh các đối tượng **AutoCAD**, tạo các lệnh **AutoCAD** mới.

2.2.1 Nhập giá trị cho tham số (hàm Getpoint)

Một trong những ưu điểm của **AutoLISP** so với *script file* hoặc *menu macro* là khả năng **nhập giá trị cho tham số**. Ví dụ, chương trình **AutoLISP** có chức năng vẽ lỗ khoan sẽ yêu cầu người sử dụng nhập giá trị cho các tham số là kích thước và vị trí lỗ khoan.

Hàm GETPOINT

Hàm **Getpoint** yêu cầu người sử dụng nhập vào tọa độ một điểm bằng một trong các phương pháp nhập tọa độ điểm của **AutoCAD**.

(**Getpoint** [PT] [PROMPT])

Hàm **Getpoint** không tham số sẽ dừng quá trình tính toan (mà không hiện thông báo) cho đến khi người sử dụng nhập vào một điểm hoặc thoát khỏi hàm này bằng phím ESC.

Ví dụ:

Command: **(Getpoint)**↵
(4.54783 6.98083 0.0)

Dừng lại đợi nhập tọa độ một điểm. Trả về tọa độ điểm nhập vào X = 4.54783, Y = 6.98083, Z = 0.0. Luôn luôn trả về toa độ điểm 3D, ngay cả khi ta nhập vao tọa độ điểm 2D.

Hàm **Getpoint** trả về giá trị ở dạng một danh sách. Danh sách này gọi là *danh sách lưu trữ dữ liệu (data storage list)*. Loại danh sách này khác với biểu thức ở chỗ phần tử đầu tiên trong danh sách không phải là tên hàm. Để **AutoLISP** không xem danh sách này là một biểu thức, và do đó không đinh giá trị danh sách này, ta dùng hàm **Quote** (hoặc dấu ').

Chú ý

Trong **AutoLISP** các tọa độ x,y,z của một điểm ngăn cách nhau bằng *khoảng trắng*, chứ không phải bằng dấu phẩy như trong **AutoCAD**.

Ví dụ:

Command: **(2.0 2.0 0.0)**↵
Error: bad function

Danh sách này được xem như biểu thức, nhưng phần tử đầu tiên không phải là tên hàm. Do đó xuất hiên lỗi.

(2.0 2.0 0.0)
Cancel
Command :**(quote (2.0 2.0 0.0))** ↵
(2.0 2.0 0.0)

Danh sách này được xem như là một giá trị.

Command: **'(2.0 2.0 0.0)** ↵
(2.0 2.0 0.0)

Dùng dấu ' thay cho hàm **Quote**. Cách này thuân tiện hơn vì sử dụng dấu ngoặc ít hơn, tuy nhiên không dùng được tại dòng nhắc lệnh của **AutoCAD**.

Tham số PT

Khi dùng lệnh **Line** của **AutoCAD**, trên màn hình xuất hiện "*sợi dây thun*" (*rubberband line*) nối điểm gốc thứ nhất của đoạn thẳng sắp vẽ với vị trí con trỏ trên màn hình. Khi ta dịch chuyển con trỏ, sợi dây thun dịch chuyển theo và nó biến mất khi tọa độ điểm thứ hai được đưa vào.

Tham số PT cung cấp tọa độ điểm gốc thứ nhất của sợi dây thun như trên.

Ví dụ:

Command: **(getpoint '(2.0 2.0 0.0))**↵
(4.11421 5.21345 0.0)

AutoLISP dừng lại đợi người sử dụng nhập tọa độ một điểm. Trên màn hình xuất hiện sợi dây thun có điểm gốc là (2.0,2.0,0.0)

Command: **(setq PT1 (getpoint))**↵
(4.11421 5.21345 0.0)

Biến PT1 được gán giá trị toa độ điểm nhập vào.

Command: **(getpoint PT1)**↵
(6.30445 2.11894 0.0)

Sợi dây thun đi qua điểm gốc có tọa độ chứa trong biến PT1.

Tham số PROMPT

Xuất hiện dòng nhắc tại cửa sổ lệnh.

Ví dụ:

Command: **(getpoint "Pick a point: ")**↵
Pick a point: **4,2** ↵
(4.0 2.0 0.0)

Dòng nhắc *Pick a point:* xuất hiện tại cửa sổ lệnh, đợi người sử dụng nhập vào tọa độ điểm.

Command: **(getpoint '(4 4) "Select next point: ")**↵
Select next point: **5,5** ↵

Dòng nhắc *Select next point:* xuất hiện

(5.0 5.0 0.0)

tại cửa sổ lệnh. Sợi dây thun đi qua điểm gốc (4,4,0).

Command: **(getpoint "First point: ")(getpoint "\nSecond point: ")** ↵
First point: (Chọn điểm thứ nhất)
Second point: (Chọn điểm thứ hai)

Sử dụng ký tự xuống dòng \n

Chú ý

Ta nên sử dụng sợi dây thun và các dòng nhắc tương tự như trong **AutoCAD** để người sử dụng dễ làm quen với chương trình **AutoLISP** do chúng ta viết.

2.2.2 Hiện thông báo lên màn hình

Hàm **Prompt** dùng để hiện thông báo lên màn hình. Những thông báo này có thể là các dòng thông tin, các dòng nhắc nhở người sử dụng nhập số liệu...

(**Prompt** MESSAGE)

Hàm **Prompt** hiện dữ liệu kiểu *chuỗi* chứa trong tham số MESSAGE và trả về giá trị *nil*.

Ví dụ:

Command: **(prompt "OK")** ↵
Oknil

Hiện chuỗi "OK" và trả về *nil*.

Command: **(prompt "Select point: ") (getpoint)** ↵
Select point: **4,3.5** ↵
(4.0 3.5 0.0)

Chỉ trả về kết quả biểu thức cuối cùng là tọa độ điểm nhập vào, do đó *nil* không xuất hiện

Command: **(setq E "HELLO")** ↵
"*HELLO*"
Command: **(prompt E)** ↵
HELLOnil

Hiện nội dung chứa trong biến E và trả về *nil*.

Command: **(prompt)**↵
Error: too few arguments
(PROMPT)

Hàm **Prompt** không có tham số nên gây ra lỗi.

Command: **(prompt "OK" "ABC")** ↵
Oknil

Khi có nhiều tham số, hàm **Prompt** chỉ sử dụng tham số đầu tiên.

Do hàm **Prompt** thiếu tính linh hoạt nên trong thực tế ta thường dùng hàm **Princ** (sẽ được trình bày ở chương 7). Cú pháp hàm này tương tự như hàm **Prompt**:

(**Princ** STRING)

Hàm **Princ** hiện chuỗi thông báo STRING lên màn hình và trả về kết quả là chuỗi thông báo (không trả về nil như hàm **Prompt**). Khi dùng hàm **Prompt** ở cuối chương trình, nó sẽ trả về giá trị nil. Ta có thể thay thế bằng hàm **Princ** không tham số (giá trị trả về là null, không xuất hiện gì trên màn hình):

(**Princ**)

Ví dụ: Sửa lại file chương trình **ABC.LSP** ở trên.

```
;Tên file: ABC.LSP
(setq A (/ 100 4))
(setq B (+ A 75))
(setq C (/ B A))
(princ)
;Kết thúc chương trình
```

2.2.3 Sử dụng các lệnh AutoCAD trong AutoLISP

Ta có thể dùng hàm **Command** để thực hiện các lệnh của **AutoCAD**.

(**Command** [ARGUMENT] . . .)

Trình tự các tham số của hàm **Command** tương ứng với trình tự nhập lệnh tại dòng nhắc lệnh của **AutoCAD**.

Ví dụ:

Command: **(command "LINE" "2,2" "4,4" "")**↵
Line From point: **2,2** ↵
To point: **4,4** ↵
To point:

Hàm **Command** thực hiện lệnh **Line** tại dòng nhắc lệnh. Tiếp theo là các toạ độ điểm 2,2 và 4,4 Chuỗi

Command: nil

rỗng "" tương đương với phím ENTER sẽ kết thúc lệnh **Line**. Hàm **Command** trả về giá trị *nil*.

Command: **(setq PT1 '(4 2))** ↵
Command: **(command: "LINE" '(6 4) PT1 '(4 6) "C")** ↵
LINE From point:
To point:
To point:
To point: **C** ↵
Command: nil

Có thể sử dụng danh sách '(4 2) thay cho chuỗi "4,2". Sử dụng biến PT1 để cung cấp toạ điểm. "C" dùng để đóng (close) lệnh **Line**. Hàm **Command** trả về giá trị *nil*.

✌ **Ví dụ:**

(command "circle" '(1 1 0) 2.5) Vẽ đường tròn có tâm tại (1,1,0) và bán kính bằng 2.5.

(command "circle") Bắt đầu lệnh vẽ đường tròn, người sử dụng tự thực hiện phần còn lại của trình tự vẽ.

(command ".arc") Bắt đầu lệnh vẽ cung tròn, người sử dụng tự thực hiện phần còn lại của trình tự vẽ. Khi có dấu chấm đặt trước tên lệnh, chương trình sẽ thực hiện lệnh gốc của **AutoCAD**, chứ không thực hiện các lệnh đã được sửa đổi lại và có cùng tên với lệnh gốc (ví dụ lệnh viết tắt hoặc hàm tự tạo trong **AutoLISP**...).

(command ".line" pause pause "") Bắt đầu lệnh **Line** (lệnh gốc) của **AutoCAD**, sau đó dừng lại 2 lần để người sử dụng nhập vào 2 điểm, rồi kết thúc lệnh. **Pause** là lệnh trong **AutoLISP** dùng để dừng chương trình đợi tín hiệu nhập từ chuột hoặc bàn phím. Không được đặt pause trong cặp dấu ngoặc kép.

(command "_circle" '(2 2) "_d" 4) Vẽ đường tròn có tâm tại điểm (2,2) và có đường kính bằng 4. Ngoài phiên bản tiếng Anh, **AutoCAD** còn có các phiên bản dành cho các ngôn ngữ khác. Khi đang sử dụng các loại phiên bản này, dấu gạch dưới (_) đặt trước tên lệnh hoặc các lựa chọn có tác dụng tự động phiên dịch chúng về tiếng Anh.

(Command ".line" pause pause "" "_circle" '(2 2) "_d" 4)
Trong biểu thức **Command** có thể sử dụng nhiều lệnh **AutoCAD**.

(command "") Tương đương nhấn phím ENTER tại dòng nhắc lệnh.

(command) Tương đương nhấn phím ESC tại dòng nhắc lệnh để kết thúc lệnh đang thực hiện.

(command ".dim" ".leader" PT1 PT2 ^C ^C ".insert" . . .)
^C tương đương việc nhấn phím ESC thoát khỏi lệnh đang thực hiện. Trong ví dụ này, để thoát khỏi lệnh **Dim** ta phải nhấn phím ESC 2 lần, sau đó tại dòng nhắc lệnh xuất hiện lệnh **Insert** tiếp theo.

✌ **Ví dụ:** Về cách dùng **Pause**.

Command: **(Command ".line")**↵
From point:nil
From point: (nhập điểm thứ nhất)
To point: (nhập điểm thứ hai)
To point:↵
Command:

Biểu thức *(Command ".line")* trả về giá trị nil, sau đó mới thực hiện tiếp tục lệnh **Line**. Vì nil không phải là toạ độ điểm nên xuất hiện lại dòng nhắc *"From point:"*

Command: **(command ".line" pause)**↵
From point: (nhập điểm thứ nhất)
To point: nil
To point: (nhập điểm thứ hai)
To point:↵
Command:

Biểu thức *(Command ".line")* tạm hoãn trả về giá trị nil một lần. Sau khi nhập điểm thứ nhất tại dòng nhắc *"From point:"*, giá trị nil được trả về cho dòng nhắc *"To point:"* Vì nil không phải là toạ độ điểm nên xuất hiện lại dòng nhắc *"To point:"*.

Command: **(command ".line" pause pause)**↵
From point: (nhập điểm thứ nhất)
To point: (nhập điểm thứ hai)
To point: nil

Biểu thức *(Command ".line")* tạm hoãn trả về giá trị nil 2 lần. Sau khi nhập điểm thứ nhất và điểm thứ

To point: ↵
Command:

hai, giá trị nil được trả về cho dòng nhắc *"To point:"*. Vì nil không phải là tọa độ điểm nên xuất hiện lại dòng nhắc *"To point:"*. Khi đó ta nhập ENTER để kết thúc lệnh.

Command: **(command ".line" pause pause "")**↵
From point: (nhập điểm thứ nhất)
To point: (nhập điểm thứ hai)
To point:
Command:nil
Command:

Biểu thức *(Command ".line")* tam hoãn trả về giá trị "" 2 lần. Sau khi nhập điểm thứ nhất và điểm thứ hai, giá trị "" được cung cấp cho dòng nhắc *"To point:"* và do đó kết thúc lệnh **Line**. Sau đó giá trị trả về nil của hàm sẽ cung cấp cho dòng nhắc lệnh *"Command:"*

Kết hợp các hàm **AutoLISP** vào trong một file chương trình, ta có thể tạo ra được các chương trình thực hiện các nhiệm vụ phức tạp.

Ví dụ: Chương trình vẽ đường thẳng đi qua 2 điểm:

```
; 1LINE.LSP
; Yêu cầu người sử dụng nhập vào 2 điểm. Vẽ đường thẳng qua 2 điểm
;này.
(prompt "\n This program draws a line between two points ..."); Thông báo
(setq PT1 (getpoint "\n Enter the starting point: "))        ; Nhập điểm PT1
(setq PT2 (getpoint PT1 "\n Enter the endding point: "))  ; Nhập điểm PT2
(command "LINE" PT1 PT2 "")          ; Vẽ đường thẳng qua PT1 và PT2
(prompt "\nDone.\n")                                  ;Thông báo kết thúc
(princ)
;Kết thúc file
```

2.2.4 Các hàm tự tạo (hàm Defun)

Mặc dù **AutoLISP** cung cấp sẵn một số hàm, nhưng trong nhiều trường hợp ta cần xây dựng các hàm tự tạo chứa nhiều biểu thức phức tạp để có được kết quả mong muốn. Trước tiên ta định nghĩa hàm tự tạo bằng hàm **Defun** (DEfine FUNction). Sau đó, bất kỳ khi nào **AutoLISP** gặp tên hàm này nó sẽ tính toán và trả về kết quả của hàm.

(**Defun** FUNCTION_NAME ARGUMENT_LIST EXPRESSION ...)

Function_Name

Tên hàm tự tạo. Tên hàm tuân theo các quy định về đặt tên biến.

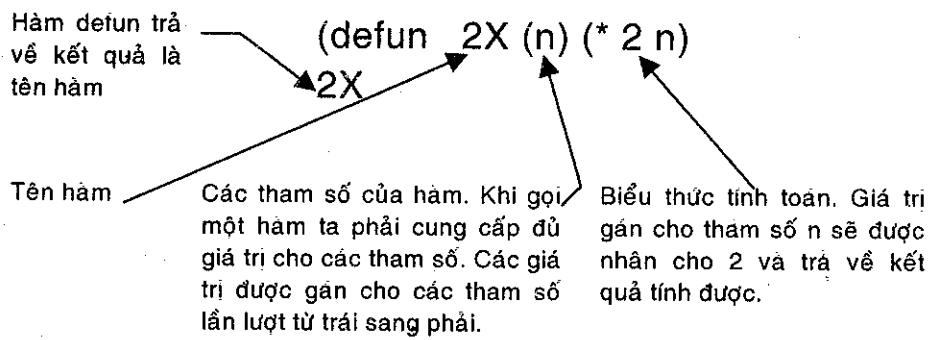
Argument_List

Gồm hai phần ngăn cách nhau bằng dấu /. Phần thứ nhất chứa các tham số cần thiết khi gọi hàm. Phần thứ hai chứa các biến cục bộ của hàm.

Expression

Các biểu thức tính toán của hàm. Khi hàm được gọi, các biểu thức này lần lượt được tính toán.

Ví dụ: Tạo hàm **2X** có chức năng nhân đôi giá trị tham số đưa vào.



Sau khi đã định nghĩa hàm 2X, ta gọi hàm này để tính toán tương tự như các hàm **AutoLISP**:

Command: **(2X 10.3)** ↵
20.6

Ví dụ: Tạo hàm **2XADD** có hai tham số. Hàm này sẽ nhân tham số thứ nhất cho 2, sau đó cộng với tham số thứ hai, và trả về kết quả tính được.
Command: **(defun 2XADD (n1 n2) (+ (* n1 2) n2))** ↵
2XADD
Command: **(2XADD 36 15.0)**↵
87.0

Số nguyên 36 được gán cho tham số n1. Số thực 15.0 được gán cho tham số n2. Kết quả trả về là 87.0.

Ví dụ:

(defun ZA () (command "ZOOM" "ALL")) Hàm tự tạo ZA không có tham số và biến cục bộ.

2.2.5 Biến toàn cục và biến cục bộ

Biến toàn cục là các biến vẫn giữ nguyên giá trị trong phạm vi bản vẽ hiện hành. *Biến cục bộ* là các biến được định nghĩa trong phạm vi một hàm và giá trị của nó sẽ mất đi khi việc gọi hàm kết thúc. Trong phần lớn các trường hợp ta nên dùng các biến cục bộ để dễ quản lý.

✌ **Ví dụ:** Tạo file **2X.LSP** như sau:

```
(setq W 14)                          W là biến toàn cục được gán giá trị
(defun 2X (n / w)                    14. Ngoài ra có một biến cục bộ
  (setq w "2X the value is:")        khác cũng có tên là W được khai
  (prompt w)                         báo trong hàm 2X
  (* 2 n)
)
```

Sau đó tải file này và gọi hàm 2X như sau:

```
Command: (load "2X.lsp").↵           Tải file chương trình 2X.LSP
2X
Command: (2X 3).↵                    Gọi hàm 2X. Giá trị 3 được truyền
2X the value is: 6                   cho tham số n. Giá trị biến cục bộ w
                                     là chuỗi "2X the value is: 6.
Command: !w .↵                       Biến toàn cục w không bị thay đổi
14                                   giá trị 14.
```

2.2.6 Tạo các lệnh AutoCAD mới

Ta có thể sử dụng một trong hai đặc điểm sau để áp dụng cho các hàm tự tạo: **C: Option** và **S::Startup Option**.

C: Option

Để có thể sử dụng các hàm tự tạo tương tự như các lệnh của **AutoCAD**, ta đặt ký hiệu **C:** vào trước tên hàm trong phần định nghĩa hàm tự tạo.

✌ **Ví dụ:** Xây dựng hàm **1LINE** dùng để vẽ đường thẳng đi qua 2 điểm (dùng ký hiệu C: trước tên hàm trong định nghĩa hàm)

```
; 1LINE.LSP
; Yêu cầu người sử dụng nhập vào 2 điểm. Vẽ đường thẳng qua 2 điểm
;này.
(defun C:1LINE (/ PT1 PT2)                    ;định nghĩa hàm C:1LINE
```



```
  (prompt "\n This program draws a line between two points ...")
  (setq PT1 (getpoint "\n Enter the starting point: "))  ; Nhập điểm PT1
  (setq PT2 (getpoint PT1 "\n Enter the endding point: ")); Nhập điểm PT2
  (command "LINE" PT1 PT2 "")          ; Vẽ đường thẳng qua PT1 và PT2
  (prompt "\nDone.\n")                 ;Thông báo kết thúc
  (princ)
) ;Kết thúc file
```

Ta có thể tải file chương trình này và gọi lệnh **1Line** tương tự như lệnh của **AutoCAD**.

```
Command: (load "1LINE").↵
C:1LINE
Command: 1LINE ↵
This program draws a line between two points ...
Enter the starting point: (Chọn một điểm)
Enter the endding point: (Chọn một điểm)
Done.
```

Ta có thể tiếp tục gọi hàm **1Line** này bất kỳ lúc nào cần. Nhưng nếu đóng chương trình **AutoCAD** lại, **AutoCAD** sẽ không ghi nhớ lại lệnh này. Để sử dụng nó khi mở lại các bản vẽ khác, ta phải tải file chương trình **1LINE.LSP** bằng hàm **Load**.

Trong **AutoCAD Release 14**, có một file **ACADR14.LSP** được tự động tải mỗi khi mở chương trình **AutoCAD**. Nếu trong file này ta xây dựng một số hàm tự tạo, thì các hàm này sẽ tự động được tải, ta không cần phải tải bằng hàm **Load**.

Sau đây là đoạn cuối cùng trong file **ACADR14.LSP** đã bổ sung phần định nghĩa hàm **1Line**:

```
;; Save current filedia & cmdecho setting.
 (setq filedia-save (getvar "FILEDIA"))
 (setq cmdecho-save (getvar "CMDECHO"))
 (setvar "FILEDIA" 0)
 (setvar "CMDECHO" 0)

;; Call 3DSIN and pass in filename.
 (c:3dsin 1 filename)

;; Reset filedia & cmdecho
```

```
  (setvar "FILEDIA" filedia-save)
  (setvar "CMDECHO" cmdecho-save)
  (princ)
)
;; Silent load.
(princ)

;; The following line conditionally loads AutoLISP routines for AutoCAD
Release 14
;; Altering this line will affect bonus functionality
(load "bonus.lsp" "")

;
; 1LINE.LSP
;  Yêu cầu người sử dụng nhập vào 2 điểm. Vẽ đường thẳng qua 2 điểm
;này.
(defun C:1LINE (/ PT1 PT2)                    ;định nghĩa hàm C:1LINE
  (prompt "\n This program draws a line between two points ...")
   (setq PT1 (getpoint "\n Enter the starting point: ")) ; Nhập điểm PT1
   (setq PT2 (getpoint PT1 "\n Enter the endding point: ")); Nhập điểm PT2
   (command "LINE" PT1 PT2 "")       ; Vẽ đường thẳng qua PT1 và PT2
   (prompt "\nDone.\n")                          ;Thông báo kết thúc
   (princ)
)  ;Kết thúc file
```

Khi mở lại chương trình **AutoCAD**, file ACADR14.LSP được tự động tải và ta có thể gọi hàm **1Line** tương tự như lệnh của **AutoCAD**.

Ta có thể tạo các lệnh tắt để gọi các lệnh của **AutoCAD** trong file *acad.pgp*. Ví dụ, tạo lệnh tắt "Z" để gọi lệnh "ZOOM". Tuy nhiên các lệnh tắt chỉ có thể gọi các lệnh cơ bản của **AutoCAD**. Ví dụ, ta không thể tạo ra lệnh ZOOM Previous theo cách này. Dùng hàm tự tạo của **AutoLISP** có thể giải quyết vấn đề này:

Ví dụ:

```
(defun C:ZP () (command "ZOOM" "Previous"))
```

```
Command: ZP ↵
nil
```

Để không xuất hiện nil trên màn hình, ta định nghĩa lại hàm này như sau:

```
(defun C:ZP () (command "ZOOM" "Previous") (princ))
```

S::Startup Option

Khi khởi động **AutoCAD**, hàm **S::Startup** định nghĩa trong file ACADR14.LSP sẽ được tự động gọi thi hành. Đây là hàm *duy nhất* có tính chất này. Vì được thực hiện tự động, nên nó không được chứa bất kỳ tham số nào. Thông thường hàm này được dùng để thực hiện các thao tác khởi tạo bản vẽ.

Ví dụ:

```
(defun S::Startup ()
  (command ".limits" '(0 0) '(420 297)          ;Gán LIMITS
           ".zoom" "All"                        ;ZOOM All
           ".layer" "make" "Border" ""          ;Tạo lớp "Border"
           ".pline" '(0.5 0.5) '(419.5 0.5)     ;Điểm 1st và 2nd
            '(419.5 296.5) '(0.   96.5) "c"     ; Điểm 3rd và 4th
           ".layer" "make" "Draw" ""            ;Tạo lớp "Draw"
            ".text" '(1 1) 7 0.0 "PROJECT-1"    ;Nhập text
  )                                             ;Đóng COMMAND
)                                               ;Kết thúc STARTUP
```

Tóm tắt:

1. Một ưu điểm lớn của chương trình **AutoLISP** so với *script file* và *menu macro* là cho phép người sử dụng nhập giá trị cho các tham số của chương trình.
2. Hàm **Getpoint** yêu cầu nhập vào một điểm và trả về tọa độ điểm ở dạng danh sách.
3. Hàm **Getpoint** có thể làm xuất hiện dòng nhắc và sợi dây thun trên màn hình nếu ta sử dụng các tham số thích hợp.
4. Hàm **Quote** tạo ra một danh sách không định giá trị (không cho **AutoLISP** định giá trị các biểu thức).
5. Hàm **Prompt** làm xuất hiện dòng thông báo trên màn hình, và trả về giá trị *nil*.
6. Hàm **Defun** định nghĩa các hàm tự tạo.

7. Các hàm tự tạo và các biến sẽ không được ghi nhớ khi thoát khỏi chương trình **AutoCAD**. Khi mở lại chương trình **AutoCAD**, ta phải định nghĩa lại chúng.
8. Các tham số và các biến cục bộ chỉ có giá trị trong phạm vi một hàm, và sẽ bị mất đi khi việc gọi hàm kết thúc.
9. Việc sử dụng **C:Option** cho phép hàm tự tạo được gọi tương tự như cách gọi lệnh của **AutoCAD**.
10. Hàm **S::Startup** định nghĩa trong file ACADR14.LSP sẽ được tự động gọi thực hiện mỗi khi khởi động **AutoCAD**.

2.3 Bài tập

1. Hãy thêm các dòng chú thích cho file ABC.LSP.
2. Dùng hàm **Load** để chạy chương trình ABC.LSP sau khi đã thêm các dòng chú thích để kiểm tra các chú thích thêm vào có gây ra lỗi không.
3. Viết các chương trình thực hiện các chức năng sau (nhớ ghi chú thích đầy đủ):

a. Dùng hàm **Getpoint** không tham số để nhập toạ độ cho hai điểm PT1 và PT2. Tên file là GPT1.LSP
b. Dùng hàm **Getpoint** với tham số PROMPT để nhập tọa độ cho hai điểm PTA và PTB. Tên file là GPT2.LSP
c. Dùng hàm **Getpoint** với các tham số PT và PROMPT để nhập toạ độ cho hai điểm P2 và P1. Tên file là GPT3.LSP
d. Trong 3 chương trình trên, chương trình nào cung cấp giao diện tốt nhất?

4. Hãy sửa lại chương trình GPT3.LSP sao cho dòng thông báo sau xuất hiện trước khi yêu cầu nhập tọa độ hai điểm:
   "This program will set the values for points P1 and P2."
5. Viết chương trình cho phép người sử dụng nhập vào tọa độ 3 đỉnh của một tam giác. Sau đó vẽ tam giác bằng đường polyline. Đặt tên file là TRIANGLE.LSP
6. Tạo file chương trình có tên CHAP2P1 thực hiện các chức năng sau:

a. Tạo một lệnh có tên EL (erase last) dùng để xóa đối tượng được vẽ cuối cùng, sau đó hiện thông báo kết thúc.
b. Tạo một lệnh có tên C3 dùng để vẽ đường tròn đi qua 3 điểm. Các điểm này được nhập vào bằng hàm **Getpoint**.
c. Hiển thị thông báo khi file chương trình được tải xong.

7. Tạo một hàm C:QUA, yêu cầu nhập vào 4 điểm, sau đó vẽ một tứ giác bằng đường đa tuyến đóng (closed polyline) đi qua 4 điểm này. Tên file chương trình là QUA.LSP

8. Tạo một hàm có tên SQ dùng để tính lũy thừa 2 của một số.

9. Tạo ít nhất 15 lệnh viết tắt và đưa chúng vào trong file ACADR14.LSP Nhớ chú thích đầy đủ ý nghĩa các lệnh.

2.4 Lời giải

3.

a.
```
(setq PT1 (getpoint))
(setq PT2 (getpoint))
```

b.
```
(setq PTA (getpoint "Select first point: "))
(setq PTB (getpoint "\nSelect second point: "))
```

c.
```
(setq PTA (getpoint "Select first point: "))
(setq PTB (getpoint PTA "\nSelect second point: "))
```

4.
```
(prompt "This program will set the values for points P1 and P2.")
(setq PTA (getpoint "\nSelect first point: "))
(setq PTB (getpoint PTA "\nSelect second point: "))
```

5.
```
;Tên file: TRIANGLE.LSP
, Vẽ tam giác qua 3 đỉnh do người sử dụng nhập vào
(prompt "\n This program draws a triangle ...")
(setq PT1 (getpoint "\n Enter the first point: "))          ; Điểm PT1
(setq PT2 (getpoint PT1 "\n Enter the second point: "))    ; Điểm PT2
(setq PT3 (getpoint PT2 "\n Enter the third point: "))     ; Điểm PT3
(command "PLINE" PT1 PT2 PT3 PT1 "")                       , Vẽ tam giác
(prompt "\nDone.\n")                                ;Thông báo kết thúc
(princ)
```

6.
```
;Tên file: CHAP2P1.LSP
(defun C:EL () (command ".erase" "last" ""))
(prompt "\nDone.\n")
(defun C3 (/ PT1 PT2 PT3)
```

```
        (setq PT1 (getpoint "\nFirst point: "))
        (setq PT2 (getpoint "\nSecond point: "))
        (setq PT3 (getpoint "\nThird point: "))
        (command ".circle" "3P" PT1 PT2 PT3)
)
(prompt "File CHAP2P1.LSP has been loaded")
(princ)
```

7.

```
; Tên file: QUA.LSP
(defun C:QUA (/ PT1 PT2 PT3 PT4)
  (setq PT1 (getpoint "\n Enter the first point: "))          ; Điểm PT1
  (setq PT2 (getpoint PT1 "\n Enter the second point: "))   ; Điểm PT2
  (setq PT3 (getpoint PT2 "\n Enter the third point: "))     ;Điểm PT3
  (setq PT4 (getpoint PT2 "\n Enter the fourth point: "))    ;Điểm PT4
  (command "PLINE" PT1 PT2 PT3 PT4 "close")                  ;Vẽ tứ giác
)
```

8.

```
;Tên file: SQ.LSP
;
(defun SQ (n) (* n n))
```

Chương 3

XỬ LÝ DANH SÁCH

Nội dung chương

1. Phân loại danh sách.
2. Tạo danh sách: hàm **List.**
3. Tạo ra các tọa độ điểm mới từ tọa độ điểm có sẵn. Các hàm xử lý danh sách: **Car, Cdr, Cadr, Caddr, Cadar, Cddr, Cadddr, Getcorner, Last, Length.**

3.1 Danh sách

3.1.1 Phân loại

Danh sách (List) được phân thành 3 loại chính:

- *Biểu thức (Expression list)*: chứa tên hàm và các tham số của hàm. Biểu thức cần được định giá trị.

- *Tọa độ điểm (Point Coordinate List)*: có hàm **Quote** (hoặc dấu ') ở phía trước, chứa tọa độ X,Y hoặc X,Y,Z của một điểm. Đây là trường hợp đặc biệt của danh sách kho dữ liệu, trong đó thông tin lưu trữ là tọa độ điểm.

- *Kho dữ liệu (Data Storage List)*: có hàm **Quote** (hoặc dấu ') ở phía trước và có thể chứa bất kỳ kiểu dữ liệu nào.

3.1.2 Tạo danh sách

Khi viết chương trình ta cần phải quản lý dữ liệu chặt chẽ và hiệu quả. Ta có thể lưu từng dữ liệu vào từng biến (cục bộ hoặc toàn cục). Tuy nhiên cách này sẽ không hiệu quả khi số lượng dữ liệu tăng lên, kéo theo việc phải dùng nhiều biến để lưu trữ. Vì vậy, trong trường hợp này ta phải lưu trữ dữ liệu vào các danh sách *(data storage list)*. Một danh sách dù chứa nhiều phần tử *(element)* vẫn có thể gán cho một biến, nhờ đó làm đơn giản việc quản lý và truy cập dữ liệu.

Một trong các phương pháp tạo ra danh sách là dùng hàm **List**:

(**List** EXPRESSION ...)

✌ **Ví dụ:**

Command: **(setq L1 (list "FLOOR1" "TABLE" 37.6 '(20 20 0) "CHAIR"))**
("FLOOR1" "TABLE" 37.6 (20 20 0) "CHAIR")

Trong ví dụ trên, danh sách L1 có 5 phần tử. Các phần tử này có các kiểu dữ liệu khác nhau:

"FLOOR", "TABLE" và "CHAIR" : kiểu chuỗi
37.6 : kiểu số thực
'(20 20 0) : là danh sách không định giá trị (dùng dấu ' hoặc hàm **Quote** ở phía trước) chứa 3 phần tử là 3 số thực (biểu diễn tọa độ điểm).

Trước khi tạo ra danh sách, hàm **List** sẽ định giá trị cho các tham số của mình (ngoại trừ tham số là danh sách không định giá trị).

✌ **Ví dụ:**

Command: **(setq L2 (list (+ 14 24) (getpoint "\nPick a point: ")))**↵
Pick a point: (nhập tọa độ một điểm)
(38 (20 20 0))

Danh sách gồm 2 phần tử: phần tử thứ nhất là giá trị biểu thức cộng, phần tử thứ hai là tọa độ điểm.

Command: **(setq N 4 S "String")**↵
"String"
Command: **(setq L3 (list N S))**↵
(4 "String")

Danh sách gồm hai biến N và S. Các biến này sẽ được định giá trị trước khi tạo ra danh sách gán cho biến L3.

Command: **(setq L4 (list N 'S))**↵
(4 S)

Biến S không được định giá trị (vì có dấu ' ở phía trước). Danh sách tạo ra chứa *biến* S chứ không phải *giá trị* của biến S.

Command: **(setq L5 '((getpoint "\nPick a point: ")(+ 30 40)))**↵
(getpoint "\nPick a point: ")(+ 30 40))

Dùng hàm **Quote** để không định giá trị các phần tử khi tạo ra danh sách. Hãy so sánh với danh sách gán cho biến L2.

Command: **(setq '(N S))**↵
(N S)

Danh sách tạo ra chứa các biến chưa định giá trị. Khả năng lưu trữ dữ liệu ở trạng thái chưa định giá trị giúp ta quản lý dữ liệu linh hoạt hơn.

Tóm tắt

1. Có 3 loại danh sách: biểu thức, tọa độ điểm và kho dữ liệu.
2. Bằng cách dùng danh sách, nhiều dữ liệu khác nhau có thể gộp thành một nhóm và lưu trữ trong một biến duy nhất.
3. Hàm **Quote** trả về một phần tử chưa định giá trị.
4. Tọa độ điểm là một danh sách có 2 hoặc 3 phần tử kiểu số thực.

3.2 Các hàm xử lý danh sách cơ bản

Sau khi tập hợp nhiều dữ liệu vào trong một danh sách, khi cần thiết ta có thể truy cập đến từng phần tử trong danh sách.

Hàm CAR

Hàm **Car** dùng để trích ra phần tử đầu tiên của danh sách.

(**Car** LIST)

Ví dụ:

Biểu thức **AutoLISP**	Kết quả
(car (list "A" "B" "C"))	"A"
(car '(D E F))	D
(car (list 7.59 6.38 4.81))	7.59
(car '())	Nil
(setq HUE (car '("RED" "BLUE")))	"RED"

Ví dụ: Lấy ra tọa độ X từ toạ độ một điểm.

Command:**(setq PTA (getpoint "\nPick any location in your drawing: "))**↵
Pick any location in your drawing: **(70.6 85.24 32.15)** ↵
Command: **(car PTA)**↵
70.6

Ví dụ:

Sau đây là một hàm tự tạo dùng để lấy ra tọa độ X của một điểm.

```
;Tên file: XCOORD.LSP
;Chương trình sau đây yêu cầu nhập vào một điểm,
;sau đó lấy ra thành phần toạ độ X của điểm này.
(defun xcoord (/ PTA)
(setq PTA (getpoint "\n Pick any location in your drawing:-"))
   (prompt "\nThe X-axis coordinate value is: ")
   (car PTA)
);Kết thúc file
```

Hàm CDR

Hàm **Cdr** tạo ra một danh sách từ một danh sách gốc bằng cách loại bỏ phần tử đầu tiên trong danh sách gốc chỉ lấy các phần tử còn lại.

(**Cdr** LIST)

Ví dụ:

Biểu thức **AutoLISP**	Kết quả
(cdr (list "A" 5.5 "10.75"))	(5.5 "10.75")
(cdr '((D E F G) H I))	(H I)
(cdr '())	Nil
(setq YZ (cdr '("x" "y" "z")))	("y" "z")

Hàm **Cdr** thường dùng để loại bỏ phần tử đầu tiên (không dùng đến) của một danh sách.

Ví dụ:

Command:**(setq ROOF '("SHINGLES" "SHAKES" "ALUMINIUM"))**↵
("SHINGLES" "SHAKES" "ALUMINIUM")
Command: **(setq ROOF (cdr ROOF))**↵
("SHAKES" "ALUMINIUM")

Trong ví dụ trên, biến ROOF được gán lại giá trị mới. Đây là một phương pháp thường được sử dụng khi viết chương trình. Nhờ đó giảm được số lượng các biến và dễ theo dõi giá trị của chúng.

Ví dụ:

Sau đây là chương trình dùng để in tất cả các phần tử của một danh sách bằng lệnh **Text** của **AutoCAD**.

```
;Tên file: ROOFLIST.LSP
(defun C:ROOFLIST (/ ROOF)
  (setq ROOF '("SHINGLES" "SHAKES" "ALUMINIUM"))  ;Tao ra danh
                                                  ;sách ROOF
  (prompt "\nPick text start point: ")
  (command ".TEXT" pause)          ;Bắt đầu lệnh TEXT. Dừng lại để
                                   ;người sử dụng chọn điểm bắt đầu
  (prompt "\nText height: ")
  (command pause 0.0 (car ROOF))   ;Dừng lại để người sử dụng nhập
                                   ;chiều cao chữ. (car ROOF) lấy ra
                                   ;phần tử đầu tiên trong danh sách là
                                   ;"SHINGLES"
  (setq ROOF (cdr ROOF))           ;Tao lai danh sách ROOF
```

```
(command ".text" "" (car ROOF))   ; (car ROOF) lấy ra phần tử đầu tiên
                                  ;trong danh sách là "SHAKES"
(setq ROOF (cdr ROOF))            ;Tao lai danh sách ROOF lần nữa
(command ".text" "" (car ROOF))   ; (car ROOF) lấy ra phần tử đầu tiên
                                  ;trong danh sách là "SHAKES"
)
; Kết thúc chương trình
```

Để thực hiện chương trình này, trước tiên ta phải gán biến hệ thống TEXTEVAL = 1. Điều này cho phép ta nhập biểu thức tại dòng nhắc *Text:* của lệnh **Text** và biểu thức này được đinh giá trị.

Ví dụ:

Command: **Setvar** ↵
Variable name or ?: **TEXTEVAL** ↵
New value for TEXTEVAL <0>: **1** ↵
Command: **(setq ROOF '("SHINGLES" "SHAKES" "ALUMINIUM"))** ↵
("SHINGLES" "SHAKES" "ALUMINIUM")
Command: **Text** ↵
Justify/Style/<Start point> (Chọn một điểm)
Height <0.2000>: **7** ↵
Rotation angle <0>: ↵
Text : **(car ROOF)** ↵

Trả về phần tử đầu tiên "SHINGLES" cho dòng nhắc *Text:* . Nếu biến TEXTEVAL = 0, chuỗi trả về là *"(car ROOF)"*

Trở lại ví dụ ROOFLIST.LSP

Command: **Setvar** ↵
Variable name or ?: **TEXTEVAL** ↵
New value for TEXTEVAL <0>: **1** ↵
Command: **(load "ROOFLIST")** ↵
C:ROOFLIST
Command: **ROOFLIST** ↵
Pick Text start point: (Chọn một điểm)
Text height: (Nhập chiều cao chữ)

Ta có thể phối hợp các hàm **Car** và **Cdr** với nhau để tạo ra các hàm có dạng **CxxxxR**. Xét một vài hàm trong số các ham dạng này.

Hàm CADR

Hàm **Cadr** trả về phần tử thứ hai trong danh sách.

(**Cadr** LIST)

Hàm này kết hợp giữa hai hàm **Car** và **Cdr**: (**Car** (**Cdr** LIST))

Ví dụ:

Biểu thức **AutoLISP**	Kết quả	Giải thích
(cadr '(x y z))	Y	(cdr '(x y z)) trả về (y z) (car '(y z)) trả về y
(setq P (cadr (list "A" "B")))	"B"	(cdr (list "A" "B")) trả về danh sách ("B") (car (list "B")) trả về "B"
(cadr '(-3 (178 29)))	(178 29)	(cdr '(-3 (178 29))) trả về ((178 29)) (car '((178 29))) trả về (178 29)
(cadr (list "v-weld")	nil	(cdr (list "v-weld")) trả về () (car '()) trả về nil

Hàm **Cadr** có thể dùng để lấy ra toạ độ Y của toạ độ một điểm:

Command: **(setq Y (cadr '(1.63 5.74 7.81)))** ↵
5.74

Hàm CADDR

Hàm **Caddr** trả về phần tử thứ ba trong danh sách.

(**Caddr** LIST)

Hàm này kết hợp giữa các hàm **Car** và **Cdr** như sau
(**Car** (**Cdr** (**Cdr** (LIST)))

Hàm này thường được dùng để lấy ra toạ độ Z của một điểm.

Ví dụ:

Biểu thức **AutoLISP**	Kết quả	Giải thích
(caddr '(15 62 96))	96	Lấy ra toạ độ Z của một điểm. (cdr '(15 62 96)) trả về (62 96) (cdr '(62 96)) trả về (96) (car '(96)) trả về 96

(caddr (list "revA" "revB"))	Nil	(cdr (list "revA" "revB")) trả về ("revB") (cdr '("revB")) trả về () (car ()) trả về nil
(caddr (list 5 "x" (list "y" 6)))	("y" 6)	(cdr (list 5 "x" (list "y" 6))) trả về ("x" (list "y" 6)) (cdr '("x" (list "y" 6))) trả về ((list "y" 6)) (car ((list "y" 6))) trả về ("y" 6)

Ta có thể tạo ra nhiều hàm có dạng **CxxxxR**. Tối đa là 4 mức tương ứng với 4 chữ x.

Ví dụ:

Hàm **Cadar** tương đương với biểu thức **(car (cdr (car LIST)))**
Hàm **Cddr** tương đương với biểu thức **(cdr (cdr LIST))**
Hàm **Cadddr** tương đương với biểu thức **(car (cdr (cdr (cdr LIST))))**

Hàm GETCORNER

Tương tự như hàm **Getpoint**, hàm **Getcorner** yêu cầu nhập vào một điểm và trả về tọa độ của điểm này. Tuy nhiên điểm này là điểm gốc đối diện hình chữ nhật.

(**Getcorner** PT [PROMPT])

Sự khác nhau giữa các hàm này là hàm **Getpoint** làm xuất hiện *sợi dây thun* trên màn hình, còn hàm **Getcorner** tạo ra một *cửa sổ động* trên màn hình.

Hàm LAST

Hàm **Last** trả về phần tử cuối cùng của danh sách.

(**Last** LIST)

Ví dụ:

Command: **(setq LL (list "pt1" "pt2" "pt3" "pt4" "pt5" "pt6"))**↵
Command: **(setq MM (last LL))** ↵
"pt6"

Hàm LENGTH

Hàm **Length** trả về số lượng các phần tử có trong danh sách.

(**Length** LIST)

Ví dụ:

Command: **(setq LL (list "pt1" "pt2" "pt3" "pt4" "pt5" "pt6"))**↵
Command: **(setq PP (length LL))**↵
6

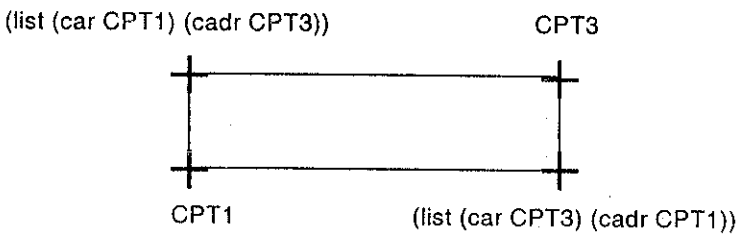
Tóm tắt:

1. Hàm **Car** trả về phần tử đầu tiên của một danh sách. Giá trị trả về thường có kiểu dữ liệu là số thực, số nguyên, chuỗi hoặc một biến.
2. Hàm **Cdr** thường dùng để xóa phần tử đầu tiên của một danh sách. Giá trị trả về là một danh sách.
3. Các hàm **Car** và **Cdr** có thể được kết nối để tạo ra các hàm có dạng **CxxxxR**, dùng để lấy ra các phần tử của một danh sách.
4. Hàm **Getcorner** thường được dùng nhiều hơn hàm **Getpoint** khi nhập các đỉnh của hình chữ nhật.
5. Hàm **Last** trả về phần tử cuối cùng của một danh sách. Hàm **Length** trả về số lượng các phần tử của một danh sách.

3.3 Các ví dụ mẫu

Ví dụ 1:

Chương trình vẽ hình chữ nhật:



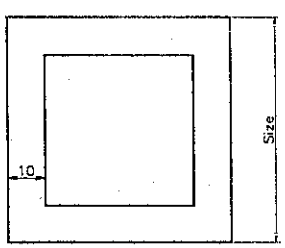

```
;Tên file: RECTANGL.LSP
;Chương trình này yêu cầu người sử dụng nhập hai đỉnh hình chữ nhật
; và sau đó dùng các hàm CAR và CADR để suy ra hai đỉnh còn lại.
(defun RECTANGL (/ CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: ")      ;Nhập điểm CPT1
        CPT3 (getcorner CPT1 "\nSecond corner point: ")      ;Nhập điểm CPT3
        CPT2 (list (car CPT3) (cadr CPT1))                   ;Tao ra đỉnh CPT2
        CPT4 (list (car CPT1) (cadr CPT3))                   ;Tao ra đỉnh CPT4
```

```
  (command ".line" CPT1 CPT2 CPT3 CPT4 "Close")      ;Vẽ hình chữ nhật
)
```

Ví dụ 2:

Viết một chương trình có tên là 2L-BOX. Dùng các hàm **Getpoint**, **Getcorner**, **List** để vẽ một hình vuông nét đôi. Kích thước hình chữ nhật do người sử dụng nhập vào. Khoảng cách giữa các nét đôi (offset distance) là 10.

```
;Tên file: 2L-BOX.LSP
;Vẽ một hình vuông nét đôi
(defun C:2L-BOX (/ PT1 PT2 PT3 PT4 PT5 PT6 PT7 PT8 L)
  (setq PT1 (getpoint "\nSelect first point: ")
        L (getreal "\nEnter the size: ")
        ,
        PT2 (list (+ (car PT1) L) (cadr PT1))
        PT3 (list (car PT2) (+ (cadr PT2) L))
        PT4 (list (car PT1) (+ (cadr PT1) L))
        ;
        PT5 (list (+ (car PT1) 10) (+ (cadr PT1) 10))
        PT6 (list (- (car PT2) 10) (+ (cadr PT2) 10))
        PT7 (list (- (car PT3) 10) (- (cadr PT3) 10))
        PT8 (list (+ (car PT4) 10) (- (cadr PT4) 10))
  )
  (command ".line" PT1 PT2 PT3 PT4 "c"
           ".line" PT5 PT6 PT7 PT8 "c"
  )
)
```



3.4 Bài tập

1. Hãy cho biết kết quả trả về của các biểu thức sau:

A = 15.0 B = "OK" C = '(2.0 1.5) D = "ABC" E = 23 X = E

a. (list "DEF" D)
h. (list (quote a))

b. (list 4.5 8.2 A)
c. (list "ABC" D "A" a)
d. (list 5 6 (list 9 10)
e. (setq p '(a b c))
f. (quote (a b c d e x))
g. (list e x)
i. (setq L8 '(e x a))
j. (setq L9 (list e x a))
k. (list c)
l. (setq L10 '(c A (list d b)))
m. (list (+ a E) x)
n. (setq L11 '((+ a E) x))

2. Hãy cho biết giá trị trả về của các biểu thức sau:

a. (car '("ONE" "TWO" "THREE"))
b. (car (list 9.77 4.61))
c. (setq b (car '(8 "A-B-C")))
d. (setq p (car '("GREEN" "BLUE")))
e. (cdr (list "BOLTS" 3.0 4.25))
f. (setq alpha (cdr '(ABC DEF)))
g. (cdr (list "ibeam" "bar" "wfbeam"))
h. (setq e (cdr '(m n o p))
i. (car (cdr '(3.44 9.28 5.31)))
j. (setq w (cdr (car (list '(a b c) 'd))))
k. (cdr (cdr '(lowpt midpt hipt)))
l. (setq q (car (cdr (list '(uv w) 'xy 'z))))

3. Viết chương trình (tương tự như chương trình XCOORD.LSP) thực hiện các công việc sau:

a. Dùng hàm **Getpoint** yêu cầu nhập vào 3 điểm trong không gian ba chiều. Gán tọa độ 3 điểm này cho 3 biến PT1, PT2, PT3.
b. Dùng các hàm **Car**, **Cadr** và **Caddr** để tạo ra điểm thứ tư như sau: điểm này có cùng tọa độ X với điểm thứ nhất, cùng tọa độ Y với điểm thứ hai và cùng tọa độ Z với điểm thứ ba.
c. Dùng lệnh **3dpoly** vẽ 4 hình tam giác có các đỉnh là 4 điểm nói trên.

4. Đinh giá trị các biểu thức sau đây:

a. (cadr '("FOUR" "FIVE" "SIX"))
b. (cadr (list 5.39 9.48))
c. (setq h (cadr '(8 12 46 2)))
d. (cadr (list 3 4 (cadr '(5 6))))
e. (caddr '("GREEN" "BLUE" "RED"))
f. (setq beta (caddr '(AB CD EF)))
g. (caddr '(a z))
h. (setq w (car (cddr '(m n o p))))
i. (cadar '((7.43 2.28 4.99) 8.92))

5. Các biến L1, L2 được gán các danh sách như sau:

```
(setq L1 '(a b c (d e f)))
(setq L2 '((a b) (c d) (e f)))
```

a. Hãy lấy ra phần tử c trong danh sách L1
b. Hãy lấy ra phần tử e trong danh sách L2
c. Hãy lấy ra danh sách '(d e f) trong danh sách L1

d. Hãy lấy ra phần tử b trong danh sách L2

6. Sử dụng các hàm **Getcorner**, **Car**, **Cdr** hãy viết chương trình có tên FILLBOX.LSP yêu cầu nhập vào hai điểm, sau đó dùng lệnh SOLID của **AutoCAD** để tạo một SOLID hình chữ nhật.

7. Hãy cho biết kết quả của các biểu thức sau:
a. (last '("WOOD" "STEEL" "IRON"))
b. (last (list 1.23 4.56))
c. (setq y (last (cdr '(1 3 5 7))))
d. (setq s (last (car '((a b c) d))))
e. (length '("RED" "GREEN" "BLUE"))
f. (length (list (last (list "BOLTS" 4.25))))
g. (setq gamma (length '(AB CD EF)))
h. (setq u (length (list (car (cdr '(9 5 1))))))
i. (length (list "plate" "bar" "rod"))

8. Viết một chương trình có tên là 3D-RECT.LSP. Dùng các hàm **Getpoint** và **Getcorner** để nhập vào các đỉnh của một hình chữ nhật. Dùng các hàm xử lý danh sách để tạo ra 4 đỉnh còn lại, tạo thành một hình khối chữ nhật có chiều dày 20 đơn vị. Sau đó vẽ các 3DFACE tạo thành các mặt của khối chữ nhật.

3.5 Lời giải

1.

a. ("DEF" "ABC")
b. (4.5 8.2 15.0)
c. ("ABC" "ABC" "A" 15.0)
d. (5 6 (9 10))
e. (a b c)
f. (a b c d e x)
g. (23 23)
h. (a)
i. (e x a)
j. (23 23 15.0)
k. ((2.0 1.5))
l. (c A (list d b))
m. (38.0 23)
n. ((+ a E) x)

2.

a. "ONE"
b 9.77.
c. 8
d. "GREEN"
e. "BOLTS"
f. (DEF)
g. ("bar" "wfbeam")
h. (n o p)
i. 9.28
j. (b c)
k. (hipt)
l. xy

3.

```
;Tên file: XYZ-TRI.LSP
;
(defun C:XYZ-TRI (/ PT1 PT2 PT3 PT4)
   (setq PT1 (getpoint "\nSelect first 3D point: "))
   (setq PT2 (getpoint "\nSelect second 3D point: "))
   (setq PT3 (getpoint "\nSelect third 3D point: "))
   ;
   (setq PT4 (list (CAR PT1) (CADR PT2) (CADDR PT3)))
   ;
   (command ".3DPOLY" PT1 PT2 PT3 "c")
   (command ".3DPOLY" PT1 PT2 PT4 "c")
   (command ".3DPOLY" PT2 PT3 PT4 "c")
   (command ".3DPOLY" PT3 PT1 PT4 "c")
)
```

4.

a. "FIVE"
b. 9.48
c. 12
d. 4
e. "RED"
f. EF
g. nil
h. o
i. 2.28

5.

a. (caddr L1)
b. (caaddr L2)
c. (cadddr L1)
d. (cadar L2)

6.

```
;Tên file FILLBOX.LSP
;
(defun C:FILLBOX (/ CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: ")
  CPT3 (getcorner CPT1 "\nSecond corner point: ")
    CPT2 (list (car CPT3) (cadr CPT1))
    CPT4 (list (car CPT1) (cadr CPT3))
)
(command ".solid" CPT1 CPT2 CPT4 CPT3 "")
)
```

7.

a. "IRON"
b. 4.56
c. 7
d. c
e. 3
t. 1
g. 3
h. 1
i. 3

8.

```
;Tên file: 3d-RECT.LSP
;Vẽ hộp chữ nhật với chiều cao (height) 20
(defun C:3D-RECT (/ PT1 PT2 PT3 PT4 PT5 PT6 PT7 PT8)
  (setq PT1 (getpoint "\nSelect first corner point: ")    ;Đỉnh 1 mặt đáy
        PT3 (getcorner PT1 "\nSecond corner point: ") ;Đỉnh 3 mặt đáy
        PT2 (list (car PT3) (cadr PT1))                 ;Đỉnh 2 mặt đáy
        PT4 (list (car PT1) (cadr PT3))                 ;Đỉnh 4 mặt đáy
        ;
        PT5 (list (car PT1) (cadr PT1) 20)              ;Đỉnh 5 mặt trên
        PT6 (list (car PT2) (cadr PT2) 20)              ;Đỉnh 6 mặt trên
        PT7 (list (car PT3) (cadr PT3) 20)              ;Đỉnh 7 mặt trên
        PT8 (list (car PT4) (cadr PT4) 20)              ;Đỉnh 8 mặt trên
    )
  ,
  (command vpoint "" 1,-1,1  "")                        ;Đỉnh điểm nhìn
  (command ".3DFACE" PT1 PT2 PT3 PT4 ""                 ;Vẽ các mặt
           ".3DFACE" PT5 PT6 PT7 PT8 ""
           ".3DFACE" PT1 PT4 PT8 PT5 ""
           ".3DFACE" PT2 PT3 PT7 PT6 ""
           ".3DFACE" PT1 PT2 PT6 PT5 ""
           ".3DFACE" PT3 PT4 PT8 PT7 ""
  )
  (command "zoom" "all")                                ;Phóng to hình
)
```


NHẬP DỮ LIỆU

Nội dung chương

1. Nhập các kiểu dữ liệu khác nhau. Sử dụng các hàm: **Getint, Getreal, Getstring.**
2. Kiểm soát giá trị các dữ liệu nhập bằng các hàm: **Initget, Getkword.**
3. Thay đổi giá trị các biến hệ thống của **AutoCAD**. Hàm **Getvar, Setvar.**
4. Nhập màu bằng hàm **Acad_Colordlg**

4.1 Nhập dữ liệu

Chương trình **AutoLISP** linh hoạt hơn các *script file* và *menu macro* nhờ có khả năng **nhập dữ liệu từ người sử dụng**. Kết quả do chương trình tạo ra thay đổi tùy thuộc vào dữ liệu nhập.

4.1.1 Nhập dữ liệu số nguyên (integer)

Trong một số trường hợp, dữ liệu nhập vào phải là kiểu số nguyên (integer). Ví dụ: hệ số tỉ lệ, số cột số hàng trong lệnh **Array**, số đoạn chia trong lệnh **Divide**...

Hàm **Getint** được dùng để yêu cầu nhập vào một số nguyên.

(**Getint** [PROMPT])

✌ **Ví dụ:**

Command: **(getint)**↵
12 ↵
12

Yêu cầu nhập một số nguyên. Số nguyên nhập vào là 12. Giá trị trả về là 12.

Command: **(getint)** ↵
12.0 ↵
Requires an integer value
Try again: **12** ↵
12

Nếu dữ liệu nhập vào không phải là số nguyên, chương trình sẽ hiện thông báo lỗi và yêu cầu nhập lại. Dòng nhắc mặc định là *"Try again: "*

Command: **(getint "\nEnter an integer: ")**↵
Enter an integer: **43** ↵
43

Dòng nhắc chứa trong tham số PROMPT.

Command: **(setq NUM (getint ""\nEnter an integer: "))** ↵
Enter an integer: **120** ↵
120

Dữ liệu nhập vào được gán cho biến NUM.

Command: **Array** ↵
Seiect objects: (chọn đối tượng)
Select objects:↵
*Rectangular or Polar array (<R>/<P>):***R** ↵
Number of rows (---)<1>: **!NUM** ↵
Number of columns (|||)<1>: **!NUM** ↵

Giá trị biến NUM là 120 được gán cho số hàng và số cột.

Unit cell or distance between rows (---):
Distance between columns (|||):

Command: **(getint "\nEnter an Integer: ")**↵
Enter an integer: **32768**↵
Requires an integer between -32768 and 32767.
Enter an integer: **32767**↵
32767

Số nguyên có giá trị trong khoảng từ –32768 đến 32767

✌ **Ví dụ:** Chương trình gán đơn vị đo chiều dài và số chữ số thập phân.

```
;Tên file: QU.LSP (Quick Unit)
(defun C:QU (/ LUS LUP)
  (setq LUS (getint "\nEnter the Linear Units <1-5>: ")
        LUP (getint "\nEnter the Precision: ")
  )
  (command ".LUNITS" LUS "LUPREC" LUP)
) ;Kết thúc chương trình QU
```

4.1.2 Nhập dữ liệu số thực (real)

Hàm **Getreal** được dùng để yêu cầu nhập vào một số thực.

(**Getreal** [PROMPT])

✌ **Ví dụ:**

Command:**(getreal)**↵
3.66 ↵
3.66

Yêu cầu nhập một số thực. Số thực nhập vào là 3.66. Giá trị trả về là 3.66

Command:**(getreal "\nEnter a number: ")**↵
Enter a number: **43**↵
43.0

Có thể nhập vào số nguyên, nhưng giá trị trả về được tự động chuyển đổi thành số thực.

Command:**(getreal "\nEnter a number: ")**↵
Enter a number: **HELLO** ↵
Requires numeric value.
Enter a number: **622**↵
622.0

Nếu dữ liệu nhập vào không phải là một số, chương trình sẽ hiện thông báo lỗi và yêu cầu nhập lại.

✌ **Ví dụ:** Chương trình tính thể tích hình nón.

```
;Tên file: CONEVOL.LSP
(defun C:CONEVOL (/ R H)
  (setq R (getreal "\nEnter radius of cone: ")
        H (getreal "\nEnter height of cone: ")
  )
  (prompt "\nThe volume of this cone is: ")
  (* (/ 1 3.0) PI (* R R) H)      ;Tính thể tích hình nón
)  ;Kết thúc chương trình CONEVOL
```

4.1.3 Nhập dữ liệu kiểu chuỗi (string)

Hàm **Getstring** dùng để yêu cầu nhập vào một chuỗi. Chuỗi nhập vào không được vượt quá 132 ký tự.

(**Getstring** [PROMPT])

✌ **Ví dụ:**

Command: **(getstring)** ↵ **Hello** ↵ "Hello"	Yêu cầu nhập vào một chuỗi. Chuỗi nhập vào là "Hello". Giá trị trả về là chuỗi "Hello"
Command: **(getstring)**↵ **154.77** ↵ *"154.77"*	Nếu giá trị nhập vào là một số, hàm **Getstring** tự động chuyển đổi sang kiểu chuỗi.
Command: **(getstring "\nEnter text: ")** *Enter text:* **Material** ↵ "Material"	Dòng nhắc chứa trong tham số PROMPT.
Command:**(getstring T "Enter text:")**↵ *Enter text:* **Notes (unless otherwise specified)** ↵ *"Notes (unless otherwise specified)"*	Để chuỗi nhập vào có thể chứa các khoảng trắng, ta phải sử dụng tham số không rỗng **T** (non-nil argument).

4.1.4 Tham số không rỗng (non-nil argument)

Nhiều hàm **AutoLISP** sử dụng các tham số không rỗng. *Tham số không rỗng* là tham số chỉ chấp nhận giá trị khác rỗng. Có hai cách cung cấp tham số không rỗng: dùng biến **T** hoặc dùng một giá trị cụ thể.

Biến **T** được **AutoLISP** định nghĩa trước và có giá trị là T (có ý nghĩa là TRUE). Ta không nên thay đổi giá trị của biến **T**. Để xem giá trị của **T** ta dùng cách sau:

Command: **!T** ↵
T

Ta có thể cung cấp một tham số không rỗng bằng cách dùng một giá trị cụ thể ví dụ như một số nguyên:

Command: **(getstring 2 "\nEnter text: ")** ↵
Enter text: **Notes (unless otherwise specified)** ↵
"Notes (unless otherwise specified)"

4.2 Kiểm soát dữ liệu nhập

Nhiều chương trình đưa ra các lựa chọn cho người sử dụng chọn. Ví dụ, lệnh **Circle** cung cấp nhiều lựa chọn như *3-Point, Center, Radius...*

Để làm điều này, trong **AutoLISP** có hàm **Initget** cung cấp danh sách các giá trị nhập hợp lệ tương ứng với các lựa chọn.

Hàm INITGET

Hàm **Initget** cung cấp danh sách các giá trị nhập hợp lệ bằng cách gán các *bit kiểm tra (bit code)* và danh sách các từ khóa. Các loại hàm nhập dữ liệu như **Getpoint**, **Getcorner**, **Getint**, **Getreal** ... (ngoại trừ hàm **Getstring**) đều bị kiểm soát bởi hàm **Initget**. Tuy nhiên, hàm **Initget** chỉ có tác dụng với một lần dùng hàm **Getxxxx**. Nó bị mất tác dụng khi dùng hàm **GETxxxx** các lần kế tiếp. Để kiểm soát các hàm nhập dữ liệu tiếp theo ta phải sử dụng lại hàm **Initget**.

(**Initget** [BITS] [STRING])

Tham số BITS là một số nguyên. Giá trị tham số này bằng tổng các *bit code* tương ứng với các chế độ kiểm soát mà ta mong muốn.

Tham số STRING chứa danh sách các từ khóa.

Bit code	Chế độ kiểm soát
1	Giá trị phải được nhập; không chấp nhận giá trị null (chỉ nhấn phím ENTER mà không nhập giá trị).
2	Giá trị nhập phải khác không.
4	Giá trị nhập không được là số âm.

8	Chấp nhận điểm nhập vào nằm ngoài giới hạn bản vẽ ngay cả khi biến hệ thống LIMCHECK = ON (lựa chọn ON của lệnh LIMITS).
16	Không sử dụng trong release 14.
32	Dùng nét đứt để thể hiện "sợi dây thun". Nếu biến hệ thống POPUPS = 0 thì bit này không có tác dụng.
64	Dùng cho hàm **Getdist**. Hàm **Getdist** sẽ loại bỏ toạ độ Z của hai điểm cần tính khoảng cách (tương tự như chiếu hai điểm này lên mặt phẳng xy, sau đó tính khoảng cách giữa hai điểm chiếu này).
128	Cho phép nhập chuỗi ký tự không có trong danh sách các từ khóa. Các bit code khác sẽ được ưu tiên trước, nhưng nếu bit 1 và 128 được sử dụng cùng với nhau (129) thì giá trị null nhập vào (nhấn ENTER) sẽ được chuyển đổi thành chuỗi rỗng "".

Ví dụ:

Command: **(initget (+ 1 2 4)** ↵
Nil

Hoặc **(initget 7)**. Dữ liệu nhập vào phải khác rỗng (bit code 1), khác không (bit code 2), và không âm (bit code 4).

Command: **(setq NUM (getint "\nEnter a positive integer: ")**↵
Enter a positive integer: ↵
Value must be positive and nonezero.
Enter a positive integer: **0** ↵
Value must be positive and nonezero.
Enter a positive integer: **-72** ↵
Value must be positive and nonezero.
Enter a positive integer: **14** ↵
14

Không cho phép nhập giá trị rỗng, bằng không và giá trị âm.

Command: **(setq NOM (getint "\nEnter a positive integer: ")**↵
Enter a positive integer: **-72** ↵
-72

Hàm **Initget** chỉ có tác dụng với một lần dùng hàm **Getxxxx**. Nó bị mất tác dụng khi dùng hàm **Getxxxx** các lần kế tiếp. Do đó, ta có thể nhập vào giá trị âm (-72).

Command: **(initget 1 "Cancel")** ↵
Nil

Không cho phép nhập giá trị rỗng. Từ khóa nhập hợp lệ là *"Cancel"*

Command: **(getint "\nCancel/<Floor #>: ")** ↵

Cancel/<Floor #>: **F** ↵
Requires an integer value or option keyword.
Cancel/<Floor #>: **C** ↵
"Cancel"

Chỉ có thể nhập một số nguyên hoặc nhập từ khóa hợp lệ là *"Cancel"*

Command: **(getint "\nCancel/<Floor #>: ")** ↵
Cancel/<Floor #>: **C** ↵
Requires a numeric value.
Cancel/<Floor #>: **4** ↵
4

Initget chỉ có tác dụng với hàm **Getxxxx** đầu tiên.

Hàm GETKWORD

Hàm **Getkword** yêu cầu nhập dữ liệu ở dạng từ khóa. Tương tự các hàm **GETxxxx** khác, các từ khóa hợp lệ được cung cấp bởi hàm **initget**.

(**Getkword** [PROMPT]

Hàm **Getkword** chỉ chấp nhận 2 bit code trong hàm **Initget** là 1 và 128.

Ví dụ:

Command: **(initget 1 "Y N")** ↵
nil

Không chấp nhận nhập giá trị rỗng. Các từ khóa hợp lệ là "Y" và "N".

Command: **(setq RES (getkword "\nCreate DIMension layer?<Y/N>: "))** ↵
Create DIMension layer?<Y/N>: **LATER** ↵
Invalid option keyword.
Create DIMension layer?<Y/N>: **N** ↵
"N"

"LATER" Không có trong danh sách các từ khóa.
"N" là từ khóa hợp lệ.

Ví dụ:

Command: **(initget "Y N")**↵
nil

Chấp nhận nhập giá trị rỗng. Các từ khóa hợp lệ là "Y" và "N"

*Command:***(setq INP1 (getkword "\nDefine new value BLOCK?<Y/N>:"))**↵
Define new value BLOCK? <Y/N>. ↵
Nil

Giá trị rỗng được nhập vào.

Command: **(initget)**↵
nil

Không cung cấp danh sách từ khóa.

Command: **(setq INP2 (getkword "\nDefine new LType? <Y/N>: "))**↵
Define new LType? <Y/N>: nil

Nếu không được cung cấp danh sách các từ khoa hợp lệ, hàm **Getkword** sẽ tự động kết thúc, không dừng lại đợi người sử dụng nhập dữ liệu.

Ta có thể nhập một vài ký tự đầu tiên của từ khóa (nhưng phải nhập đủ các ký tự viết hoa), hàm **Getkword** sẽ trả về từ khóa đầy đủ.

✌ **Ví dụ:**

```
(defun C:BYE (/ OPT)
  (initget 1 "Save Wblock Quit")
  (setq OPT (getkword "\nSave, Wblock or Quit drawing? <S/W/Q>:"))
  (command OPT)
)
```

Nếu ta nhập vào ký tự S trong ví dụ trên, hàm **Getkword** sẽ trả về chuỗi "Save" và gán nó cho biến OPT. Sau đó, lệnh **Save** sẽ được thực hiện tại biểu thức (command OPT).

4.3 Các biến hệ thống

Các *biến hệ thống* **AutoCAD** dùng để điều khiển môi trường làm việc của **AutoCAD**. Biến hệ thống của **AutoCAD** không giống các biến của **AutoLISP** (cục bộ hoặc toàn cục). Ta không thể dùng hàm **Setq** để gán giá trị cho các biến hệ thống. Các biến hệ thống của **AutoCAD** giới thiệu đầy đủ trong phụ lục II.

Hàm **Getvar** cho phép xem nội dung các biến hệ thống.

(**Getvar** VARNAME)

✌ **Ví dụ:**

Command:**(setq PT1 (getvar "LASTPOINT"))**↵ ;Toạ độ điểm cuối cùng
(6.625 3.9213 0.0)
Command: **(setq CL (getvar "CLAYER"))**↵ ;Layer hiện hành
"0"

Command: **(setq LM (getvar "LIMMAX"))** ↵ ;Giá trị max của giới ;hạn bản vẽ
(420 297)
Command: **(setq FR (getvar "FILLETRAD"))**↵ ;Bán kính bo góc.
20

Hàm **Setvar** cho phép gán giá trị cho các biến hệ thống.

(**Setvar** VARNAME VALUE)

✌ **Ví dụ:**

Command: **(setvar "LUPREC" 2)**↵
2
Command: **(setvar "BLIPMODE" 0)** ↵
0
Command: **(setvar "DIMLFAC" 1)** ↵
1

Trong **AutoCAD** có một biến hệ thống tên là **CMDECHO**. Nếu CMDECHO = 1 (mặc định), khi thực hiện chương trình **AutoLISP**, kết quả tính toán của các hàm trung gian sẽ xuất hiện trên màn hình. Khi CMDECHO = 0, các thông tin này sẽ không xuất hiện. Do đó, tại đầu các chương trình **AutoLISP** ta nên có biểu thức sau đây:

(**Setvar** "CMDECHO" 0)

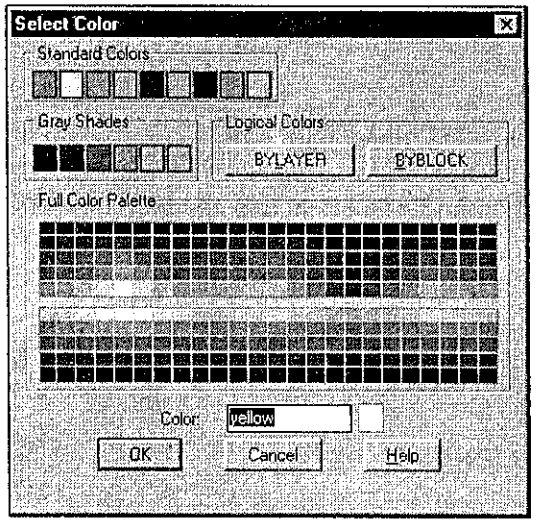
4.4 Hộp thoại Select Color

Hàm **Acad_Colordlg** làm xuất hiện hộp thoại **Select Color** của **AutoCAD** để người sử dụng chọn màu. Giá trị trả về là một số nguyên đại diện cho màu người sử dụng chọn.

(**Acad_Colordlg** COLORNUM [FLAG])

Tham số COLORNUM dùng để gán màu mặc định khi hiện hộp thoại **Select Color**.

Tham số FLAG = nil dùng để vô hiệu hóa hai nút BYLAYER và BYBLOCK trên hộp thoại.



Ví dụ:

Command: **(acad_colordlg 2)** ↵ — Màu mặc định là màu vàng (2). Người sử dụng chọn màu đỏ (1)

1

Command: **(acad_colordlg 2 nil)** ↵ — Vô hiệu hóa hai nút BYLAYER và BYBLOCK

1

Ví dụ:

Gán màu cho chữ số kích thước (dimension text). Người sử dụng chọn màu, sau đó màu này được gán cho biến NC. Hàm **Command** sẽ gán màu chứa trong biến NC cho biến hệ thống DIMCLRT.

Command: **(setq NC (acad_colordlg 7))(command "DIMCLRT" NC)**↵
DIMCLRT
New value for DIMCLRT<0>: **1**↵
Command:*nil*

Tóm tắt

1. Các hàm có dạng **GETxxxx** dùng để nhập các kiểu dữ liệu khác nhau.
2. Hàm **Getint** dùng để nhập số nguyên có giá trị trong khoảng từ –32768 đến 32767.
3. Hàm **Getreal** dùng để nhập số thực và số nguyên. Giá trị trả về là số thực.

4. Hàm **Getstring** dùng để nhập một chuỗi ký tự (tối đa là 132 ký tự). Nếu nhập vào số nguyên hoặc số thực, chúng sẽ được chuyển thành chuỗi. Để chuỗi có thể chứa khoảng trắng, ta phải dùng tham số không rỗng.
5. Hàm **Initget** cung cấp các giá trị nhập hợp lệ cho các hàm **GETxxxx**.
6. Hàm **Initget** phải xuất hiện trước hàm **Getkword**. Hàm **Getkword** yêu cầu nhập dữ liệu dưới dạng từ khóa.
7. Các hàm **Getvar** và **Setvar** dùng để xem và gán giá trị cho các biến hệ thống **AutoCAD**.
8. Hàm **Acad_Colordlg** yêu cầu người sử dụng nhập vào một màu bằng cách chọn trên hộp thoại.

4.5 Bài tập

1. Viết chương trình có tên CYL.LSP. Yêu cầu nhập vào đường kính và chiều cao hình trụ tròn, sau đó tính diện tích xung quanh của nó :

   Area = Pi x Diameter x Height

2. Viết chương trình có tên INTOMM.LSP chuyển đổi dữ liệu nhập vào từ đơn vị inch sang mm.
3. Viết chương trình có tên MMTOIN.LSP chuyển đổi dữ liệu nhập vào từ đơn vị mm sang inch.
4. Tạo một hàm có tên LL dùng được tại dòng nhắc lệnh của **AutoCAD**. Hàm này sử dụng hàm **Getstring** yêu cầu người sử dụng nhập vào tên một file chương trình **AutoLISP**, sau đó dùng hàm **Load** để tải chương trình này.
5. Hãy sửa lại chương trình CYL.LSP trong bài tập 1, sao cho không thể nhập các giá trị âm hoặc bằng 0 cho đường kính và chiều cao hình trụ tròn.
6. Hãy viết một hàm có tên FR dùng được tại dòng nhắc lệnh **AutoCAD**. Hàm này yêu cầu nhập bán kính bo góc, dùng hàm **Setvar** để gán giá trị mới cho biến hệ thống FILLETRAD, sau đó thực hiện lệnh **Fillet**.
7. Hãy viết một hàm có tên BOR dùng được tại dòng nhắc lệnh **AutoCAD**. Hàm này yêu cầu nhập các giá trị giới hạn bản vẽ, dùng hàm **Setvar** để gán giá trị mới cho các biến hệ thống tương ứng, sau đó vẽ một hình chữ nhật quanh giới hạn này bằng đường polyline có chiều rộng bằng 1.
8. Viết chương trình **AutoLISP** CALC-IT.LSP. Tạo các hàm tính toán các công thức sau. Các dữ liệu nhập phải được kiểm tra hợp lệ.

a. Tính chu vi hình chữ nhật : P = 2(a + b)

   P – Chu vi

a – Chiều dài
b – Chiều rộng

b. Tính diện tích hình tam giác: $A = \frac{1}{2}bh$

A – Diện tích
b – Cạnh đáy
h – Chiều cao

c. Tính diện tích xung quanh của hình trụ tròn: $A = 2\pi rh$

A – Diện tích
r – Bán kính
h – Chiều cao

d. Tính diện tích hình tròn: $A = \frac{\pi d^2}{4}$

A – Diện tích
d – Đường kính

e. Tính thể tích hình chop: $V = \frac{1}{3}bh$

V – Thể tích
b – Diện tích đáy
h – Chiều cao

9. Bổ sung các hàm **Iniget** và **CMDECHO = 0** cho các chương trình đã viết:

ABC.LSP	1LINE.LSP
SAMPLE.LSP	TRIANGLE.LSP
GPT1.LSP	2X.LSP
GPT2.LSP	RECTANGLE.LSP
GPT3.LSP	3D-RECT.LSP

10. Lập chương trình để chèn lại block đã được chèn trên bản vẽ (BKN.LSP)

4.6 Lời giải

1.
```
;Tên file: CYL.LSP
(defun C:CYL (/ D H)
  (setq D (getreal "\nEnter diameter of a cylinder: ")
        H (getreal "\nEnter Height of a cylinder: ")
  )
  (prompt "Surface area of this cylinder is: ")
```



```
  (* Pi D H)
)
```

2:
```
;Tên file INTOMM.LSP
;Chuyển đổi từ inch sang mm
(defun INTOMM (/ NUM)
  (setq Num (getreal "\nEnter a number in inches: "))
  (prompt "\nThe value in milimeters is: ")
  (* 25.4 NUM)
)
```

3.
```
;Tên file MMTOIN.LSP
;Chuyển đổi từ mm sang inch
(defun MMTOIN (/ NUM)
  (setq Num (getreal "\nEnter a number in milimeters: "))
  (prompt "\nThe value in inch is: ")
  (/ NUM 25.4)
)
```

4.
```
;Tên file LL.LSP
;Yêu cầu nhập tên file chương trình AutoLISP
;Sau đó tải chương trình này
;
(defun C:LL (/ FNAME)
  (setq FNAME (getstring T "\nEnter file name to load: "))
  (load FNAME)
)
```

5.
```
;Tên file CYL.LSP
(defun C:cyl (/ D H)
  (initget (+ 1 2 4))
  (setq D (getreal "\nEnter diameter of a cylinder: "))
  (initget (+ 1 2 4))
  (setq H (getreal "\nEnter Height of a cylinder: "))
  (prompt "Surface area of this cylinder is: ")
  (* Pi D H)
)
```

6.

```
;Tên file FR.LSP
;
(defun C:FR (/ R)
  (setvar "CMDECHO" 0)
  (initget (+ 1 2 4))
  (setq R (getreal "\nEnter fillet radius: "))
  (setvar "FILLETRAD" R)
  ;
  (command ".fillet")
)
```

7.

```
;Tên file BOR.LSP
;Nhập giới hạn bản vẽ và vẽ hình chữ nhật bao quanh
;
(defun C:BOR (/ PT1 PT2 PT3 PT4)
   (setvar "CMDECHO" 1)
   (setq PT1 (getpoint "\nEnter lower left corner: ")
         PT3 (getpoint "\nEnter upper right corner: ")
         PT2 (list (car PT3) (cadr PT1))
        PT4 (list (car PT1) (cadr PT3))
   )
   (setvar "LIMMIN" (list (car PT1) (cadr PT1)))
   (setvar "LIMMAX" (list (car PT3) (cadr PT3)))
   (command ".LIMITS" "" ""
            ".Zoom" "All")
   (command ".Pline" PT1 "WIDTH" "1" "" PT2 PT3 PT4 "Close")
)
```

8.

```
;Tên file CALC-IT.LSP
;
; Tính chu vi hình chữ nhật
(defun C:PR (/ a b)
   (initget (+ 1 2 4))
   (setq a (getreal "\nLength: "))
   (initget (+ 1 2 4))
   (setq b (getreal "\nWidth: "))
   (prompt "\nPerimeter of rectangle: ")
   (* 2 (+ a b))
)
;
;Diện tích tam giác
(defun C:AT (/ b h)
   (initget (+ 1 2 4))
   (setq b (getreal "\nBase: "))
   (initget (+ 1 2 4))
   (setq h (getreal "\nHeight: "))
   (prompt "\nArea of triangle: ")
   (/ (* b h) 2)
)
;
;Diện tích mặt cong hình trụ
(defun C:SA (/ r h)
   (initget (+ 1 2 4))
   (setq r (getreal "\nRadius: "))
   (initget (+ 1 2 4))
   (setq h (getreal "\nHeight: "))
   (prompt "\nSurface area of Cylinder: ")
   (* 2 pi r h)
)
;
;Diện tích đường tròn
(defun C:AC (/ d)
   (initget (+ 1 2 4))
   (setq d (getreal "\nDiameter: "))
   (prompt "\nArea of Circle: ")
   (/ (* pi d d) 4)
)
;Thể tích lăng trụ
(defun C:VP (/ b h)
   (initget (+ 1 2 4))
   (setq b (getreal "\nArea of base: "))
   (initget (+ 1 2 4))
   (setq h (getreal "\nHeight: "))
   (prompt "\nVolume of pyramid: ")
   (/ (* b h) 3)
)
```

9. Sử dụng chương trình để chèn lại một block. Khi đó bạn chỉ cần chọn lại block đã chèn (không cần nhập tên) và định lại *Insertion point*, *X* và *Y scale*, *rotation angle*.

```
;Tên file: BKN.LSP
;Sử dụng để chèn lại một block
(defun C:BKN ( / ENT  BNAME)
     (setq ENT (entget  (car   (entsel "\nSelect block to insert again: " ) ) ) )
    (setq BNAME (cdr (assoc 2 ENT )))
    (command ".insert" BNAME)
    (princ)
); Kết thúc chương trình
```

Sau khi tải chương trình bằng lệnh **Appload**, ta nhập **BKN** tạo dòng lệnh:

Command:**BKN** ↵
Select block to insert again: (Chọn block đã chèn trên bản vẽ)
Insertion point: (Chọn lại điểm chuẩn chèn)
Insertion point: X scale factor <1> / Corner / XYZ: (Tỉ lệ phương X)
Y scale factor (default=X): (Tỉ lệ phương Y)
Rotation angle <0>: (Góc quay)

Chương 5

KHOẢNG CÁCH và GÓC ĐO

Nội dung chương

1. Hàm chuyển đổi đơn vị đo: **Cvunit**.
2. Các hàm xác định khoảng cách và góc đo: **Angle**, **Distance**, **Polar**, **Getangle**, **Getorient**, **Getdist**.
3. Xác định tọa độ điểm bằng chức năng truy bắt đối tượng, tọa độ cực, giao điểm giữa hai đường thẳng. Các hàm: **Osnap**, **Inters**.

5.1 Các hàm chuyển đổi đơn vị đo

Trong môi trường **AutoCAD**, ta có thể định đơn vị đo chiều dài và đơn vị đo góc phù hợp. Thông thường ta sử dụng đơn vị đo góc là độ thập phân (decimal degree). Nhưng **AutoLISP** chỉ sử dụng đơn vị đo góc là radian. Do đó, khi cần thiết ta phải chuyển đổi giá trị góc đo từ độ sang radian hoặc từ radian sang độ.

Hàm CVUNIT

Hàm **Cvunit** (*ConVert UNITs*) dùng để chuyển đổi một giá trị hoặc tọa độ một điểm từ đơn vị đo này sang đơn vị đo khác.

(**Cvunit** VALUE FROM TO)

trong đó, VALUE – số nguyên, số thực hoặc tọa độ điểm 2D, 3D.
FROM – Đơn vị đo hiện tại (kiểu chuỗi).
TO – Đơn vị đo sẽ chuyển sang (kiểu chuỗi).

Ví dụ:

(cvunit 180 "DEGREE" "RADIAN") trả về 3.14159
(cvunit PI "RADIANS" "DEGREE") trả về 180.0
(cvunit '(1.0 3.0) "FT" "IN") trả về (12.0 36.0)
(cvunit '(1 2) "FEET" "INCHES") trả về (12.0 24.0)
(cvunit '(4 2) "FOOT" "INCH") trả về (48.0 24.0)
(cvunit 10 "HOURS" "CENTIMETERS") trả về nil vì không chuyển đổi được
(cvunit 4 "INCHES" "GALLONS") trả về nil vì không chuyển đổi được

Hàm **Cvunit** lấy thông tin về đơn vị đo từ file **ACAD.UNT**. Chúng ta có thể sửa đổi, bổ sung nội dung trong file này cho phù hợp. Nhưng vì vậy có thể có trường hợp file này bị đổi tên hoặc bị xóa, hoặc các đơn vị đo cần thiết không có trong file này. Vì vậy, ta nên tạo riêng các hàm chuyển đổi thường dùng và đặt chúng ở đầu chương trình. Nhờ đó, chương trình không phụ thuộc vào file ACAD.UNT.

Ví dụ:

```
(defun DTR (A)
      (* PI (/ A 180.0))        ;Chuyển đổi từ độ sang radian
)

(defun RTD (A)
      (/ (* A 180.0) PI)        ;Chuyển đổi từ radian sang độ
)
```



```
Command: (dtr 180) ↵
3.14159
Command: (rtd 1.570796) ↵
90.0
```

5.2 Xác định khoảng cách và góc đo

Hàm ANGLE

Hàm **Angle** trả về góc đo (tính bằng radian) tạo bởi đường thẳng qua hai điểm với trục X trong mặt phẳng XY.

(**Angle** PT1 PT2)

Nếu các điểm PT1, PT2 là các điểm 3D, chúng sẽ được chiếu lên mặt phẳng XY hiện hành, sau đó hai điểm chiếu này được dùng để tính góc.

Ví dụ:

```
Command: (angle '(0 0) '(0 4)) ↵          Trả về góc đo tính bằng radian
1.5708
Command: (rtd (angle '(0 0) '(0 4))) ↵    Dùng hàm Rtd để chuyển sang độ
90.0
Command: (cvunit (angle '(0 0) '(0 4)) "RADIANS" "DEGREES") ↵
90.0                                      Dùng hàm Cvunit chuyển sang độ
```

Hàm DISTANCE

Hàm **Distance** trả về khoảng cách giữa hai điểm (2D hoặc 3D).

(**Distance** PT1 PT2)

Ví dụ:

```
Command: (distance '(0 0) '(0 4)) ↵       Hai điểm 2D. Trả về khoảng cách
4.0                                       giữa hai điểm 2D.
Command: (distance '(0 0 0) '(0 1 1)) ↵   Hai điểm 3D. Trả về khoảng cách
1.41421                                   giữa hai điểm 3D.
Command: (distance '(0 0) '(0 1 1)) ↵     Một điểm 2D và một điểm 3D.
1.0                                       Tọa độ Z của điểm 3D sẽ bị bỏ
                                          qua. Trả về khoảng cách giữa hai
                                          điểm 2D.
```

Hàm POLAR

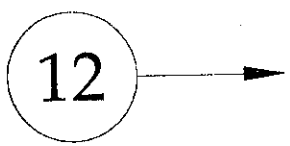
Hàm **Polar** dùng tọa độ cực để tạo ra điểm (2D) mới từ điểm ban đầu.

(**Polar** PT ANGLE DISTANCE)

Ví dụ:

Command: **(polar '(1 0) 0 3)** ↵	Điểm 2D. Góc bằng 0, khoảng cách bằng 3 so với điểm (1, 0)
(4.0 0.0)	
Command: **(polar '(1 0 0) 0 3)** ↵	Điểm 3D. Điểm trả về của hàm **Polar** luôn có cùng toa độ Z với điểm ban đầu PT (1, 0, 0).
(4.0 0.0 0.0)	
Command: **(polar '(3.5 8.0 11.5) 0 -3)**↵	Góc bằng 0, khoảng cách bằng -3.
(0.5 8.0 11.5)	
Command: **(polar '(3.5 8.0 11.5) Pi 3)**↵	Góc bằng Pi (180^0), khoảng cách bằng 3
(0.5 8.0 11.5)	
Command: **(setq PT1 '(1 0 9))** ↵	
(1 0 9)	
Command: **(polar PT1 (* 0.5 Pi) 7.25)**↵	Góc bằng 0.5*Pi, khoảng cách bằng 7.25
(1.0 7.25 9.0)	

Ví dụ: Phối hợp các hàm **Angle**, **Distance**, **Polar** để vẽ ký hiệu số vị trí.




```
;Tên file: BALLOON.LSP
;Chương trình yêu cầu nhập vào vị trí vẽ ký hiệu, số vị trí và vị trí mũi tên.
;Sau đó vẽ vòng tròn đường kính 8mm chứa số vị trí và vẽ mũi tên
;
;Các biến cục bộ:      CC- Tâm đường tròn
;                      IN- Số vị trí
;                      LS- Điểm vẽ mũi tên
(defun C:BLN (/ CC IN LS)
  (setq CC (getpoint "\nCenter point for balloon: ")
    IN (getstring "\nItem Callout <max. 2 characters: ")
       LS (getpoint "\nLeader start point: ")
  )
  (command ".CIRCLE" CC "d" 12      ;Vẽ đường tròn 8mm có tâm tại CC
   ".TEXT" "M" CC 5 0 IN            ;Dòng chữ 3mm tại tâm đường tròn
   ".DIM" "lea" LS                  ;Vẽ mũi tên từ điểm LS đến giao
                                    ;điểm giữa đường tròn và đường
                                    ;thẳng qua CC và LS
   (polar                           ; Xác định điểm bằng toa độ cực
     LS                             ; từ điểm LS
     (angle LS CC)                  ; góc hợp bởi LS và CC
     (-                             ;có khoảng cách
       (distance LS CC)             ; bằng khoảng cách từ LS đến CC
       4                            ; trừ cho bán kính đường tròn là 4
     )                              ;
   )                                ;Đóng POLAR
   ^C ^C                            ;Hủy leader, hủy DIM
  )                                 ;Đóng COMMAND
)                                   ;Kết thúc chương trình
```

Hàm GETANGLE

Hàm **Getangle** yêu cầu nhập vào giá trị góc đo, hoặc chọn hai điểm trên màn hình bằng cách nhấn chuột. Hàm **Getangle** trả về góc đo tính bằng radian.

(**Getangle** [PT] [PROMPT])

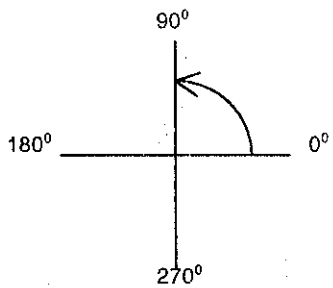
Ví dụ:

Command: **(getangle)**↵	Nhập giá trị góc đo (hoặc dùng chuột chọn hai điểm trên màn hình). Hàm trả về góc đo tính bằng radian.
180 ↵	
3.14159	
Command: **(getangle '(80 120))**↵	Sử dụng điểm thứ nhất (80,120) làm gốc tọa độ tương đối, trên màn hình xuất hiện "*sợi dây thun*" tương tự như ở hàm getpoint. Điểm thứ hai nhập vào bằng cách nhấn chuột hoặc nhập giá trị tọa độ tuyệt đối, tương đối hoặc tọa độ cực.
@-100,0 ↵	
3.14159	
Command: **(getangle "\nSpecify the angle: ")**↵	
Specify the angle:	Xuất hiện dòng nhắc trên màn hình.
Command: **(getangle '(80 120) "\nSpecify the angle: ")** ↵	
*Specify the angle:***@-100,0** ↵	Sử dụng cả hai tham số
3.14159	

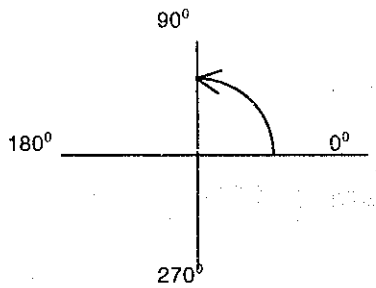
Tất cả các giá trị góc đo nhập vào cho hàm **Getangle** đều dựa theo các khai báo về đường chuẩn và hướng đo góc (các biến **ANGBASE** và **ANGDIR**). Tuy nhiên, giá trị trả về của hàm **Getangle** tuy vẫn tính theo đường chuẩn (biến ANGBASE) nhưng luôn luôn tính theo ngược chiều kim đồng hồ (không phụ thuộc vào biến ANGDIR).

✌ **Ví dụ:**

AUNITS = 0 (decimal degrees)
ANGBASE = 0 (Hướng đông)
ANGDIR = 0 (Ngược chiều kim đồng hồ)



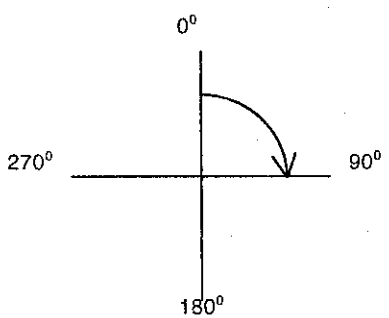
Góc nhập vào = 90⁰



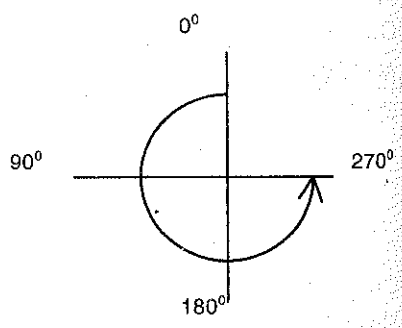
GETANGLE trả về 1.5708r (90⁰)

Command: **(getangle "\nIndicate rotation angle: ")**↵
Indicate rotation angle: **90**↵
1.5708

AUNITS = 0 (decimal degrees); ANGBASE = 90 (Hướng bắc)
ANGDIR = 1 (Chiều kim đồng hồ)



Góc nhập vào = 90⁰



GETANGLE trả về 4.71239r (270⁰)

Trong ví dụ thứ hai, khi ta nhập góc đo 90⁰, hàm **Getangle** trả về góc 270⁰.

Command: **(getangle "\nIndicate rotation angle: ")**↵
Indicate rotation angle: **90**↵
4.71239

Thông thường khi được yêu cầu nhập vào góc đo, người sử dụng nhập vào một số, ít khi chọn điểm trên màn hình. Do đó, nếu có thể ta nên dùng hàm **Getreal** để yêu cầu nhập một số biểu diễn góc đo.

Hàm GETORIENT

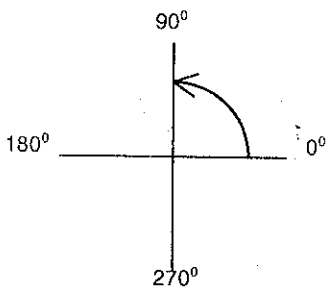
Hàm **Getorient** trả về số đo một góc dựa theo các khai báo về đường chuẩn ANGBASE = 0 và hướng đo góc ANGDIR = 0 (mặc định ban đầu của **AutoCAD**).

(**Getorient** [PT] [PROMPT])

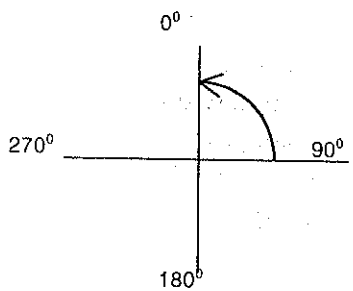
Hàm **Getorient** sử dụng tương tự như hàm **Getangle**. Điều khác biệt duy nhất, hàm **Getorient** luôn luôn trả về giá trị góc đo tính theo ANGBASE = 0 và ANGDIR = 0.

✌ **Ví dụ:**

AUNITS = 0 (decimal degrees); ANGBASE = 0 (Hướng đông)
ANGDIR = 0 (Ngược chiều kim đồng hồ)

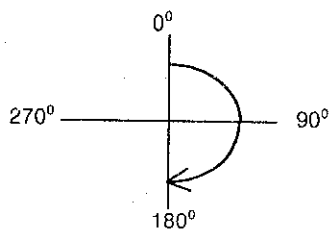


Góc nhập vào = 90⁰

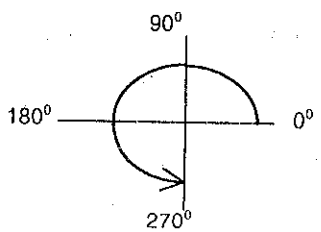


GETORIENT trả về 1.5708r (90⁰)

AUNITS = 0 (decimal degrees); ANGBASE = 90 (Hướng bắc)
ANGDIR = 1 (Cùng chiều kim đồng hồ)



Góc nhập vào = 180⁰



GETANGLE trả về 4.71239r (270⁰)

Ví dụ:

Command: **(getorient)**↵ Nhập giá trị góc đo (hoặc dùng chuột chọn hai điểm trên màn hình).

Command: **(getorient '(80 120))**↵ Sử dụng điểm thứ nhất (80 120) làm gốc tọa độ tương đối, trên màn hình xuất hiện "*sợi dây thun*" tương tự như ở hàm **Getpoint**. Điểm thứ hai nhập vào bằng cách nhấn chuột hoặc nhập giá trị tọa độ tuyệt đối, tương đối hoặc tọa độ cực.

Command: **(getorient "\nOrientation angle: ")**↵
Xuất hiện dòng nhắc *Orientation angle:* trên màn hình.

Command: **(getorient '(80 120) "\nOrientation angle: ")**↵
Xuất hiện dòng nhắc và sử dụng điểm (80, 120) làm gốc tọa độ tương đối.

Ví dụ:

Chương trình tính góc đo tạo bởi đường thẳng qua hai điểm và trục X.

```
;ABSANGLE.LSP
;Chương trình này định nghĩa hàm Rtd, dòng nhắc cho người sử dụng nhập
;hai điểm bằng các tọa độ X, Y, Z và  sau đó đưa ra hàm Getorient
;tìm lại góc tuyệt đối.
(defun rtd (A)
  (/ (* A 180.0) Pi)
)
(setq AGL (getorient "\nChoose two X, Y, Z coordinates: "))
(prompt "\nThe absolute angle is: ")(rtd AGL)
```

Hàm GETDIST

Hàm **Getdist** yêu cầu nhập vào giá trị khoảng cách, hoặc chọn hai điểm trên màn hình bằng cách nhấn chuột. Hàm **Getdist** luôn luôn trả về một số thực biểu diễn khoảng cách.

(**Getdist** [PT] [PROMPT])

Ví dụ:

Command: **(getdist)**↵ Nhập giá trị khoảng cách (hoặc dùng chuột chọn hai điểm trên màn hình).

Command: **(getdist '(5.0 7.0))**↵ Sử dụng điểm thứ nhất (5.0 7.0), trên màn hình xuất hiện "*sợi dây thun*" tương tự như ở hàm **Getpoint**. Điểm thứ hai nhập vào bằng cách nhấn chuột hoặc nhập giá trị tọa độ tuyệt đối, tương đối hoặc tọa độ cực.

Command: **(getdist "\nSpecify the distance: ")**↵
Xuất hiện dòng nhắc trên màn hình.

Command: **(getdist '(5.0 7.0) "\nSpecify the distance: ")**↵
Sử dụng cả hai tham số.

Hàm **Getdist** thường được sử dụng chỉ khi các điểm không có sẵn trước trong chương trình. Trong trường hợp ngược lại, tốt nhất chúng ta nên dùng hàm **Getpoint** gán các điểm cho các biến, sau đó dùng hàm **Distance** để tính khoảng cách.

Ví dụ: Tính khoảng cách giữa hai điểm nhập vào:

```
;Tên file: DISTONLY.LSP
;Chương trình yêu cầu nhập vào hai điểm
;Sau đó dùng hàm Getdist để tính khoảng cách giữa hai điểm này.
(defun C:DISTONLY (/ pt1 dst)
  (setq pt1 (getpoint "\nPick a starting point in your drawing: ")
        pt2 (getpoint pt1 "\nPick a second point in your drawing: ")
  )
  (prompt "\nThe specified distance is ")
  (distance pt1 pt2)
)
```

Ví dụ:

Chương trình tạo cửa ra vào trên tường hai nét đôi. Người sử dụng nhập chiều rộng cửa, sau đó chọn liên tiếp ba điểm như sau:

```
;Tên file: DOOR.LSP
;Các chức năng truy bắt điểm đối tượng như sau:
;       OSMODE =    0  = NONe
;       OSMODE =  32   = INTersection
```

```
;       OSMODE = 128  = PERpendicular
;       OSMODE = 512  = NEArest
;
(setvar "CMDECHO" 0)
(setq DORSIZ (getdist "\nSpecify door width: "))      ;Nhập chiều rộng cửa
(setvar "OSMODE" 32)
(setq PX (getpoint "\nPick intersection: "))          ;Chon điểm 1
(setvar "OSMODE" 512)
(setq P1 (getpoint "\nPick opening
                    start point (on same line): "))   ;Chon điểm 2
(setvar "LASTPOINT" P1)
(setvar "OSMODE" 128)
(setq P2 (getpoint "\nPick second line: "))           ;Chon điểm 3
(setvar "OSMODE" 0)
(setq  ANG1 (angle PX P1)                  ;Tính toán các điểm còn lai
       ANG2 (angle P1 P2)
       P3   (polar P1 ANG1 DORSIZ)
       P4   (polar P3 ANG2 (distance P1 P2))
)
(command ".break" P1 P3                    ;Vẽ cửa ra vào
         ".break" P2 P4
          ".line" P1 P2 ""
         ".line" P3 P4 ""
) ; Kết thúc chương trình
```

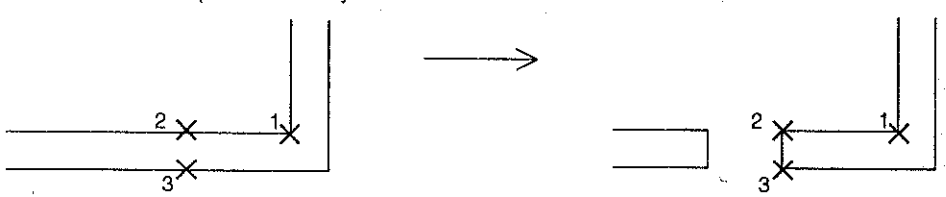
Khi thực hiện chương trình xuất hiện các dòng nhắc sau:

Specify door width: **100** ↵

Pick intersection: (Bắt điểm 1)

Pick opening start point (on same line): (Bắt điểm 2)

Pick second line: (Bắt điểm 3)



Tóm tắt:

1. Hằng số Pi = 3.1415926 cần thiết trong các phép chuyển đổi đơn vị đo góc. Ta không nên sử dụng các phép gán để thay đổi giá trị của Pi.

2. Các hàm **Angle**, **Getangle** trả về số đo góc tính bằng radian. Hàm **Getangle** phụ thuộc vào biến ANGBASE và luôn luôn tính góc theo ngược chiều kim đồng hồ.
3. Hàm **Getorient** trả về số đo góc tính bằng radian và luôn luôn trả về giá trị góc đo tính theo ANGBASE = 0 và ANGDIR = 0.
4. Hàm **Distance** và **Getdist** trả về khoảng cách ở dạng số thực.
5. Hàm **Polar** dùng toạ độ cực để tạo ra toạ độ điểm mới tính theo toạ độ điểm ban đầu.

5.3 Truy bắt điểm đối tượng và giao điểm giữa hai đường thẳng

Để tăng cường sự chính xác khi chọn điểm, ta nên dùng chức năng truy bắt điểm đối tượng.

Hàm OSNAP

Hàm **Osnap** cung cấp phương thức bắt điểm đối tượng khi chọn điểm và trả về điểm 3D.

(**Osnap** PT MODE-STRING)

Khi thực hiện hàm **Osnap**, ta hình dung tâm của ô vuông truy bắt (aperture) đang trùng với điểm PT. Điểm nào ở gần điểm PT nhất trong số các điểm thỏa mãn các đặc điểm truy bắt điểm đối tượng chứa trong tham số MODE-STRING sẽ được chọn. Nếu ô vuông truy bắt không chọn được điểm nào thì hàm **Osnap** trả về nil.

Ví dụ:

Command: **(OSNAP '(50 40) "CEN")**↵
(30 40 0)

Ô vuông truy bắt ở tại điểm (50 40). Tâm đường tròn được chọn.

Command: **(OSNAP '(60 30) "INT")**↵
nil

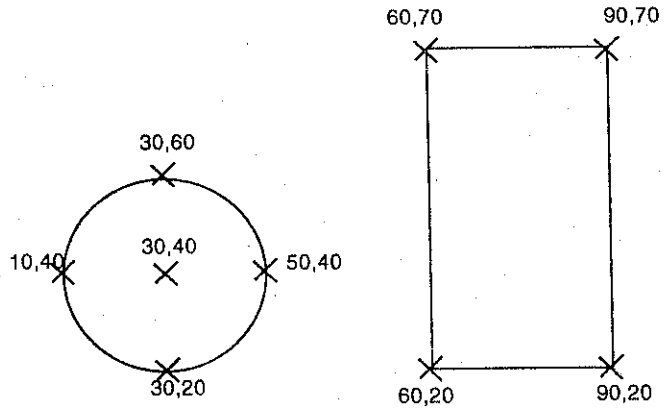
Không tìm thấy điểm INTersection trong ô vuông truy bắt.

Command: **(OSNAP '(60 30) "MID,END")**↵
(60 20 0)

Có thể chứa nhiều chế độ truy bắt, cách nhau bằng các dấu phẩy. Điểm ENDpoint gần nhất sẽ được chọn.

Command: **(OSNAP '(60 40) "MID,END")**↵

Điểm MIDpoint gần nhất sẽ được chọn.



(60 45 00)
Command:**(OSNAP'(60 40) "MID,END,NEA")**↵ Điểm NEArest luôn luôn là điểm gần nhất được chọn
(60 40 0)

Chú ý

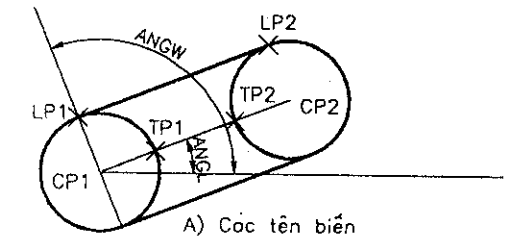
- Hàm **Osnap** chỉ nhận biết được các đối tượng đang xuất hiện trên màn hình. Các đối tượng ở các lớp bị tắt hoặc bị đóng băng sẽ không thể chọn được.
- Hàm **Osnap** không làm thay đổi phương thức bắt điểm thường trú đang có trên bản vẽ.

Trong nhiều trường hợp, thay cho việc dùng hàm **Osnap**, ta nên dùng hàm **Setvar** gán giá trị cho biến hệ thống OSMODE sau đó dùng hàm **Getpoint** để nhập tọa độ điểm. Giá trị gán cho biến OSMODE bằng tổng các *bit code* tương ứng với các phương thức bắt điểm.

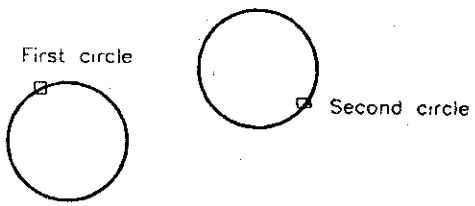
0	=	NONe	128	=	PERpendicular
1	=	ENDpoint	256	=	TANgent
2	=	MIDpoint	512	=	NEArest
4	=	CENter	1024	=	QUICK
8	=	NODe	2048	=	APParent intersection
16	=	QUAdrent			
32	=	INTersection			
64	=	INSertion			

Ví dụ:

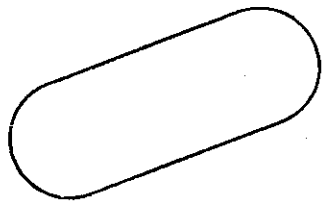
Chương trình khoét rãnh từ hai lỗ khoan có sẵn:



A) Các tên biến



B) Trước MAKESLOT



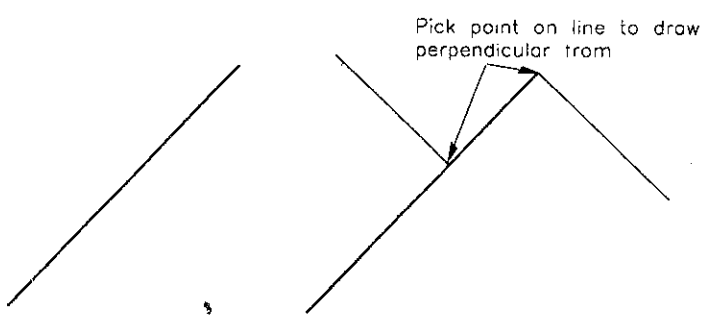
C) Sau MAKESLOT

```
;Tên file: MAKESLOT.LSP
;Mục đích:  Khoét rãnh từ hai lỗ khoan có sẵn
;
;Chú ý: Hai đường tròn phải có đường kính bằng nhau.
;Các biến cục bộ:      NP1 – Điểm chọn trên đường tròn thứ nhất
;                      NP2 – Điểm chọn trên đường tròn thứ hai
;                      CP1 – Tâm đường tròn thứ nhất
;                      CP2 – Tâm đường tròn thứ hai.
;                     ANGL – Góc tạo bởi đường thẳng đi qua hai tâm.
;                     ANGW – Góc tạo bởi đường thẳng vuông góc với
;                            đường thẳng trên
;                     CRAD – Bán kính đường tròn
;                      LP1 – Điểm đầu của đường thẳng thứ nhất
;                      LP2 – Điểm cuối của đường thẳng thứ nhất
;                      OP1 – Điểm chọn của lệnh offset
;                      TP1 – Điểm xén bỏ đường tròn thứ nhất
;                      TP2 – Điểm xén bỏ đường tròn thứ hai
(defun C:MAKESLOT (/ NP1 NP2 CP1CP2 ANGL ANGW CRAD LP1 LP2
OP1 TP1 TP2)
  (setvar "OSMODE" 512)              ;Truy bắt NEA
  (setq NP1 (getpoint "\nPick a point on the first circle: ")
        NP2 (getpoint "\nPick a point on the second circle: ")
  )
  (setvar "OSMODE" 0)                ;Truy bắt NONe
  (setq CP1 (osnap NP1 CEN")         ;Tâm đường tròn 1
```

```
        CP2 (osnap NP2 "CEN")          ;Tâm đường tròn 2
        ANGL (angle CP1 CP2)           ;Góc tạo bởi đường thẳng CP1, CP2
        ANGW (+ (* 0.5 Pi) ANGL)       ;
        CRAD (distance CP1 NP1)        ;Bán kính đường tròn 1
        LP1 (polar CP1 ANGW CRAD);Điểm đầu đường thẳng 1
        LP2 (polar LP1 ANGL (distance CP1 CP2));Điểm cuối đường thẳng 1
    )                                  ;Đóng setq
    (command ".line" LP1 LP2 ""
             ".select" "last" ""       ; Chọn đường thẳng LP1 LP2 vừa vẽ
    )
    (setq OP1 (osnap LP1 "MID"))       ;Điểm offset
    (command ".offset" (* 2 CRAD) OP1 CP1 "");Vẽ đường thẳng thứ 2
    (setq TP1 (polar CP1 ANGL CRAD)            ;Điểm xén 1
          TP2 (polar CP2 (+ pi ANGL) CRAD)     ;Điểm xén 2
    )
    (command ".trim" "p" "last" "" TP1 TP2 "");Chon các canh cắt
    (prompt "\nDone.\n")                       ;Thông báo kết thúc
)
;Kết thúc file MAKESLOT.LSP
```

Ví dụ:

Sử dụng chương trình này để vẽ line vuông góc với line sẵn có. Khi ta chọn một điểm bất kỳ trên line sẵn có thì **AutoCAD** tự động tính góc của line được chọn và vẽ đường vuông góc với line này (tham khảo thêm hàm **If** chương 8).




```
; Tên file: PERPL.LSP
(Defun C:PERPL (/ SA SB SNP OM OS PT1 PT2)
  (setvar "cmdecho" 0)
  (setq
    SA  (getvar "snapang")
    SB  (getvar "snapbase")
    SNP (getvar "snapmode")
    OM  (getvar "orthomode")
    OS  (getvar "osmode")
    PT1 (osnap (getpoint
          "\nPick point on line to draw perpendicular from: "
        )
        "nea"
    )
  )
  (setvar "osmode" 0)
  (setq PT2 (osnap PT1 "end"))
  (if (equal PT1 PT2)
    (setq PT2 (osnap PT1 "MID"))
  )
  (command ".snap" "r" PT1 PT2)
  (setvar "snapmode" 0)
  (setvar "orthomodE" 1)
  (prompt "\nto point:")
  (command ".line" PT1 pause "")
  (setvar "snapang" SA)
  (setvar "snapbase" SB)
  (setvar "snapmode" SNP)
  (setvar "orthomode" OM)
  (setvar "osmode" OS)
  (setvar "cmdecho" 1)
  (princ)
); Kết thúc file PERPL.LSP
```

Hàm INTERS

Hàm **Inters** trả về giao điểm giữa hai đường thẳng không song song.

(**Inters** PT1 PT2 PT3 PT4 [ONSEG])

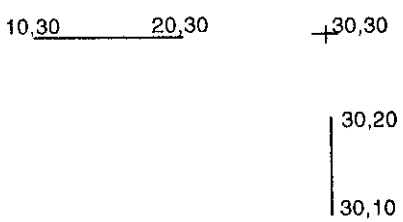
PT1 và PT2 là hai điểm tạo ra đường thẳng thứ nhất.
PT3 và PT4 là hai điểm tạo ra đường thẳng thứ hai.

Nếu PT1, PT2, PT3, PT4 là các điểm 3D thì hàm **Inters** sẽ trả về giao điểm giữa hai đường thẳng trong không gian 3D. Ngược lại, các đường thẳng sẽ được chiếu lên mặt phẳng XY, sau đó hàm **Inters** sẽ tìm giao điểm giữa các đường thẳng chiếu này.

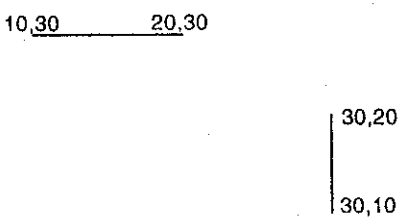
Nếu tham số ONSEG = nil, các đường thẳng sẽ được kéo dài để tìm giao điểm. Ngược lại, nếu tham số ONSEG <> nil hoặc không được cung cấp, các đường thẳng không được kéo dài khi tìm giao điểm.

Ví dụ:

(inters '(30 10) '(30 20) '(10 30) '(20 30) t) trả về nil
(inters '(30 10) '(30 20) '(10 30) '(20 30)) trả về nil
(inters '(30 10) '(30 20) '(10 30) '(20 30) nil) trả về (30 30)
(inters '(20 20 0) '(30 30 20) '(60 20 0) '(50 30 20) nil) trả về (40 40 40)
(inters '(20 20) '(30 30 20) '(60 20 0) '(50 30 20) nil) trả về (40 40)



ONSEG = nil, hàm **Inters** trả về giao điểm (30, 30)



ONSEG <> nil , không tìm thấy giao điểm, hàm **Inters** trả về nil

Tóm tắt

1. Hàm **Osnap** cung cấp chức năng truy bắt điểm đối tượng khi chọn điểm và trả về điểm 3D. Hàm này có nhiều hạn chế khi sử dụng, nên ta phải cẩn thận khi dùng nó.
2. Ta có thể thay thế hàm **Osnap** bằng cách dùng phối hợp hàm **Getpoint** với biến hệ thống OSMODE.
3. Hàm **Inters** dùng để tìm giao điểm giữa hai đường thẳng. Tham số ONSEG cho phép lựa chọn chế độ kéo dài hoặc không kéo dài các đường thẳng trong khi tìm giao điểm.

5.4 Bài tập

1. Dựa theo chương trình DISTONLY.LSP, hãy viết chương trình tính góc tạo bởi đường thẳng qua hai điểm và trục x.
2. Hãy tính các biểu thức sau:
   (setq PT1 '(3.0 5.0 0.0) PT2 '(12.0 23.0 0.0) M 90 N 270)
   a. (angle PT1 PT2)
   b. (rtd (angle PT1 PT2))
   c. (setq A (rtd (angle PT2 PT1)))
   d. (setq B (dtr A))
   e. (setvar "ANGBASE" N)
   f. (setq C (angle '(0.0 0.0 0.0) PT1))
   g. (distance PT1 PT2)
   h. (setq D (distance PT2 PT1))
   i. (setq E (+ D (distance '(0 0 0) PT1)))
   j. (setvar "ANGBASE" M)
   k. (* B (/ 180.0 PI))
   l. (* PI (/ 180.0 PI))

3. Hãy sửa lại chương trình MAKESLOT.LSP dùng cho hai đường tròn có đường kính khác nhau. (Gợi ý: vẽ đường thẳng tiếp xúc hai đường tròn). Lưu file với tên MAKESLOT2.LSP
4. Hãy tính các biểu thức sau:
   (setq PT1 '(85 10 65) PT2 '(125 70 65) A 90 B 270)
   (setq PTM '(15 90 65) PTN '(45 10 65))
   Giả sử có một đường thẳng qua hai điểm PT1 và PT2
   a. (osnap PT2 "NEA")
   b. (setq PT3 (osnap PT1 "MID"))
   c. (polar PT1 (angle PT2 PT1) 25)
   d. (setq PT4 (polar PT2 (dtr A) 55))
   e. (osnap PT1 "CEN")
   f. (osnap PT1 "CEN,END")
   g. (setq PT5 (polar PT1 (dtr B) 30))
   h. (polar PT2 PI (distance PT2 PT1))
   i. (inters PT1 PT2 PTM PTN)
   j. (inters PT1 PT2 PTM PTN nil)
   k. (inters PT1 PTM PT2 PTN)
   l. (inters PT1 PTM PT2 PTN nil)

5. Nghiên cứu chương trình vẽ ống 3D như sau:

```
;Tên chương trình: PIPSLOPE.LSP
;Muc đích :      Tính và vẽ ống 3D bằng cách quét đường tròn
;
(defun C:PIPSLOPE (/ SLP RIS P1 P2 DIS ANG SIZ LU LP TH)
  (setvar "CMDECHO" 0)
;
; Giữ các đơn vị và chính xác mặc định
```

```
  (setq LP (getvar "luprec")
        LU (getvar "lunits")
        TH (getvar "thickness")
  )
 (setvar "lunits" 2)
(setvar "luprec" 1)

;Goi cac tham so de tao ong 3D
  (setq P1      (getpoint "\nStart point for sloped eD pipe: ")
        ANG     (getorient "\nEnter numeric X-Y angle from start point: ")
        DIS     (getdist "\nEnter linear pipe distance in X-Y plane: ")
        SLP     (getreal "\nDecimal slope per foot [Specify + or -]: ")
        SIZ     (getstring "\nDiameter size of pipe: ")
        P2      (polar P1 ANG DIS)
        RIS     (* (/ DIS 12.0) SLP)
        P2      (list (car P2) (cadr P2) (+ (caddr P2) RIS))
  )
  (setvar "thickness" (distance P1 P2))
  (command ".ucs" "za" P1 P2
           ".circle" "0,0,0" "d" SIZ
           ".ucs" ""
  )
;Goi lai mac dinh
  (setvar "lunits" LU)
  (setvar "luprec" LP)
  (setvar "thickness" TH)
  (setvar "cmdecho" 1)
  (prompt "\nDone.\n")
)
; Kết thúc file PIPSLOPE.LSP
```

5.5 Lời giải

1.

```
;Tên file: ANGLONLY.LSP
;Chương trình yêu cầu nhập vào một điểm
;Sau đó dùng hàm Getangle và chon điểm thứ hai để tính góc
;
(defun rtd (A)  ;Đổi radian sang độ
  (/ (* A 180.0) Pi)
)
(defun C:ANGLONLY (/ pt1 agl)
  (setq pt1 (getpoint "\nPick a starting point in your drawing: "))
  (rtd (getangle pt1 "\nthe angle is "))  ;Chọn điểm thứ hai
)
```

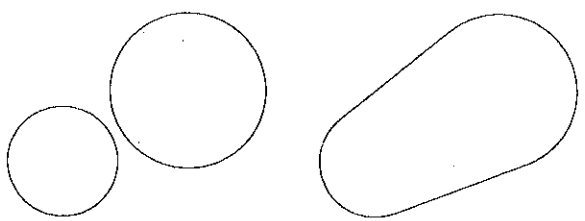
2.

a. 1.10715	g. 20.1246
b. 63.4349	h. 20.1246
c. 243.435	i. 25.9556
d. 4.24874	j. 90
e. 270	k. 243.435
f. 1.03038	l. 180.0

3.

```
;Tên file: MAKESLOT2.LSP
;Mục đích:  Tạo rãnh từ hai đường tròn sẵn có
;
;Các dòng nhắc: Pick a point on the first circle
;                Pick a point on the second circle
;Các biến cục bộ:      NP1 – Điểm chon trên đường tròn thứ nhất
;                      NP2 – Điểm chon trên đường tròn thứ hai
;                      CP1 – Tâm đường tròn thứ nhất
;                      CP2 – Tâm đường tròn thứ hai
;                    ANGL – Góc tạo bởi đường thẳng đi qua hai tâm
;                    ANGW – Góc tạo bởi đường thẳng vuông góc với
;                           đường thẳng trên
;                   CRAD – Bán kính đường tròn
;                     LP1 – Điểm đầu của đường thẳng thứ nhất
;                     LP2 – Điểm cuối của đường thẳng thứ nhất
;                     OP1 – Điểm chon của lệnh offset
;                     TP1 – Điểm xén bỏ đường tròn thứ nhất
;                     TP2 – Điểm xén bỏ đường tròn thứ hai
;
(defun C:MAKESLOT (/ NP1 NP2 CP1 CP2 ANGL ANGW1 ANGW2
CRAD1 CRAD2 LP1 LP2 OP1 TP1 TP2)
  (setvar  "OSMODE" 512)              ;Bắt điểm NEA
  (setq NP1 (getpoint "\nPick a point on the first circle: ")
        NP2 (getpoint "\nPick a point on the second circle: ")
  )
  (setvar "OSMODE" 0)                 ; Bắt điểm NONe
  (setq CP1 (osnap NP1 "CEN")         ; Bắt điểm CEN của đường tròn 1
```

```
        CP2 (osnap NP2 "CEN")          ; Bắt điểm CEN của đường tròn 2
        ANGL (angle CP1 CP2)           ;Góc giữa hai điểm CP1 và CP2
        ANGW (+ (* 0.5 Pi) ANGL)       ;Góc vuông góc đường thẳng trên
        CRAD1 (distance CP1 NP1)       ;Bán kính đường tròn  1
        LP1 (polar CP1 ANGW CRAD1);Điểm tiếp xúc thứ 1
        CRAD2 (distance CP2 NP2)       ;Bán kính đường tròn thứ 2
        LP2 (polar CP2 ANGW CRAD2);Điểm tiếp xúc thứ hai
    )
    (command ".line" "Tan" LP1 "Tan" LP2 ""
             ".select" "last" ""                ; Tao nhóm đối tượng chon
           ".mirror" "last" "" CP1 CP2 "N"
    )
    (setq TP1 (polar CP1 ANGL CRAD1);Xén tại điểm  1
          TP2 (polar CP2 (+ pi ANGL) CRAD2)    ;Xén tai điểm 2
    )
    (command ".trim" "p" "last" "" TP1 TP2 "");Chọn đường thẳng làm cạnh cắt
    (prompt "\nDone.\n")                  ;dòng nhắc sau khi thực hiện xong
)
;Kết thúc file MAKESLOT2.LSP
```



a) Trước Makeslot

b) Sau Makeslot

4.

a. (125.0 70.0 65.0)
b. (105.0 40.0 65.0)
c. (71.1325 -10.8013 65.0)
d. (125.0 125.0 65.0)
e. nil
f. (85.0 10.0 65.0)
g. (85.0 -20.0 65.0)
h. (52.889 70.0 65.0)
i. nil
j. (59.4 -28.4 65.0)
k. (69.1509 28.1132 65.0)
l. (69.1509 28.1132 65.0)

Chương 6

CÁC HÀM TOÁN HỌC

Nội dung chương

1. Các hàm kiểm soát dạng số: **Fix**, **Float**, **Abs**, **Rem**, **Gcd**, **Max**, **Min**.
2. Các hàm lượng giác: **Sin**, **Cos**, **Atan**.
3. Các hàm lũy thừa, khai căn, logarit: **Expt**, **Sqrt**, **Log**, **Exp**.

6.1 Các hàm kiểm soát dạng số

Nhiều chương trình ứng dụng yêu cầu giá trị nhập cho tham số kiểu số phải có dạng thức đúng.

Hàm FIX

Hàm **Fix** trả về phần số nguyên của một số. Phần số nguyên này không được làm tròn.

(**Fix** NUMBER)

Ví dụ:

(fix 52.14)	trả về 52
(fix 52.98)	trả về 52
(fix (/ 80 6))	trả về 13
(fix 20)	trả về 20

Hàm này có thể xem như là bộ lọc dữ liệu. Dữ liệu là số nguyên hoặc số thực sẽ được lọc thành số nguyên trước khi được gán cho một tham số.

Ta cũng có thể dùng hàm này để lấy ra phần thập phân của một số.

Ví dụ:

Command: **(setq N 25.457)(setq N (- N (fix N)))**↵
0.457

Hàm FLOAT

Hàm **Float** chuyển một số có kiểu số nguyên hoặc số thực thành kiểu số thực. Nói cách khác, các số khi đi qua bộ lọc này sẽ chuyển thành số thực.

(**Float** NUMBER)

Ví dụ:

(float 45)	trả về 45.0
(float 32.375)	trả về 32.375
(float (/ 50 25))	trả về 2.0

Hàm ABS

Hàm **Abs** trả về giá trị tuyệt đối của một số. Nói cách khác, các số dù âm hay dương khi đi qua bộ lọc này sẽ chuyển thành số dương.

(**Abs** NUMBER)

Ví dụ:

(abs 45)	trả về 45
(abs -45)	trả về 45
(abs –45.0)	trả về 45.0
(abs (* -20.0 20.0))	trả về 400.0
(abs (- 10 50))	trả về 40

6.2 Các hàm khác

Hàm REM

Hàm **Rem** trả về số dư của phép chia. Cách dùng tương tự như hàm chia (/):

(**Rem** NUMBER NUMBER...)

Ví dụ:

(rem 70 8)	trả về 6
(rem 36 9)	trả về 0
(rem 36.0 9)	trả về 0.0
(rem 40 7.0)	trả về 5.0
(rem 230.0 80)	trả về 70.0
(rem 230.0 80 20)	230.0 chia cho 80 có số dư là 70.0. Tiếp theo 70.0 chia cho 20 có số dư là 10.0. Do đó kết quả trả về là 10.0
(rem 620.0 500 50 7 5)	620.0 chia cho 500 có số dư là 120.0 120.0 chia cho 50 có số dư là 20.0 20.0 chia cho 7 có số dư là 6 6 chia cho 5 có số dư là 1.0 Giá trị trả về là 1.0

Ta có thể dùng hàm **Fix** để định phần thương số của phép chia, dùng hàm **Rem** để lấy phần số lẻ của phép chia.

Ví dụ:

```
(setq Q (fix (/ 27.5 5)))
(setq R (rem 27.5 5))
```

Hàm GCD

Hàm **Gcd** *(Greatest Common Denominator)* trả về ước số chung lớn nhất của hai số nguyên.

(**Gcd** NUMBER1 NUMBER2)

Ví dụ:

(gcd 64 96)	trả về 32
(gcd 20 15)	trả về 5
(gcd 51 17)	trả về 17
(gcd 8 2 234.315)	trả về 2. Chỉ sử dụng hai tham số. Các tham số còn lại bị bỏ qua.

Hàm MAX

Hàm **Max** trả về giá trị lớn nhất trong các số.

(**Max** NUMBER NUMBER...)

Ví dụ:

(max 45 12)	trả về 45
(max 12 460 40)	trả về 460
(max 5.0 98)	trả về 98.0 (số thực)
(max "XY" "XYZ")	không chấp nhận chuỗi

Hàm MIN

Hàm **Min** trả về giá trị nhỏ nhất trong các số.

(**Min** NUMBER NUMBER ...)

Ví dụ:

(min 12 365 67)	trả về 12
(min 92.13 65 18)	trả về 18.0 (số thực)

6.3 Các hàm lượng giác

Hàm SIN

Hàm **Sin** trả về giá trị sin của một góc.

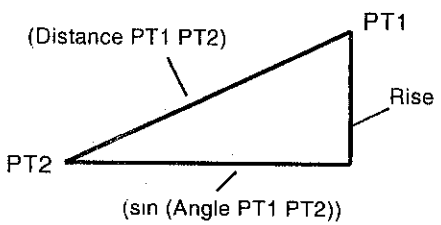
(**Sin** ANGLE)

Giá trị trả về là số thực. Tham số ANGLE phải được tính bằng radian.

Ví dụ:

(sin 0.4507)	trả về 0.435596
(sin (/ pi 4))	trả về 0.707107
(sin (cvunit 60 "DEGREES" "RADIANS"))	trả về 0.866025
(sin (dtr 90.0))	trả về 0.5

Ví dụ: Tính chiều cao (rise) của đường thẳng nghiêng.




```
;Tên file: RISE.LSP
;(sin (angle PT1 PT2)) là sin của góc tạo bởi đường thẳng nghiêng và
;đường thẳng nằm ngang
;(distance PT1 PT2) là cạnh huyền
;tích số của hai giá trị trên là chiều cao của đường thẳng nghiêng (rise)
(defun C:RISE (/ PT1 PT2)
  (setvar "OSMODE" 1)        ;Bắt điểm ENDpoint
  (setq PT1 (getpoint "\nPick highest endpoint: "))
  (setq PT2 (getpoint "\nPick lowest endpoint: "))
  (setvar "OSMODE" 0)        ;Bắt điểm NONe
  (prompt "\nThe total rise of this line is: ")
  (abs (* (distance PT1 PT2) (sin (angle PT1 PT2))))
)
```

Hàm COS

Hàm **Cos** trả về giá trị cosin của một góc.

(**Cos** ANGLE)

Giá trị trả về là số thực. Tham số ANGLE phải được tính bằng radian.

Ví dụ:

(cos (/ Pi 8))	trả về 0.92388
(cos (cvunit 30 "DEGREES" "RADIANS"))	trả về 0.866025
(cos (dtr 60.0))	trả về 0.5

Hàm ATAN

Hàm **Atan** trả về giá trị arc tang của một góc.

(**Atan** NUM1 [NUM2])

Góc trả về tính bằng đơn vị radian, có giá trị từ pi/2 ($+90^0$) đến –pi/2 (-90^0).

Ví dụ:

(rtd (atan 1))	trả về 45.0
(rtd (atan 4.0))	trả về 75.9638
(atan -1)	trả về –0.785398
(cvunit (atan –1) "RADIANS" "DEGREES")	trả về –45.0

Nếu có tham số NUM2, hàm sẽ trả về giá trị arc tang của phép chia NUM1/NUM2.

Ví dụ:

(atan 1 1)	trả về 0.785398
(cvunit (atan 60 15) "RADIANS" "DEGREES")	trả về 75.9638
(cvunit (atan 11 0) "RADIANS" "DEGREES")	trả về 90.0

6.4 Các hàm lũy thừa, khai căn, logarit

Các hàm lũy thừa, khai căn, logarit thường dùng để giải các phương trình phức tạp.

Hàm EXPT

Hàm **Expt** dùng để tính lũy thừa của một số.

(**Expt** BASE POWER)

Ví dụ:

(expt 4 2)	trả về $(4)^2 = 16$
(expt 5.0 2)	trả về $(5.0)^2 = 25.0$
(expt 1.5 4)	trả về $(1.5)^4 = 5.0625$
(expt 8 3)	trả về $(8)^3 = 512$
(expt 2 –4.0)	trả về $(2)^{-4} = 0.0625$
(expt 2 0.5)	trả về $(2)^{0.5} = 1.41421$
(expt 2 2.5)	trả về $(2)^{2.5} = 5.65685$

Hàm SQRT

Hàm **Sqrt** trả về căn bậc hai của một số.

(**Sqrt** NUMBER)

Ví dụ:

(sqrt 2.0)	trả về 1.41421
(sqrt 16)	trả về 4.0
(sqrt –4)	trả về lỗi
(sqrt (abs –4))	trả về 2.0

Hàm LOG

Hàm **Log** trả về giá trị logarit của một số.

(**Log** NUMBER)

Ví dụ:

(log 6)	trả về 1.79176
(log 9.33)	trả về 2.23324
(log 21)	trả về 3.04452
(log -2)	trả về lỗi

Hàm EXP

Hàm **Exp** trả về giá trị lũy thừa e^n.

(**Exp** NUMBER)

Ví dụ:

(exp 1.79176) trả về 5.99904 (= $e^{i.79176}$)
(exp (log 6)) trả về 6.0
(exp 4.2) trả về 66.6863 (= $e^{4.2}$)
(exp 1) trả về 2.71828 (= e^{i})

Ví dụ:

Chương trình khai căn một số có bậc bất kỳ.

```
; Tên file: ROOT.LSP
(defun C:ROOT (/ NUM R)
  (setq NUM (abs (getreal "\nBase: ")))
  (setq R (getreal "\nRoot level: "))
  (prompt "\nThe specified root is:")
  (exp (/ (log NUM) R))
)
```

Tóm tắt:

1. Các hàm **Fix**, **Float**, **Abs** cho phép chương trình kiểm soát dạng thức các dữ liệu nhập là số.
2. Các hàm lượng giác cơ bản là **Sin**, **Cos**, **Atan**.
3. Các hàm **Sqrt**, **Exp**, **Log** thường dùng để giải các phương trình phức tạp.

6.5 Ví dụ mẫu

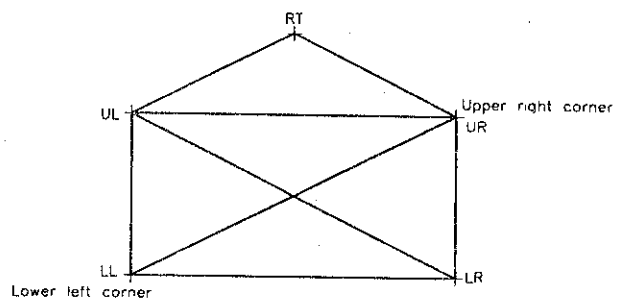
Trong mục này ta khảo sát hai ví dụ mẫu.

Ví dụ 1: Vẽ hình dạng khai triển hình bao thư sử dụng các hàm toán học và các hàm xử lý danh sách:

```
; Tên file: LETTER.LSP
(defun C:LETTER (/ LL UR LR UL RT)
(Setvar "BLIPMODE" 0)
(Setvar "CMDECHO" 0)
(Setq LL (getpoint "\Lower left corner:"))
(Setq UR (getcorner "\Upper right corner:" LL))
(Setq LR (list (car UR) (cadr LL)))
(Setq UL (list (car LL) (cadr UR)))
```



```
(Setq RT
  (list
    (+
     (car UL)
     (/ (- (car UR) (car UL)) 2)
    )
    (+
     (cadr UL)
     (/ (- (cadr UL) (cadr LL)) 2)
    )
  )
 )
(command "pline"
  LL LR UR LL UL RT UR UL LR ""
 )
(setvar "BLIPMODE" 1)
(setvar "CMDECHO" 1)
)
```



Ví dụ 2: Viết chương trình vẽ đa giác đều POLYGON, yêu cầu nhập vào số cạnh, diện tích đa giác và điểm bắt đầu của một cạnh. Biết rằng bán kính đường tròn nội tiếp đa giác được tính theo công thức sau:

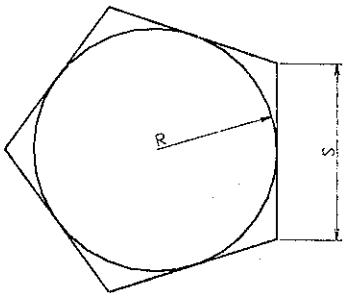
$$R = \sqrt{\frac{A \cos \frac{Pi}{N}}{N(\sin \frac{Pi}{N})}}$$

trong đó, A = diện tích; N = số lượng các cạnh; R = bán kính đường tròn nội tiếp; Pi = 180^0= 3.14159 rad.

Chiều dài một cạnh được tính theo công thức sau:

$$S = 2R\left(\frac{\sin\frac{Pi}{N}}{\cos\frac{Pi}{N}}\right)$$

trong đó, R = bán kính đường tròn nội tiếp; S = chiều dài cạnh; Pi = 180^0.




```
;Tên file: POLYAREA.LSP
;Vẽ đa giác đều
(defun C:POLYAREA (/ AREA NUM RAD SIDE SP EP)
  (setq AREA (getreal "\nEnter area: ")          , Nhập diện tích đa giác
        NUM  (getint "\nEnter number of sides: "); Nhập số cạnh
        SP   (getpoint "\nSpecify starting point of edge: "); Nhập điểm đầu
        EP   (getpoint SP "\nSpecify second point to indentify direction: ")
        RAD  (sqrt (/ AREA (* NUM (/ (sin (/ Pi NUM)) (cos (/ pi NUM))))))
        SIDE (* 2 RAD (/ (sin (/ Pi NUM)) (cos (/ Pi NUM))))
        EP   (polar SP (angle SP EP) SIDE)
  )
  (command ".polygon" NUM "E" SP EP)
)
```

6.6 Bài tập

1. Hãy tính các biểu thức sau:

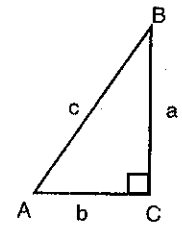
a. (fix (- 33.0 30))
b. (float (- 24))
c. (fix (+ 1.5 2.5))
d. (abs (- 18.5 19))
e. (abs (- 19 18.5))
f. (float (+ 3 4 5))
g. (fix (* 8.0 7.0))
h. (abs (* 4 5 6))
i. (fix (/ 25.0 5))
l. (float (/ 18 5))

2. Viết chương trình tìm giá trị lớn nhất trong các biến hệ thống **AutoCAD**:
a. Chamfera

b. Chamferb
c. Filletrad
d. Circlerad

Gợi ý: Dùng hàm **Getvar** để trả về giá trị các biến hệ thống. Đặt tên file là EX6-2.LSP.

3. Cho tam giác như sau:
a. Cho a = 50 A = 53.13^0
Tính c.
b. Cho b = 100 A = 63.435^0
Tính c.



4. Viết chương trình có tên HYP.LSP tạo ra một hàm C:HYP dùng để tính cạnh huyền của một tam giác vuông khi biết hai cạnh góc vuông.

5. Viết chương trình có tên TRI-AREA.LSP tạo ra một hàm C:TA dùng để tính diện tích tam giác khi biết ba cạnh.

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

trong đó, s = (a + b + c)/2 và a, b, c là ba cạnh tam giác.

6. Hãy viết chương trình EXPONENT.LSP tính lũy thừa a^x

6.7 Lời giải

1.
a. 3
b. –24.0
c. 4
d. 0.5
e. 0.5
f. 12.0
g. 56
h. 120
i. 5
j. 3.6

2.
```
;Tên file: EX6-2.LSP
(prompt "\nMaximum value is: ")
(max (getvar "Chamfera") (getvar "Chamferb")
     (getvar "Filletrad") (getvar "Circlerad")
)
```

3.
a. (/ a (sin A))
b. (/ b (cos A))

4.

```
;Tên file: HYP.LSP
;tính cạnh huyền của tam giác vuông
(defun C:HYP (/ a b)
  (setq a (getreal "\nEnter length of leg: ")
        b (getreal "\nEnter length of other leg: ")
  )
  (prompt "\nHypotenuse is: ")
  (sqrt (+ (expt a 2) (expt b 2)))
)
```

5.

```
;Tên file: TRI-AREA.LSP
;Tính diện tích tam giác
(defun C:TA (/ a b c s)
  (setq a (getreal "\nEnter first side: ")
        b (getreal "\nEnter second side: ")
        c (getreal "\nEnter third side: ")
          s (* 0.5 (+ a b c))
  )
  (prompt "Area is: ")
  (sqrt (* s (- s a) (- s b) (- s c)))
)
```

6.

```
;Tên file: EXPONENT.LSP
;tính a^x
(defun C:exponent (/ a x)
  (setq a (getreal "\nBase: ")
        x (getreal "\nPower: ")
  )
  (prompt "Exponent is: ")
  (expt a x)
)
```

Chương 7

CHUYỂN ĐỔI KIỂU DỮ LIỆU VÀ XỬ LÝ CHUỖI

Nội dung chương

1. Sự cần thiết phải chuyển đổi kiểu dữ liệu và xử lý chuỗi.
2. Các hàm chuyển đổi kiểu dữ liệu: **Atof**, **Distof**, **Atoi**, **Rtos**, **Itoa**, **Angtos**, **Angtof**, **Ascii**, **Chr**, **Read**.
3. Các hàm hiển thị thông tin kiểu chuỗi: **Prin1**, **Princ**, **Print**.
4. Các hàm xử lý chuỗi: **Strcase**, **Strcat**, **Strlen**, **Substr**, **Acad_strlsort**.

7.1 Sự cần thiết phải chuyển đổi kiểu dữ liệu và xử lý chuỗi

AutoLISP cung cấp các hàm **Getreal** và **Getstring** để nhập số thực và chuỗi. Dữ liệu nhập vào có thể sử dụng cho các hàm **AutoLISP** hoặc các lệnh **AutoCAD**. Tuy nhiên, trên thực tế, dữ liệu có thể nhập vào từ các nguồn khác và không đúng các dạng thức mong muốn. Ví dụ để hiển thị giới hạn bản vẽ, ta lấy thông tin từ các biến hệ thống LIMMAX và LIMMIN có kiểu danh sách các số thực. Hàm **Prompt** chỉ hiển thị thông tin kiểu chuỗi, nên không thể dùng trong trường hợp này. Do đó ta phải có cách chuyển đổi dữ liệu kiểu số sang kiểu chuỗi và ngược lại.

Ngoài ra, để người sử dụng có cảm giác thân thiện và dễ sử dụng chương trình, ta cần phải có các thao tác xử lý chuỗi như ghép nối chuỗi và số, cắt xén chuỗi... áp dụng khi yêu cầu nhập dữ liệu hoặc hiển thị thông tin.

7.2 Các hàm chuyển đổi dữ liệu từ chuỗi thành số và ngược lại

Hàm ATOF

Hàm **Atof** (*Ascii TO Floating point decimal*) chuyển đổi một chuỗi thành một số thực.

(**Atof** STRING)

Ví dụ:

(setq A "5.25" B "45" C "CAPSCREW" D "24x18 BORDER")

(atof "33.75")	trả về 33.75	
(atof "21")	trả về 21.0	
(atof "10.")	trả về 10.0	
(atof ".45")	trả về 0.45	
(atof A)	trả về 5.25	
(atof B)	trả về 45.0	
(atof "33.2E-09")	trả về 3.32e-08	Hàm **Atof** nhận biết được ký tự số mũ E
(atof "7-1/2")	trả về 7.0	Hàm **Atof** không nhận biết được phần số lẻ
(atot C)	trả về 0.0	Không tìm được số trong chuỗi.
(atof D)	trả về 24.0	Tìm được số ở phần đầu chuỗi.
(atof "")	trả về 0.0	
(atof "3.15" "62")	trả về lỗi	Hàm **Atof** chỉ chấp nhận 1 tham số.

Command: **(prompt "\nA x B is: ")(* (atof A) (atof B))**↵

A x B is: 236.25

(setq A (atof A))	Thay đổi giá trị chứa trong biến A. Giá trị mới chứa trong A là số thực 5.25

Hàm DISTOF

Hàm **Distof** (*DIStance TO Floating point decimal*) chuyển đổi một chuỗi thành một số thực, tương tự như hàm **Atof**.

(**Distof** STRING [MODE])

Tham số MODE xác định kiểu đơn vị đo chiều dài mà tham số STRING phải tuân theo. Nếu không có tham số MODE, hàm **Distof** sử dụng giá trị chứa trong biến hệ thống LUNITS.

Mode	Format
1	Scientific
2	Decimal
3	Engineering
4	Architectural
5	Arbitrary fractional units

Ví dụ:

(distof "8.2500E+01" 1)	trả về 82.5	MODE 1 = Scientific
(distof "82.50" 2)	trả về 82.5	MODE 2 = Decimal
(distof "6'-10.50\"" 3)	trả về 82.5	MODE 3 = Engineering
(distof "6'-10 1/2\"" 4)	trả về 82.5	MODE 4 = Architectural
(distof "82 1/2" 5)	trả về 82.5	MODE 5 = Fractional
(distof "8.2500E+01" 2)	trả về nil	Chuỗi không thích hợp MODE

Hàm ATOI

Hàm **Atoi** (*Asscii TO Integer*) chuyển đổi một chuỗi thành một số nguyên.

(**Atoi** STRING)

Ví dụ:

(setq A "5.25" B "45" C "CAPSCREW" D "24x18 BORDER")

(atoi "33.75")	trả về 33	
(atoi "21")	trả về 21	
(atoi A)	trả về 5	
(atoi "4.3e03")	trả về 4	Hàm **Atoi** không nhận biết ký tự số mũ E.
(atoi C)	trả về 0	Không tìm được số trong chuỗi.
(atoi D)	trả về 24	Tìm được số ở phần đầu chuỗi.
(fix (atof "4.3e03"))	trả về 4300	Dùng hàm **Atof** chuyển thành số thực, sau đó dùng hàm **Fix** để lấy phần số nguyên.

Hàm RTOS

Hàm **Rtos** *(Real TO String)* chuyển đổi một số thành một chuỗi.

(**Rtos** NUMBER [MODE [PRECISION]])

Các tham số MODE và PRECISION có ý nghĩa tương tự các biến hệ thống LUNITS (đơn vị đo chiều dài) và LUPREC (số chữ số thập phân). Nếu không có các tham số này, hàm **Rtos** sử dụng các giá trị LUNITS và LUPREC hiện hành. Hàm **Rtos** chuyển số NUMBER sang kiểu đơn vị đo thích hợp sau đó chuyển thành chuỗi.

Ví dụ:

(setq F 21.5 G 963 H "36.5 x 24.5")

(rtos F 1 2)	trả về "2.15E+01"	Kiểu scientific
(rtos G 2 1)	trả về "963.0"	Kiểu decimal
(rtos (atof H) 2 1)	trả về "36.5"	Hàm **Atof** trả về 36.5
(rtos F 6 3)	trả về "1'-9 1/2""	
(rtos F 3 10)	trả về "1'-9.5000000000"	10 số lẻ
(rtos 54.75)	trả về "54.7500"	LUNITS = 2, LUPREC = 4
(rtos 54.75)	trả về "54.75"	LUNITS = 2, LUPREC = 2
(rtos 54.75 4)	trả về "4-6 3/4""	LUPREC = 2
(rtos 54.75)	trả về "55"	LUNITS = 2, LUPREC = 0
(rtos 54.5)	trả về "55"	LUNITS = 2, LUPREC = 0
(rtos 54.49)	trả về "54"	LUNITS = 2, LUPREC = 0
(rtos 54)	trả về "54"	LUNITS = 2, LUPREC = 0

Hàm ITOA

Hàm **Itoa** *(Integer TO Ascii)* chuyển đổi một số nguyên thành một chuỗi.

(**Itoa** INTEGER)

Ví dụ:

(setq K 9.45 J 69 M "FLOORPLN"

(itoa 21)	trả về "21"	
(itoa K)	trả về lỗi	Chỉ chấp nhận tham số là số nguyên
(itoa J)	trả về "69"	
(itoa M)	trả về lỗi	Không nhận tham số là chuỗi
(itoa 16 2145)	trả về lỗi	Hàm **Itoa** chỉ có một tham số

Ví dụ:

Dùng giá trị của biến khi được yêu cầu nhập giá trị số trong lúc thực hiện các lệnh **AutoCAD**.

Command: **(setq VALUE1 2.0 VALUE2 "2.0" DWGSCALE 4.0 BLK "CAPSCREW")**↵
"CAPSCREW"
Command: **Insert** ↵
Block name (or ?): **!BLK** ↵
Insertion point: (Nhập một điểm)
X scale factor <1> / Corner /XYZ: **!VALUE1** ↵ ;trả về số thực 2.0
Y scale factor (default=X): ↵
Rotation angle <0>: ↵

Hoặc dùng:

X scale factor <1> / Corner /XYZ: !VALUE2 ↵ ;chuỗi "2.0" cũng
Y scale factor (default=X): ↵ ;được chấp nhận
Rotation angle <0>: ↵

Nhưng khi dùng biểu thức **AutoLISP**, ta phải đổi chuỗi thành số:
X scale factor <1> / Corner /XYZ: **(* VALUE1 DWGSCALE)** ↵

Hoặc dùng:
X scale factor <1> / Corner /XYZ: **(* (atof VALUE2) DWGSCALE)** ↵

Hàm ANGTOS

Hàm **Angtos** *(ANGle TO String)* chuyển đổi số đo một góc thành một chuỗi. Số đo góc ANGLE phải có đơn vị đo là radian.

(**Angtos** ANGLE [MODE [PRECISION]])

Các tham số MODE và PRECISION có ý nghĩa tương tự các biến hệ thống AUNITS (đơn vị đo góc) và AUPREC (số lẻ thập phân). Nếu không có các tham số này, hàm **Angtos** sử dụng các giá trị biến hệ thống AUNITS và AUPREC hiện hành.

Mode	Format
0	Degrees
1	Degrees/minutes/seconds
2	Grads
3	Radians
4	Surveyor's units

Ví dụ:

```
(setq ANG (cvunit 60 "DEGREES" "RADIANS") PT1 '(2 2) PT2 '(4 4)
(angtos 0.0)              trả về "0"          AUNITS = 0, AUPREC = 0
(angtos (/ pi 2))         trả về "90d0"       AUNITS = 1, AUPREC = 2
(angtos (* pi 1.5))       trả về "270.0000"   AUNITS = 0, AUPREC = 4
(angtos ANG 0 4)          trả về "60.0000"
(angtos ANG 1 4)          trả về "60d0'0""
(angtos ANG 2 4)          trả về "66.6667g"
(angtos ANG 3 4)          trả về "1.0472r"
(angtos ANG 4 4)          trả về "N 30d0'0" E"
(angtos (angle PT1 PT2) 0 2)   trả về "45.00"
```

Hàm ANGTOF

Hàm **Angtof** *(ANGle TO Floating point decimal)* chuyển đổi một chuỗi chứa số đo góc thành một số thực.

(**Angtof** STRING [MODE])

Tham số MODE xác định kiểu đơn vị đo góc mà tham số STRING phải tuân theo. Nếu không có tham số MODE, hàm **Angtof** sử dụng giá trị chứa trong biến hệ thống AUNITS.

Ví dụ:

```
(angtof "90" 0)          trả về 1.5708   MODE 0 = Degrees
(angtof "90d0'0\"" 1)    trả về 1.5708   MODE 1 = Degrees/min/sec
(angtof "100.0000g" 2)   trả về 1.5708   MODE 2 = Grads
(angtof "1.5708r" 3)     trả về 1.5708   MODE 3 = Radians
(angtof "N" 4)           trả về 1.5708   MODE 4 = Surveyor's units
```

Hàm ASCII

Hàm **Ascii** chuyển đổi ký tự đầu tiên trong một chuỗi thành mã ký tự ASCII tương ứng và trả về mã này.

(**Ascii** STRING)

Ví dụ:

```
(setq U "90.0" V 451 W "Stove Bolt" X "radians")
(ascii "A")              trả về 65
(ascii "A DRAWING FILE"  trả về 65
(ascii "a")              trả về 97
(ascii "9")              trả về 57
(ascii U)                trả về 57   Chỉ lấy ký tự đầu tiên của "90.0"
(ascii (itoa V))         trả về 52   Ký tự đầu tiên là 4
(ascii X)                trả về 114  Ký tự đầu tiên là r.
```

Hàm CHR

Hàm **Chr** *(ascii ChaRacter to string)* chuyển đổi mã ASCII thành ký tự tương ứng trong bảng mã ASCII. Các mã ASCII chuẩn có giá trị từ 32 đến 126.

(**Chr** INTEGER)

Ví dụ:

```
(setq AA 65 BB "42" CC 109.66)
(chr 65)          trả về "A"
(chr AA)          trả về "A"
(chr (+ AA 1))    trả về "B"
(chr (+ AA 2))    trả về "C"
(chr 42)          trả về "*"
(chr BB)          trả về lỗi     cần phải dùng hàm Atoi
```

(chr (atoi BB)) trả về "*"
(chr CC) trả về lỗi cần phải dùng hàm **Fix**
(chr (fix CC)) trả về "m"

Ví dụ:

Chương trình khai báo các biến kích thước, dùng (chr 34) để gán ký tự " cho biến DIMPOST.

```
, Tên file: DIMPREP.LSP
(defun C:DIMPREP (/ DSF)
  (setvar "CMDECHO" 0)
  (setq DSF (getreal "\nDimension Scale Factor: "))
  (command ".layer" "make" "DIM" ""          ;Tao lớp DIM
           ".dim" "dimunit" 5
            "dimdec" 4
            "dimscale" DSF
            "dimpost" (chr 34)        ;Gán cho biến dimpost ký tư "
  )
  (setvar "CMDECHO" 1)
)
; Kết thúc chương trình DIMPREP.LSP
```

Hàm READ

Hàm **Read** trả về phần tử đầu tiên chứa trong một chuỗi và chuyển đổi phần tử này về kiểu dữ liệu tương ứng.

(**Read** STRING)

Ví dụ:

(setq DD "DIM BLOCKS" EE "(x y) (z)" FF 109.66 GG "12 vars")

(read "the paper")	trả về the	SYMBOL
(read "1 2 3")	trả về 1	INTEGER
(read "1.5 2.5 3.5")	trả về 1.5	REAL
(read DD)	trả về DIM	SYMBOL
(read EE)	trả về (x y)	LIST
(car (read EE)	trả về x	SYMBOL
(read (rtos FF 2 2))	trả về 109.66	REAL
(read GG)	trả về 12	INTEGER
(read "A" "B")	trả về lỗi	Chỉ chấp nhận 1 tham số

Tóm tắt:

ATOF	(Ascii TO Floating point)	- Chuyển đổi chuỗi thành số thực
DISTOF	(DIStance TO Floating point)	- Chuyển đổi chuỗi thành số thực
ATOI	(Ascii TO Integer)	- Chuyển đổi chuỗi thành số nguyên
RTOS	(Real TO String)	- Chuyển đổi số nguyên thành chuỗi
ITOA	(Integer TO Ascii)	- Chuyển đổi số nguyên thành chuỗi
ANGTOS	(ANGle TO String)	- Chuyển giá trị góc đo thành chuỗi
ANGTOF	(ANGle TO Floating point)	- Chuyển đổi giá trị góc đo thành số thực
ASCII	(string to ASCII character)	- Chuyển đổi ký tự đầu tiên của chuỗi thành mã ASCII tương ứng
CHR	(ascii CHaRacter to string)	- Chuyển đổi mã ASCII thành ký tự tương ứng
READ	(READ string)	- Trả về phần tử đầu tiên của một chuỗi.

7.3 Các hàm hiển thị thông tin kiểu chuỗi

Ngoài hàm **Prompt**, **AutoLISP** còn có các hàm **Prin1**, **Princ**, **Print** dùng để hiện thông báo trên màn hình. Các hàm này sử dụng tương tự nhau, chỉ khác nhau cách hiển thị dữ liệu trên màn hình.

Hàm PRIN1

Hàm **Prin1** in dữ liệu chưa trong tham số EXPR lên màn hình hoặc in thành file và trả về kết quả là tham số này.

(**Prin1** [EXPR [FILE-DESC]])

Tham số EXPR có thể chứa các kiểu dữ liệu khác nhau. Tham số FILE-DESC là file dùng để chứa kết quả trả về của hàm.

Ví dụ:

(setq KK "BLOCKS" MM 29.75 NN '(A B C) PASTECLIP "36x24 BORDER")

(prin1 "33.75")	in lên màn hình "33.75"	trả về "33.75"
(prin1 KK)	in lên màn hình "BLOCKS"	trả về "BLOCKS"
(prin1 MM)	in lên màn hình 29.75	trả về "29.75"
(prin1 'a)	in lên màn hình A	trả về A
(prin1 nn)	in lên màn hình (A B C)	trả về (A B C)
(prin1 (car NN))	in lên màn hình A	trả về A
(prin1 (atoi PP))	in lên màn hình 36	trả về 36
(prin1 (chr 63))	in lên màn hình "?"	trả về "?"

(prin1 "\n")	in lên màn hình "\n"	trả về "\n"
(prin1 (chr 9))	in lên màn hình "\t"	trả về "\t"

Hàm PRINC

Hàm **Princ** in dữ liệu chứa trong tham số EXPR lên màn hình hoặc in thành file và trả về kết quả là tham số này, tương tự như hàm **Prin1**.

(**Princ** [EXPR [FILE-DESC]])

Ví dụ:

(princ "\n")	in một dòng mới	trả về "\n" (so sánh với hàm prin1)
(princ (chr 9))	in một khoảng TAB	trả về "\t" (so sánh với hàm prin1)
(princ (chr 75))	in ký tự K	trả về "K"
(princ 50)	in số 50	trả về 50

Hàm **(Princ)** không tham số sẽ không in dữ liệu lên màn hình và không trả về kết quả. Do đó hàm **(Princ)** không tham số thường được dùng làm biểu thức cuối cùng trong chương trình để chương trình kết thúc bình thường.

Hàm PRINT

Hàm **Print** tự động xuống dòng, sau đó in dữ liệu chứa trong tham số EXPR lên màn hình hoặc ra một file và trả về kết quả là tham số này, tương tự như hàm **Prin1**.

(**Print** [EXPR [FILE-DESC]])

Ví dụ:

Command:**(print "AutoLISP")(print "Programming") (princ)**↵
"AutoLISP" Hàm **Print** in dữ liệu trên dòng mới.
"Programming"
Command: **(print "AutoLISP")(princ "Programming")(princ)**↵
"AutoLISP" Programming Chuyển qua dòng mới, in *"AutoLISP"* với một khoảng trắng, sau đó in *Programming*
Command: **(print "The fillet radius is: ") (getvar "filletrad")**↵
"The fillet radius is: " 10.0

• So sánh các hàm PRIN1, PRINC và PRINT:

Command: **(prin1 "ABC")**↵
"ABC""ABC" ; In "ABC" và trả về "ABC"
Command: **(princ "ABC")**↵
ABC"ABC" ; In ABC và trả về "ABC"
Command: **(print "ABC")**↵
"ABC" "ABC" ;Sang dòng mới, in "ABC" và trả về "ABC"

Command: **(prin1 "\nABC")**↵
"\nABC""\nABC" ;Không nhận biết ký tự xuống dòng \n
Command: **(princ "\nABC")**↵
ABC"\nABC" ;Xem ký tự \n là ký tự xuống dòng
Command: **(print "\nABC")**↵
"\nABC""\nABC" ;Không nhận biết ký tự xuống dòng \n

Ví dụ:

Chương trình tính chi phí thi công khi biết đơn giá tính theo mét vuông sàn và diện tích thi công.

```
;Tên file:  DECAREA.LSP
(defun C:DA (/ RATE)
  (setvar "cmdecho" 0)
  (print "Setting units to decimal")     ;Hiện thông báo bằng hàm Print
  (Command ".lunits" 2)                 ;Đơn vị đo chiều dài ở hệ thập phân
  (setq RATE (getreal "\nRate per square meter: "))   ;Nhập đơn giá m2
  (prompt "\nSelect area for calculation: ") ;Hiện thông báo bằng Prompt
  (command ".area" "entity" pause);Chọn đa tuyến để tính diện tích thicông
  (princ "\nThe estimated cost is : ")        ;Hiện thông báo bằng Princ
  (prin1 (* (getvar "area") RATE))            ;Hiện kết quả bằng Prin1
  (setvar "cmdecho" 1)
  (princ)
)
;Kết thúc chương trình DECAREA.LSP
```

Thực hiện chương trình:

Command: **da** ↵
"Setting units to decimal"
Rate per square meter: **100** ↵

Select area for calculation: (Chọn đa tuyến)
The estimated cost is : 1329.89

7.4 Các hàm xử lý chuỗi

Việc giao tiếp giữa chương trình và người sử dụng thường được thực hiện thông qua chuỗi. Ta cần phải thường xuyên thay đổi dạng chuỗi, lấy ra một phần của chuỗi, hoặc kết nối nhiều chuỗi ngắn thành chuỗi dài hơn... để việc giao tiếp được dễ dàng hơn.

Hàm STRCASE

Hàm **Strcase** *(STRing CASE)* biến đổi các ký tự trong chuỗi STRING thành các chữ hoa hoặc chữ thường.

(**Strcase** STRING [WHICH])

Nếu không có tham số WHICH hoặc có nhưng bằng nil, hàm sẽ trả về chuỗi chứa trong tham số STRING với tất cả các ký tự được chuyển sang dạng chữ hoa. Ngược lại, nếu tham số WHICH khác nil, tất cả các ký tự được chuyển sang dạng chữ thường.

Ví dụ:

```
(setq AA "Angle" TT 17.4 LL "24x12 BORDER")
(strcase "42.12")          trả về "42.12"
(strcase AA)               trả về "ANGLE"
(strcase AA T)             trả về "angle"
(strcase TT)               trả về lỗi (tham số phải có kiểu chuỗi)
(strcase (rtos TT))        trả về "17.4000"
(strcase LL nil)           trả về "24X12 BORDER"
(strcase LL T)             trả về  "24x12 border"
```

Hàm STRCAT

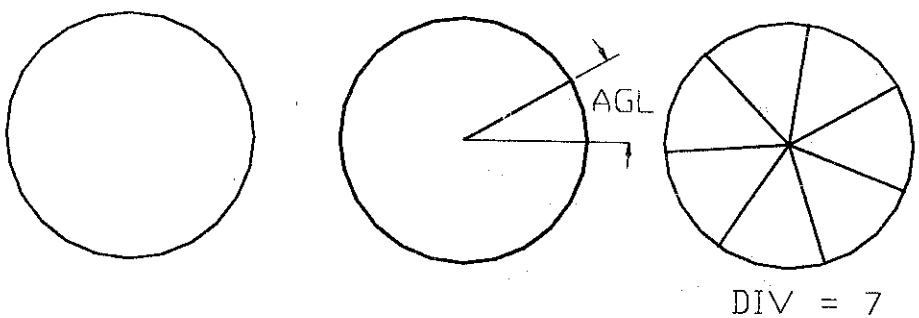
Hàm **Strcat** *(STRing conCATenate)* dùng để kết nối nhiều chuỗi thành một chuỗi duy nhất.

(**Strcat** STRING1 [STRING2] ...)

Ví dụ:

```
(setq AA "Angle" RR 14.2 BB '("A " "B" "C") LL "24x12 BORDER")
(strcat "5" AA)                trả về "5Angle"
(strcat "5 " AA)               trả về "5 Angle"
(strcat "5 " (strcase AA))     trả về "5 ANGLE"
(strcat (itoa (fix RR)) " " AA)  trả về "14 Angle"
(strcat (car BB) (cadr BB))    trả về "AB"
(strcat "The choice is: " LL)  trả về "The choice is: 24x12 BORDER"
```

Ví dụ:




```
;Tên file: CV.LSP
;Mục đích: Vẽ các đường bán kính của đường tròn, chia đường tròn thành
;các phần đều nhau. Người sử dụng chon đường tròn, chon góc xác đinh vi
;trí vẽ bán kính thứ nhất. Sau đó nhập số phần chia bằng nhau, chương
;trình sẽ thực hiện lệnh Array quanh tâm đường tròn.
;
(defun C:CV (/ PT 1 CPT DIS AGL DIV)
  (setvar "CMDECHO" 0)
  (setq PT1 (osnap (getpoint "\nSelect circle: ") "NEA") ;Chọn một điểm
                                                         ;trên đường tròn
        CPT (osnap PT1 "CEN")                            ; Tâm đường tròn
        DIS (distance PT1 CPT)                           ;Bán kính đường tròn
        AGL (getstring "\nAngle for radial line: ");Góc vẽ bán kính đầu tiên
        DIV (getint "\nNumber of equal divisions: ");Số phần chia bằngnhau
  ); Đóng setq
  (command ".line" CPT (strcat "@" (rtos DIS 2 8) "<" AGL) ""
           ".array" "L" "" "p" CPT DIV 360 "y"  ;Vẽ các bán kính còn lai
  )
  (setvar "CMDECHO" 1)
  (princ)
)
;Kết thúc chương trình CV'LSP
```

Hàm STRLEN

Hàm **Strlen** *(STRing LENgth)* trả về chiều dài của một chuỗi (số ký tự của chuỗi).

(**Strlen** [STRING] ...)

Nếu có nhiều tham số STRING, hàm trả về tổng chiều dài tất cả các chuỗi.

Ví dụ:

```
(setq LL "Angle" AA '("A" "B" "C"))
(strlen "BB")                    trả về 2
(strlen "BB ")                   trả về 3
(strlen LL)                      trả về 5
(strlen "Distance" LL)           trả về 13
(strlen (car AA) (cadr AA))      trả về 2
```

Hàm SUBSTR

Hàm **Substr** *(SUBSTRing)* trả về một chuỗi con trong tham số STRING.

(**Substr** STRING START [LENGTH])

Tham số START chứa vị trí bắt đầu chuỗi con trong chuỗi STRING. Tham số LENGTH là chiều dài của chuỗi con được lấy ra. Nếu không có tham số LENGTH, chuỗi con sẽ được lấy ra bắt đầu từ vị trí START cho đến hết chuỗi STRING.

Ví dụ:

```
(setq AA "AutoLISP Programming" DD '("First pt" "Second pt") LL "4565")
(substr 56 2)                    trả về lỗi sai kiểu dữ liệu
(substr "Yes")                   trả về lỗi chưa đủ tham số
(substr AA 1 8)                  trả về "AutoLISP"
(substr AA 10)                   trả về "Programming"
(substr (car DD) 1 5)            trả về "First"
(substr (cadr DD) 8)             trả về "pt"
(substr "The day before" 9 3)    trả về "bef"
```

Ví dụ:

Hàm tự tạo DATE hiện ngày hiện tại trên màn hình theo dạng:
"Today's date is: 25-11-2000"

```
;Tên file: DATE.LSP
;Mục đích: Hiện ngày hiện hành lên màn hình
;
(defun C:DATE (/ TODAY YEAR MONTH DAY MDY INSPT HGT)
  (setvar "cmdecho" 0)
  (setq TODAY (rtos (getvar "CDATE") 2 0)   ;Đổi ngày thành chuỗi
                                            ; ví dụ "20001125"
        YEAR   (substr TODAY 1 4)           ;Tách ra năm "2000"
        MONTH (substr TODAY 5 2)            ;Tháng "11"
        DAY    (substr TODAY 8 2)           ;Ngày "25"
        MDY (strcat "Today's date: " DAY "-" MONTH "-" YEAR);Tao
                                            ;chuỗi ngày hiện hành
        INSPT (getpoint "\nStart point for text: ")     ;Điểm chèn dòng chữ
        HGT    (getreal "\n Text height: ")             ; Chiều cao chữ
  )
  (command ".text" INSPT HGT 0.0 MDY)       ;Viết dòng chữ lên màn hình
  (princ)
)
; Kết thúc chương trình
```

Hàm ACAD_STRLSORT

Hàm **Acad_Strlsort** *(AutoCAD STRing List SORT)* nhận tham số là danh sách gồm các phần tử là chuỗi và trả về danh sách đã được sắp xếp theo thứ tự ABC.

(**Acad_Strlsort** LIST)

Ví dụ:

```
(setq LST1 '("a" "w" "A" "k") LST2 '("3/4" "Screw" "5/8" "Bolt"))
(acad_strlsort '("c" "b" "a"))          trả về ("a" "b" "c")
(acad_strlsort '("b" "a") '("d" "c"))   trả về ("a" "b")
(acad_strlsort '("abcd" "abc"))         trả về ("abc" "abcd")
(acad_strlsort '(list 4 3 2 1))         không thể sắp xếp được
(acad_strlsort LST1)                    trả về ("A" "a" "k" "w")
(acad_strlsort LST2)                    trả về ("3/4" "5/8" "Bolt" "Screw")
(acad_strlsort '("BOLt" "BOLT"))        trả về ("BOLT" "BOLt")
```

Tóm tắt:

1. Các hàm **Prin1**, **Princ**, **Print** in thông tin lên màn hình. Các hàm này sử dụng thuận tiện hơn hàm **Prompt**.
2. Hàm **Strcase** chuyển tất cả các ký tự một chuỗi thành dạng chữ hoa hoặc chữ thường.
3. Hàm **Strcat** kết nối nhiều chuỗi thành một chuỗi.
4. Hàm **Strlen** trả về chiều dài một chuỗi.
5. Hàm **Substr** trả về một chuỗi con trong một chuỗi, bắt đầu từ vị trí START và có chiều dài là LENGTH.
6. Hàm **Acad_Strlsort** sắp xếp thứ tự một danh sách gồm các phần tử là chuỗi theo thứ tự ABC.

7.5 Các ví dụ mẫu

Ví dụ 1:

Lập chương trình có sử dụng hàm **Rtos, Strcat** chuyển đổi foot, inch thành mét (m) và milimét (mm). (Tham khảo thêm hàm **If** chương 8)

```
;Tên file: FEET-M.LSP
;Chuyển đổi chuyển đổi foot, inch thành mét (m) và milimét (mm)
( princ " \n    " )
( princ " \n  CONVERTS FEET & INCHES TO METERS . TO RUN ENTER
FM or FEET-M ." )
;-------------------------------------------------------------------------;
( defun FEET-M  ( / FE_ET    IN_CH    IN_CH_T    ME_TER  mm )
  ( setq  FE_ET ( getreal  " \n ENTER FEET > " ))
  ( if    ( null FE_ET )( setq  FE_ET  0 ))
  ( setq  IN_CH ( getreal  " \n ENTER INCHES > " ))
  ( if    ( null IN_CH )( setq  IN_CH  0 ))
  ( setq  IN_CH_T ( + ( * FE_ET  12 ) IN_CH ))
  ( setq  ME_TER  ( *  IN_CH_T   0.0254 ))
  ( setq  mm      ( *  IN_CH_T   25.4   ))
  ( princ " \n   ***  PRESS ENTER TO REPEAT  *** " )
  ( princ ( strcat   " \n "
     ( rtos     FE_ET     2  3 )   " feet "
     ( rtos     IN_CH     2  3 )   " inch = "
     ( rtos     IN_CH_T   2  3 )   " inches = "
     ( rtos     ME_TER    2  3 )   " meters = "
     ( rtos     mm        2  3 )   " mm  "
  )      )
  ( princ )
)
( defun C:FM      ( ) ( FEET-M ))
( defun C:FEET-M  ( ) (.FEET-M ))
( princ )
( C:FEET-M ) ;Kết thúc file FEET-M.LSP
```


Ví dụ 2:

Sắp xếp danh sách họ tên người.

```
;Tên file: A-TEAM.LSP
;Mục đích: Nhập vào tên 4 người tạo thành một danh sách. Sau đó xếp thứ
;tự danh sách và viết lên bản vẽ hiện hành.
;
(defun C:ATEAM (/ ST M1 M2 M3 M4 LST)
  (setvar "cmdecho" 0)
  (setq ST "\nEnter Team Members"
M1 (getstring T (strcat ST "1: "))          ;Tên thứ nhất
      M2 (getstring T (strcat ST "2: "))    ;Tên thứ hai
M3 (getstring T (strcat ST "3: "))          ;Tên thứ ba
M4 (getstring T (strcat ST "4: "))          ;Tên thứ tư
LST (acad_strlsort (list M1 M2 M3 M4))      ;Sắp xếp danh sách
  )
  (princ "\nPick Text start point: ")
      (command ".text" pause)
  (princ "\nText heigh: ")
      (command pause 0.0 (car LST))
  (setq LST (cdr LST))
      (command ".text" "" (car LST))
  (setq LST (cdr LST))
      (command ".text" "" (car LST))
  (setq LST (cdr LST))
      (command ".text" "" (car LST))
  (setvar "cmdecho" 1)
  (princ)
)
;Kết thúc chương trình A_TEAM.LSP
```

7.6 Bài tập

1. Tính giá trị các biểu thức sau:

(setq A "7th Chapter – Conversions and Strings")

a. (atof A)
b. (atoi A)
c. (rtos (atof A) 2 1)
d. (itoa (atoi A))
e. (angtos (/ pi 24) 0)
f. (ascii "A")
g. (chr 40)
h. (read A)

2. Viết chương trình VARSOUT.LSP lấy giá trị các biến hệ thống sau: ACADVER, DWGPREFIX, DWGNAME, LOGINNAME, MENUNAME, PLATFORM. Sau đó dùng các hàm **Princ** và **Strcat** để hiển giá trị các biến này lên màn hình theo dạng:

(strcat "\nThe value for "VAR" is: " X)

trong đó, VAR là tên biến, X là giá trị của biến.

3. Viết chương trình FR.LSP hiển thị giá trị biến hệ thống FILLETRAD lên màn hình. Sau đó thực hiện lệnh **Fillet**. Hãy dùng các hàm **Rtos**, **Princ** và **Strcat**.

4. Viết chương trình CIR-LABL.LSP thực hiện các công việc sau:

a. Dùng hàm **Getreal** gán cho biến RAD một số thực lớn hơn 25 và nhỏ hơn 40.
b. Vẽ đường tròn có bán kính bằng giá trị trên. Dùng hàm **Getpoint** để nhập tâm đường tròn.
c. Viết dòng chữ thể hiện bán kính đường tròn "Circle radius = ...". Dòng chữ này có điểm Middle trùng với tâm đường tròn.

5. Hãy nghiên cứu chương trình sau. Sau đó, dùng các hàm **Rtos**, **Print** để bổ sung vào phần Material Report, hiển thị thông tin về vật liệu lên màn hình.

```
;Tên file: STEEL.LSP
;Purpose: Basic flat paattern computations program
;Restrictions: Selected feature must be valid for AREA command in entity
;mode
;                (pline or circle)
;Program inputs: 1. Select major feature for overall area, etc. pline
;                2. Select hole to be subtracted (circle)
;                3. Results with reveal overall area data and resultant area
;and weight data.
```



```
(defun C:STEEL (/ PT1 TPL PT2 VOID TH WT TOTA TOTW)
  (setvar "cmdecho" 0)
;template area
  (prompt "\nSelect template for area/weight calculation: ")
  (setvar "osmode" 512)
  (setq PT1 (getpoint))
   (command ".area" "e" PT1)
  (setq TPL (getvar "area"))
;Voided area
  (prompt "\nSelect hole in template to be constructed: ")
  (setq PT2 (getpoint))
   (command ".area" "e" PT2)
  (setq VOID (getvar "area"))
;Computations
  (setq TH   (getreal "\nThickness of material :")
        WT   (getreal "\nWeight of material/cubic in. : ")
        TOTA (- TPL VOID)   ;composite Area
        TOTW (* TOTA TH WT)       ;composite weight
  )
;Material Report
;
  (setvar "osmode" 0)
  (setvar "cmdecho" 1)
)
;Kết thúc chương trình
```

7.7 Lời giải

1.

a. 7.0
b. 7
c. "7.0"
d. 7
e. "8"
f. 65
g. "("
h. 7th

2.

```
;Tên file: VARSOUT.LSP
(defun C:VARSOUT (/ VAR1 VAR2 VAR3 VAR4 VAR5 VAR6)
  (setq VAR1 (getvar "Acadver")
        VAR2 (getvar "Dwgprefix")
        VAR3 (getvar "Dwgname")
        VAR4 (getvar "Loginname")
        VAR5 (getvar "Menuname")
        VAR6 (getvar "Platform")
```

```
)
  (princ (strcat "\nThe value for Acadver is: " VAR1))
  (princ (strcat "\nThe value for Dwgprefix is: " VAR2))
  (princ (strcat "\nThe value for Dwgname is: " VAR3))
  (princ (strcat "\nThe value for Loginname is: " VAR4))
  (princ (strcat "\nThe value for Menuname is: " VAR5))
  (princ (strcat "\nThe value for Platform is: " VAR6))
(princ)
)
```

3.

```
;Tên file: FRADIUS.LSP
;
(princ (strcat "Current fillet radius is: " (rtos (getvar "FILLETRAD"))))
(command ".fillet")
```

4.

```
;Tên file: CIR-LABL.LSP
(defun C:CIR-LABL (/ RAD CEN TXT)
  (setq RAD (getreal "\nEnter circle radius (25<R<40): ")
        CEN (getpoint "\nSpecify center point: ")
        TXT (strcat "Circle radius = " (rtos RAD))
  )
  (command ".circle" CEN RAD)
  (command ".text" "M" CEN 3.0 0.0 TXT "")
)
```

5.

```
;Material Report
  (print (strcat "Total area is: " (rtos TOTA)))
  (print (strcat "Total weight is: " (rtos TOTW)))
```

Chương 8

CÁC BIỂU THỨC ĐIỀU KIỆN

Nội dung chương

1. Các biểu thức điều kiện:
   - Các hàm so sánh: **=, Equal, Eq, /=, <, <=, >, >=**
   - Các hàm kiểm tra dữ liệu: **Atom, Listp, Numberp, Minusp, Zerop, Boundp, Null, Type.**
2. Rẽ nhánh chương trình bằng hàm **If** và **Progn.**
3. Các hàm logic: **And, Or, Not.**
4. Rẽ nhánh chương trình bằng hàm **Cond.**

Các chương trình chúng ta đã viết trong các chương trước thực hiện tuần tự các lệnh từ đầu đến cuối. Mỗi lần thực hiện, người sử dụng nhập các giá trị khác nhau cho tham số, chương trình sẽ tính toán và trả về các kết quả tùy thuộc vào tham số. Nhưng trình tự thực hiện các lệnh vẫn giống nhau.

Tuy nhiên, để giải quyết hiệu quả các bài toán phức tạp, chương trình cần có chức năng rẽ nhánh. Tùy theo các tham số nhập, chương trình sẽ kiểm tra một số điều kiện để quyết định đi theo nhánh nào để thực hiện các lệnh. Một số lệnh được thực hiện, một số lệnh không được thực hiện.

8.1 Biểu thức điều kiện

Biểu thức điều kiện (*test expression*) chứa các điều kiện kiểm tra quan hệ giữa hai phần tử. Biểu thức này trả về giá trị *T* hoặc giá trị khác rỗng (tương ứng với "True") nếu điều kiện được thỏa mãn, hoặc trả về giá trị *nil* (tương ứng với "False") nếu điều kiện không được thỏa mãn.

8.1.1 Các hàm so sánh

Các hàm so sánh gồm có:

=	<
EQ	>
EQUAL	<=
/=	>=

Hàm =

Hàm = trả về giá trị T nếu tất cả các phần tử bằng nhau.

(= ATOM ATOM ...)

Hàm này chỉ chấp nhận tham số kiểu số hoặc kiểu chuỗi. Khi so sánh các chuỗi, cần phân biệt dạng chữ hoa chữ thường.

Ví dụ:

(setq A "OK" B 4 X 12 Y 12)

(= 20 20)	trả về T	
(= 10 20)	trả về nil	
(= 10 10 20 10)	trả về nil	
(= 20 20 20 20)	trả về T	
(= A "OK")	trả về T	
(= A "ok")	trả về nil	phân biệt dạng chữ hoa chữ thường
(= 4 B)	trả về T	
(= X Y 12)	trả về T	
(= B X 4)	trả về nil	
(= "OK" 4)	trả về nil	
(= '(0 0) '(0 0))	trả về nil	không chấp nhận kiểu danh sách

Hàm EQUAL

Hàm **Equal** đinh giá trị các biểu thức va kiểm tra các giá trị nay có bằng nhau không. Hàm này chấp nhận mọi kiểu dữ liệu.

(**Equal** EXPR1 EXPR2 [FUZZ])

Ví dụ:

(setq A 10.0 B "YOU")

(equal '(0 0) '(0 0))	trả về T	chấp nhận kiểu danh sách
(equal B "YOU")	trả về T	
(equal 10.1 A)	trả về nil	
(equal 10.0 A)	trả về T	

Tham số FUZZ cung cấp sai số trong phép so sánh bằng nhau và chỉ áp dụng với kiểu dữ liệu số. Các giá trị được xem bằng nhau nếu sai số giữa chúng nhỏ hơn giá trị FUZZ.

Ví dụ:

(setq X 1.1121 Y 1.11215 Z 12.9 PT1 '(0.124 0.124) PT2 '(0.124 0.125))

(equal X Y)	trả về nil	
(equal X Y 0.0001)	trả về T	Vì Y – X = 0.00005 nhỏ hơn FUZZ=0.0001
(equal Z 12.8)	trả về nil	
(equal Z 12.8 0.1)	trả về T	Vì Z – 12.8 = 0.1 bằng FUZZ = 0.1
(equal Z 20.0 0.1)	trả về T	Vì 20.0 – Z = 0.1 bằng FUZZ = 0.1
(equal PT1 PT2)	trả về nil	Các phần tử tương ứng trong danh sách được so sánh với nhau.
(equal PT1 PT2 0.001)	trả về nil	Khi so sánh các danh sách, sai số phải *nhỏ hơn* FUZZ.
(equal PT2 '(0.124 0.1245) 0.001)	trả về T	
(equal '(0.1 0.2) '(0.1 0.2 0.3))	trả về nil vì số lượng các phần tử khác nhau	

Hàm EQ

Hàm **Eq** so sánh sự trùng nhau giữa hai danh sách.

(**Eq** EXPR1 EXPR2)

Nếu EXPR1 và EXPR2 trùng nhau, hàm trả về T (True).

Ví dụ:

(setq PT1 '(20 20 0))

(setq PT2 '(20 20 0))

(eq PT1 PT2)	trả về nil	Hai danh sách PT1 và PT2 bằng nhau nhưng vẫn là hai danh sách khác nhau
(setq PT3 PT1)		
(eq PT1 PT3)	trả về T	Hàm **Setq** đinh giá trị biến PT1, sau

đó gán kết quả cho PT3. Do đó PT1 và PT3 trùng nhau, nên hàm **Eq** trả về T.

Hàm /=

Hàm /= (không bằng) trả về T nếu từng phần tử khac với phần tử đứng bên phải nó, ngược lại, hàm trả về nil.

(/= ATOM ATOM ...)

Ví dụ:

(setq A 2 B 3 C "YES")

(/= 1 3)	trả về T
(/= A 2)	trả về nil
(/= C "YES")	trả về nil
(/= "YES" "NO")	trả về T
(/= A B)	trả về T
(/= 1 2 1)	trả về T
(/= 1 2 2)	trả về nil
(/= 1 2 3 1)	trả về T

Hàm nhỏ hơn <

Hàm < trả về T nếu mỗi phần tử nhỏ hơn phần tử đứng bên phải nó.

(< ATOM ATOM ...)

Hàm nhỏ hơn hoặc bằng <=

Hàm <= trả về T nếu mỗi phần tử nhỏ hơn hoặc bằng phần tử đứng bên phải nó.

(<= ATOM ATOM ...)

Hàm lớn hơn >

Hàm > trả về T nếu mỗi phần tử lớn hơn phần tử đứng bên phải no.

(> ATOM ATOM ...)

Hàm lớn hơn hoặc bằng >=

Hàm >= trả về T nếu mỗi phần tử lớn hơn hoặc bằng phần tử đứng bên phải nó.

(>= ATOM ATOM ...)

Ví dụ:

(setq X 4 Z –32 M 3 N 2)

(< X 16)	trả về T	
(< N M X 4)	trả về nil	vì X = 4
(<= 1 2 3 X 4)	trả về T	
(<= 9 11)	trả về T	
(> 14 8 M 1)	trả về T	
(> -1 N)	trả về nil	
(>= 4 4 2)	trả về T	
(>= 12 Z)	trả về T	

Các chuỗi được so sánh với nhau theo giá trị các mã ASCII tương ứng.

Ví dụ:

(> "b" "a")	trả về T
(> "abcd" "abc")	trả về T
(< "xyz" "a")	trả về nil
(< "ABC" "ABD")	trả về T
(> "ABC" "aBC")	trả về nil
(<= "xy" "xy")	trả về T
(> "xz" "xyz")	trả về T

Tóm tắt:

1. Chương trình nên có chức năng rẽ nhánh để giải quyết các bài toán phức tạp.
2. Các biểu thức điều kiện trả về giá trị T hoặc nil (True hoặc False) giúp chương trình quyết định rẽ theo nhánh nào để thực hiện các lệnh tương ứng.
3. Các hàm kiểm tra sự bằng nhau như **Equal, /=, >=** ...

8.1.2 Các hàm kiểm tra kiểu dữ liệu

Khi cần thiết, ta nên kiểm tra kiểu dữ liệu cung cấp cho tham số trước khi thực hiện chương trình. Các hàm kiểm tra kiểu dữ liệu gồm có:

Atom	**Zerop**
Listp	**Boundp**
Numberp	**Null**
Minusp	**Type**

Hàm ATOM

Hàm **Atom** kiểm tra dữ liệu có phải là *nguyên tố (atom)* hay không. Mọi dữ liệu không phải là kiểu danh sách đều được xem là *nguyên tố.*

(**Atom** ITEM)

Ví dụ:

(setq B nil S "Today" M 1 LS '(a b))

(atom M)	trả về T	
(atom S)	trả về T	
(atom LS)	trả về nil	
(atom B)	trả về T	giá trị rỗng
(atom (car (list 'A 'B)))	trả về T	

Hàm LISTP

Hàm **Listp** kiểm tra dữ liệu có phải là kiểu danh sách hay không.

(**Listp** ITEM)

Ví dụ:

(setq B nil S "Today" M 1 LS '(a b))

(listp M)	trả về nil	
(listp S)	trả về nil	
(listp LS)	trả về T	
(listp '(1 1 1))	trả về T	
(listp B)	trả về T	danh sách rỗng

Hàm NUMBERP

Hàm **Numberp** kiểm tra dữ liệu có phải là kiểu số hay không.

(**Numberp** ITEM)

Ví dụ:

(setq M 14.0 B "YES")

(numberp 'M)	trả về nil	biến M không được định giá trị
(numberp M)	trả về T	
(numberp B)	trả về nil	
(numberp 1.122)	trả về T	
(numberp (cadr '(1 2 3)))	trả về T	

Hàm MINUSP

Hàm **Minusp** kiểm tra dữ liệu có phải là số âm hay không.

(**Minusp** ITEM)

Ví dụ:

(minusp 1)	trả về nil
(minusp –1)	trả về T
(minusp (+ 5 2 –8))	trả về T

Hàm ZEROP

Hàm **Zerop** kiểm tra dữ liệu có phải là số không hay không.

(**Zerop** ITEM)

Ví dụ:

(zerop 0)	trả về T
(zerop 0.0)	trả về T
(zerop 1.90)	trả về nil
(zerop (- 14 14))	trả về T

Hàm BOUNDP

Hàm **Boundp** sẽ trả về T nếu tham số là nguyên tố và nó được gắn với một giá trị. Ngược lại, sẽ trả về nil.

(**Boundp** ATOM)

Ví dụ:

(setq A 15 B nil C 'B)

(boundp 1)	trả về nil	vì 1 không phải là biến
(boundp A)	trả về nil	vì A được đinh giá trị là 15, không phải là một biến
(boundp 'A)	trả về T	biến A được gắn với giá trị 15
(boundp 'B)	trả về nil	biến B gắn với nil
(boundp 'C)	trả về T	biến C được gắn với biến B
(boundp B)	trả về nil	biến B được gắn với giá trị của B là nil

Khi dùng hàm **Boundp**, ta nên đặt dấu ' trước các biến, để các biến không bị đinh giá trị. Khi đó tham số cho hàm **Boundp** là một biến chứ không phải là một giá trị.

Hàm NULL

Hàm **Null** kiểm tra một biến hoặc danh sách có rỗng hay không.

(**Null** ITEM)

Ví dụ:

```
(setq A '(1 2 3)  B nil)
(null A)                trả về nil
(null B)                trả về T
(null '())              trả về T
```

Hàm TYPE

Hàm **Type** trả về kiểu dữ liệu kết nối với tham số ITEM.

(**Type** ITEM)

Giá trị trả về	Ý nghĩa
REAL	Số thực
INT	Số nguyên
STR	Chuỗi
SYM	Tên biến
LIST	Danh sách
FILE	File Descriptors
PICKSET	Tập hợp các đối tượng chọn
ENAME	Tên đối tượng AutoCAD
SUBR	Internal AutoLISP Function
EXSUBR	External (ADS) Function
PAGETB	Function Paging Table

Ví dụ:

```
(setq a 10.0  B 30 C "Total"  D'(0 0 0)  E nil)
(type A)                trả về  REAL
(type 'A)               trả về  SYM
(type B)                trả về  INT
(type C)                trả về  STR
(type D)                trả về  LIST
(type E)                trả về nil
(type '(0 0))           trả về LIST
(type (type '(0 0)))    trả về SYM
```

Giá trị trả về của hàm **Type** có kiểu là tên biến SYM, chứ không phải kiểu chuỗi.

Ví dụ: Chương trình kiểm tra kiểu dữ liệu nhập vào

```
(setq A 1.0)
(if
  (numberp A)                    ;kiểm tra A có phải là số không
  (if
    (= 'INT (type A))            ;nếu phải, kiểm tra có phải là số
                                 ;nguyên không. Dùng 'INT để không
                                 ;định giá trị của INT
    (prompt "\nIt is an integer!")      ;Là số nguyên
    (prompt "\nIt is a real number!")   ;Là số thực
  )
  (prompt "\nIt is not a number!")
)
```

Ví dụ:

Một số chương trình không kiểm soát các giá trị nhập nên có thể xảy ra trường hợp giá trị trả về có kiểu dữ liệu không thích hợp. Ví dụ khi dùng hàm **Getint**, nếu ta không nhập vào số nguyên mà chỉ nhấn ENTER, thì giá trị trả về là nil. Nếu chương trình tiếp tục sử dụng giá trị này thì sẽ gây ra lỗi. Ta có thể dùng hàm **Type** kiểm tra kiểu dữ liệu để quyết định chương trình tiếp tục hay không.

```
(setq A (getint "\nEnter an integer: "))
(if
  (= nil (TYPE A))
  (prompt "Program cancel")
  (prompt "Program continue ...")
)
```

8.2 Rẽ nhánh chương trình

Để rẽ nhánh chương trình ta sử dụng hàm **If** và **Progn**.

Hàm IF

Hàm **If** có các dạng như sau:

(**If** TESTexpr THENexpr) – Dạng IF THEN. Nếu biểu thức điều kiện TESTexpr đúng (trả về T) thì biểu thức THENexpr sẽ được thực hiện. Ngược lại biểu thức THENexpr sẽ không được thực hiện.

(**If** TESTexpr THENexpr ELSEexpr) – Dạng IF THEN ELSE. Nếu biểu thức điều kiện TESTexpr đúng (trả về T) thì biểu thức THENexpr sẽ được thực hiện. Ngược lại biểu thức ELSEexpr sẽ được thực hiện.

Ví dụ:

Chương trình cho phép người sử dụng lựa chọn vẽ hoặc không vẽ đường tròn.

```
;Tên file: CIRCLE.LSP
;Nếu biểu thức = trả về T, chương trình sẽ vẽ đường tròn.
, Ngược lại, chương trình sẽ không vẽ.
(defun C:YC ()
  (initget 1 "Yes No")
  (if (= "Yes" (getkword "\nDraw a CIRCLE? <Y/N>: "))     ;TESTexpr
    (command ".Circle")                                   ;THENexpr
  )
  (princ)
)
;Kết thúc chương trình
```

Ví dụ:

Chương trình cho phép người sử dụng lựa chọn vẽ đường tròn hoặc vẽ đường thẳng.

```
, Tên file: CIRCLE-LINE.LSP
, Nếu biểu thức = trả về T, chương trình sẽ vẽ đường tròn.
; Ngược lại, chương trình sẽ vẽ đường thẳng.
(defun C:YCNL ()
  (initget 1 "Circle Line")
  (if (= "Circle" (getkword "\nDraw a CIRCLE or LINE? <C/L>: ")) ;TEST
    (command ".Circle")                                         ;THEN
    (command ".Line")                                           ;ELSE
  )
  (princ)
) ;Kết thúc chương trình
```

Ví dụ: Chương trình kiểm tra tham số nhập vào có phải là số hay không.

```
;Tên file: NUMTEST.LSP
(defun NUMTEST (NUM / INP)              ;Tham số NUM
  (if
      (numberp NUM)                     ;TEST
      (princ (setq INP (- NUM 32)))     ;THEN
      (princ "\nNum is not a number.")  ;ELSE
  )
  (princ)
);Kết thúc chương trình
```

Thực hiện chương trình:

Command: **(numtest "qq")** ↵

Num is not a number.

Hàm PROGN

Hàm **If** chỉ chấp nhận một biểu thức THENexpr và một biểu thức ELSEexpr. Nhưng trong thực tế, chương trình cần phải thực hiện nhiều biểu thức liên tiếp để có kết quả mong muốn. Ta có thể nhóm nhiều biểu thức thành một biểu thức duy nhất để sử dụng trong hàm **If** bằng hàm **Progn**.

(**Progn** EXPRESSION ...)

Ví dụ:

Nhóm ba biểu thức **Setq** thành một biểu thức duy nhất.

```
(progn
  (setq A 14)
  (setq B 22)
  (setq C (- B A))
)
```

Ví dụ: Chương trình vẽ đường tròn, đường thẳng.

```
;Tên file: LORC.LSP
(defun C:LORC (/ ANS PT1 PT2 RAD)
  (setvar "CMDECHO" 0)
  (initget 1 "L C")
  (setq ANS (getkword "\nDraw a Line or Circle?<L/C>: "))
  (if (= ANS "L")
      (progn                            ;Bắt đầu THEN progn
       (setq PT1 (getpoint "\nStart point: ")
             PT2 (getpoint PT1 "\nEnd point: ")
       )
       (command ".line" PT1 PT2 "")
      )                                 ;Đóng THEN progn
      (progn                            ;Bắt đầu  ELSE progn
       (setq PT1 (getpoint "\nCenter of Circle: ")
             RAD (getdist PT1 "\Radius: ")
       )
       (command ".Circle" PT1 RAD)
      )                                 ;Đóng ELSE progn
```

```
    )                                   ;Đóng IF
   (princ)
);Kết thúc chương trình
```

Ví dụ: Thay đổi chiều rộng cho các pline khi chọn từng đối tượng (CLW.LSP) sử dụng đồng thời hàm **If** và **Progn.**

```
;Tên file: CLW.LSP
(defun C:CLW ()
  (setvar "cmdecho" 0)
  (setq LW (getdist "\nEnter new width:  ")); Nhập chiều rộng
  (setq A 1)
  (prompt "\nSelect Polylines to change: "); Chon pline thay đổi chiều rộng
  (while (/= A nil)
    (progn
      (setq A (entsel))
      (if (/= A nil)
        (progn
          (setq B (entget (car A)))
          (setq C (cdr (assoc 40 B)))
          (princ C)
          (command "pedit" A "W" LW "")
        )
      )
    )
  )
  (princ)
);Kết thúc chương trình
```

8.3 Các hàm logic

Để phối hợp nhiều biểu thức điều kiện thành một (dùng cho hàm **If** chẳng hạn) ta có thể dùng các hàm logic. Các hàm logic gồm có: **And**, **Or**, **Not.**

Hàm AND

Hàm **And** sẽ định giá trị các tham số lần lượt từ trái sang phải cho đến khi gặp tham số bằng *nil* đầu tiên thì dừng lại và trả về kết quả là *nil*. Nếu tất cả các tham số của nó có giá trị khác *nil*, hàm **And** trả về *T*.

(**And** EXPRESSION ...)

Ví dụ:

```
(setq A 10  B "YES"  C nil  D '(0 0)  E 20.5)
(and (> A 0) (< A 20))          trả về T
(and (> A 0) (= A 3) (< A 20))  trả về nil
(and (null C))                  trả về T
(and)                           trả về T      Không có tham số
(and D E)                       trả về T      D và E được gán giá trị khác nil
```

Hàm OR

Hàm **Or** sẽ định giá trị các tham số lần lượt từ trái sang phải cho đến khi gặp tham số khác *nil* đầu tiên thì dừng lại và trả về kết quả là *T*. Nếu tất cả các tham số đều bằng *nil*, hàm **Or** trả về giá trị *nil*.

(**Or** EXPRESSION ...)

Ví dụ:

```
(setq A nil  B nil  C 20)
(or (> C -10) (C < 30))         trả về T
(or A B)                        trả về nil
(or A B C)                      trả về T
(or)                            trả về nil    Không tìm thấy tham số khác nil
(or (setq A 1) (setq B 2)       trả về T      vì (setq A 1) trả về giá trị khác nil
```

Hàm NOT

Hàm **Not** trả về *T* nếu tham số của nó có giá trị *nil*. Ngược lại, trả về *nil*. Nó có thể được dùng để đổi ngược kết quả của các hàm **And** và **Or**.

(**Not** ITEM)

Ví dụ:

```
(not nil)                       trả về T
(not T)                         trả về nil
(not (and 1 2))                 trả về nil
```

Ví dụ:

Chương trình sau dùng hàm **Or** để quyết định việc rẽ nhánh chương trình. Để ý cách dùng các hàm **If** lồng nhau.

```
;Tên chương trình: FILETIME.LSP
;DWGTITLED: Cho biết bản vẽ hiện hành có được đặt tên hay chưa. Các
;          giá trị cho biến này như sau: 0 = Bản vẽ chưa được đặt tên,
;          1  = Bản vẽ đã được đặt tên.
```

```
;DWGWRITE:  Kiểm soát chức năng đọc/ghi của bản vẽ đang mở. Các giá
;           trị cho bản vẽ này như sau: 0 = read only, 1 = read-write.
;           Chỉ dùng cho Release 13, không dùng cho Release 14.
;SAVETIME   Thời gian tự động lưu file. Giá trị tính bằng phút, từ 0 đến
;           32767.
;LOGINNAME: Tên user name khi cài đặt AutoCAD.
(defun C:FILETIME (/  VAR1 VAR2 VAR3 VAR4 RES)
  (setvar "cmdecho" 0)
  (setq VAR1 (getvar "Dwgtitled")
        VAR2 (getvar "Dwgwrite")
        VAR3 (getvar "Savetime")
        VAR4 (getvar "Loginname")
  )
  (command ".textscr")          ;Chuyển sang màn hình văn bản (F2)
  (princ "\n        ")          ;Xuống hàng và 9 khoảng trắng
  (princ VAR4)                  ;Sau đó in tên user name
  (if                                       ;Bắt đầu IF
    (or                                     ;Bắt đầu OR
      (= (getvar "Dwgname") "UNNAMED") ;Bản vẽ đã được đặt tên chưa
          (= VAR1 0)
    )                                       ;Đóng OR
    (progn                                  ;Bắt đầu PROGN – THEN
      (princ "\n\nThe current drawing file is unnamed ...")
      (setq RES (strcase
        (getstring "\nWould you like to name this file? <Y/N>: "))
      )                                     ;Đóng SETQ
      (if                                   ;Bắt đầu nested IF
(= RES "Y")                                 ;expr1
        (command ".saveas" "R14" "~" ".textscr")  ;THEN expr2
        (princ "\nNo file name given.")     ;ELSE expr3
      )                                     ;Đóng nested IF
    )                                       ;Đóng PROGN
    (progn                                  ;Bắt đầu PROGN- ELSE
      (princ "\n\nThe current drawing file is: ")
      (princ (getvar "Dwgname"))
    )                                       ;Đóng PROGN
)                                           ;Đóng IF
(if                                         ;Bắt đầu IF
   (= VAR2 1)
   (progn                                   ;Bắt đầu PROGN - THEN
     (princ "\n\nThe read-only toggle is not enabled ...")
```



```
      (setq RES (strcase
        (getstring "\nEnable the read-only feature? <Y/N>: "))
      )
      (if                                   ;Bắt đầu nested IF
      (= RES "Y")
        (setvar "Dwgwrite" 0)               ;THEN
        (princ "\nRead-only not enabled.")  ;ELSE
      )                                     ;Đóng nested IF
    )                                       ;Đóng PROGN
  )                                         ;Đóng IF
  (princ "\n\nThe automatic drawing file saver is set to: <")
  (princ VAR3)
  (princ ">")
  (setq RES (strcase (getstring "\nWould you like to reset this? <Y/N> ")))
  (if                                       ;Bắt đầu IF
    (= RES "Y")
    (setvar "Savetime" (getint "n\New setting ,0-32767>: "))
   (princ "\nNo change in Savetime.")
  )                                         ;Đóng IF
  (princ)
) ;Kết thúc chương trình
```

Ví dụ:

Chương trình sau đây bảo đảm 2 điểm được nhập vào trước khi sử dụng chúng. Nếu ta không nhập một điểm mà nhấn phím ENTER thì hàm **Getpoint** trả về nil. Do đó ta dùng biểu thức điều kiện **And** để chắc chắn nếu hai điểm PT1, PT2 được nhập thì mới vẽ đa giác.

```
...
(if                                         ;Bắt đầu if
  (and                                      ;Bắt đầu AND
    (not (prompt "\nPick two point: "))     ;prompt trả về nil, do đó hàm
                                            ;not trả về T
    (setq PT1 (getpoint))           ;Nếu chọn một điểm thì hàm setq
    (setq PT2 (getpoint PT1))       ;trả về giá trị khác nil
  )
  (command ".POLYGON" "6" "E" PT1 PT2)      ;THEN expression
  (prompt  "\nTwo  points must be  picked!")  ;ELSE expression
)                                           ;Đóng  IF
```

Tóm tắt:

1. Hàm **If** dùng để tạo sự rẽ nhánh chương trình. Hàm **If** cần có 2 tham số TESTexpr và THENexpr. Tham số thứ ba ELSEexpr là tùy chọn.
2. Hàm **Progn** dùng để gộp nhiều biểu thức thành một biểu thức.
3. Các hàm logic cơ bản là **And, Or, Not.**

8.4 Rẽ nhánh chương trình phức tạp bằng hàm COND

Hàm **If** cho phép chương trình chia thành 2 nhánh. Khi cần phải chia thành nhiều nhánh, ta dùng hàm **Cond** *(CONDition).*

(**Cond** (TEST1 RESULT1 ...) (TEST2 RESULT2 ...) ...)

Hàm này chứa nhiều tham số kiểu danh sách. Phần tử đầu tiên trong mỗi danh sách là một biểu thức điều kiện. Nếu biểu thức TEST1 có giá trị khác nil (True) thì các biểu thức RESULT1 sẽ được thực hiện. Ngược lại, biểu thức TEST2 sẽ được định giá trị xem có khác nil hay không. Nếu TEST2 khác nil thì các biểu thức RESULT2 sẽ được thực hiện. Nếu TEST2 bằng nil, thì hàm **Cond** sẽ tiếp tục định giá trị các biểu thức điều kiện tiếp theo TEST3, TEST4 ...

Ví dụ:

Chương trình sau đây yêu cầu người sử dụng chọn một điểm trên màn hình, sau đó hiển thị thông báo về vị trí của điểm chọn so với các giá trị giới hạn bản vẽ.

```
;Tên file: CHECKPLZ.LSP
(defun C:CHECKPLZ (/ CHKPT LM)
  (setvar "cmdecho" 0)
  (setq CHKPT (getpoint "\nSelect point to check: "))
  (setq LM (getvar "LIMMAX"))
  (cond                                   ;Bắt đầu cond
    (                                     ;Bắt đầu first list
      (and                                ;Bắt đầu AND- TEST1
        (> (car CHKPT) (car LM))          ;So sánh các giá trị X
        (> (cadr CHKPT) (cadr LM))        ;So sánh các giá trị Y
      )                                   ;Đóng AND
     (prompt "\nPoint exceeds upper limits on both axes.\n")  ;RESULT1
    )                                     ;Đóng first list
    (                                     ;Bắt đầu second list
      (> (car CHKPT) (car LM))            ;TEST2
      (prompt "\nPoint exceeds upper limits on X-axis.\n")  ;RESULT2
    )                                     ;Đóng second list
    (                                     ;Bắt đầu third list
      (> (cadr CHKPT) (cadr LM))          ;TEST3
      (prompt "\nPoint exceeds upper limits on Y-axis.\n")  ;RESULT3
    )                                     ;Đóng third list
    (T                                    ;begin last list – First item is T (True)
      (prompt "\nSelected point does not exceed upper limits.")
;RESULT4
    )                                     ;Đóng last list
  )                                       ;Đóng COND
  (princ)
)                                         ;Đóng DEFUN
```

Trong ví dụ trên, hàm **Cond** có 4 biểu thức điều kiện. Ba điều kiện đầu kiểm tra các giá trị X, Y của CHKPT với các giá trị giới hạn bản vẽ. Nếu 3 điều kiện đầu không thỏa, chương trình sẽ hiện thông báo trong danh sách thứ 4, do đó biểu thức điều kiện thứ 4 phải luôn trả về giá trị khác nil. Trong ví dụ này ta dùng giá trị T làm biểu thức điều kiện.

Ví dụ:

Chương trình cho phép lựa chọn thực hiện các lệnh 3D Solid.

```
;Tên file: 3DSOLMOD.LSP
(defun C:SOL (/ RES)
  (setvar "cmdecho" 0)
  (defun startcom (helpstring)
    (princ (strcat "/n3-D solids operation for: " RES "\n" helpstring))
    (command (strcat "." RES))
  )
  (initget 1 "Extrude Revolve sLice seCtion Intersect Union Subtract eXit")
  (setq RES (getkword (strcat "3D tools- Extrude/Revolve/sLice/seCtion/
   Intersect/Union/Subtract/eXit: ")
              )
  )
  (cond ((= RES "Extrude")
         (startcom "Creates solid primitives by extruding 2D objects.")
        )
        ((= RES "Revolve")
```

```
        (startcom "Creates a circular solid by revolving a 2D object about
                  an axial path")
      )
      ((= RES "sLice")
       (startcom "Sclices a group of solids with a working or imainary
                  plane")
      )
      ((= RES "seCtion")
      (startcom "Employs the intersection of a specified plane and solids
                 to form a region")
      )
      ((= RES "Intersect")
        (startcom "Calculates the common volume shared by two or more
                  existing solids.")
      )
      ((= RES "Union")
        (startcom "Creates a composite solid by adding two or more
                   existing solids")
      )
      ((= RES "Subtract")
        (startcom "Composite solid created by subtracting voidable solids
                  from base solids")
      )
      ((= RES "eXit")
        (princ "\nExiting SOL routine")
      )
  )                                        ;Đóng cond
  (princ)
)
```

Tóm tắt:

1. Hàm **Cond** được sử dụng khi chương trình cần rẽ theo nhiều nhánh.
2. Nếu không có các hàm điều kiện, chương trình **AutoLISP** chẳng khác gì các *script file*. Nhờ các hàm này ta có thể tạo ra các chương trình giải quyết các vấn đề phức tạp.

8.5 Các ví dụ mẫu

Ví dụ 1: Chương trình giải phương trình bậc 2 sử dụng hàm **If**.

;Tên file: PTBHIF.LSP


```
(defun C:PTBH (/ a b c delta X1 X2)
  (initget 2)                                  ;Hệ số a phải khác 0
  (setq a (getreal "Nhap he so a: "))
  (setq b (getreal "Nhap he so b: "))
  (setq c (getreal "Nhap he so c: "))
  (setq delta (- (* b b)(* 4 a c)))
  (setq b (- 0 b))
  (if (< delta 0)                              ;Nếu delta < 0
    (princ "\n Phuong trinh bac hai khong co nghiem")
    (if (= delta 0)                            ;Nếu delta = 0
      (progn
        (princ "\nPhuong trinh bac hai co hai nghiem kep: ")
        (princ "\nX1 = X2 = ")
        (princ (/ (+ b (sqrt delta)) 2 a))
      )                                        ;Đóng progn
      (progn                                   ;Nếu delta > 0)
        (princ "\nPhuong trinh bac hai co hai nghiem thuc: ")
        (princ "\nX1 = ")
        (princ (/ (+ b (sqrt delta)) 2 a))
        (princ "\nX2 = ")
        (princ (/ (- b (sqrt delta)) 2 a))
      )                                        ;Đóng progn
    )                                          ;Đóng if
  )                                            ;Đóng if
  (princ)
)                                    ;Kết thúc chương trình
```

Ví dụ 2: Chương trình giải phương trình bậc 2 sử dụng hàm **Cond**.

```
;Tên file: PTBHCOND.LSP
(defun C:PTBH (/ a b c delta X1 X2)
  (initget 2)                                  ;Hệ số a phải khác 0
  (setq a (getreal "Nhap he so a: "))
  (setq b (getreal "Nhap he so b: "))
  (setq c (getreal "Nhap he so c: "))
  (setq delta (- (* b b)(* 4 a c)))
  (setq b (- 0 b))
  (cond
    ((< delta 0)                               ;Nếu delta < 0
     (princ "\n Phuong trinh bac hai khong co nghiem")
    )
```

```
   ((= delta 0)                                    ;Nếu delta = 0
      (progn
         (princ "\nPhuong trinh bac hai co hai nghiem kep: ")
         (princ "\nX1 = X2 = ")
         (princ (/ (+ b (sqrt delta)) 2 a))
      )                                            ;Đóng progn
    )
     ((> delta 0)                                  ;Nếu delta > 0)
      (progn
        (princ "\nPhuong trinh bac hai co hai nghiem thuc: ")
        (princ "\nX1 = ")
        (princ (/ (+ b (sqrt delta)) 2 a))
        (princ "\nX2 = ")
        (princ (/ (- b (sqrt delta)) 2 a))
      )                                            ;Đóng progn
     )
  )                                                ;Đóng cond
  (princ)
)       ;Kết thúc chương trình
```

Ví dụ 3: Kéo dài, cắt xén các đoạn thẳng để tạo các bức tường giao nhau (WALLTEE.LSP). Tiện ích này sử dụng để tạo các bức tường giao nhau hình chữ T. Các đối tượng là line được xén hoặc kéo dài đến line là bức tường.

```
;Tên file: WALLTEE.LSP
;Không thể thực hiện với đối tượng là plines hoặc mlines.
(defun nearestpt (entlist selectpt / temppt nearpt)
  (setq ang (angle (cdr (assoc 10 entlist))(cdr (assoc 11 entlist))))
  (setq temppt (polar selectpt (+ ang 1.5707963) 1))
  (setq nearpt (inters temppt selectpt (cdr (assoc 10 entlist))(cdr (assoc 11
entlist)) nil))
)
(defun c:walltee (/ top tee1 tee2 tee1selpt tee2selpt tee2sp tee2sp topsp topep
       int1 int2 dt1spsel dt2spsel dt1spint dt2spint selint1 selint2 topsint
       topeint flg cla)
  (setvar "cmdecho" 0)
  (setvar "highlight" 0)
  (setq cla (getvar"clayer"))
  (setq topsel (nearestpt (setq top (entget (car (setq tmp (entsel "\nSelect wall
       to be teed to:  "))))(cadr tmp))
```



```
  tee1sel (nearestpt (setq tee1 (entget (car (setq tmp (entsel "\nSelect first
     wall line:  "))))(cadr tmp))
  tee2sel (nearestpt (setq tee2 (entget (car (setq tmp (entsel "\nSelect second
     wall line:  "))))(cadr tmp))
  tee1sp (cdr (assoc 10 tee1))
  tee1ep (cdr (assoc 11 tee1))
  tee2sp (cdr (assoc 10 tee2))
  tee2ep (cdr (assoc 11 tee2))
  topsp (cdr (assoc 10 top))
  topep (cdr (assoc 11 top))
  int1 (inters tee1sp tee1ep topep topsp nil)
  int2 (inters tee2sp tee2ep topep topsp nil)
  dt1spsel (distance tee1sp tee1sel)
  dt2spsel (distance tee2sp tee2sel)
  dt1epsel (distance tee1ep tee1sel)
  dt1spint (distance tee1sp int1)
  dt2spint (distance tee2sp int2)
  selint1 (distance tee1sel int1)
  selint2 (distance tee2sel int2)
  topsint (distance topsp int1)
  topeint (distance topsp int2)
)
(command "erase" (cdar top)(cdar tee1)(cdar tee2) ""
"layer" "s" (cdr (assoc 8 top)) "")
(if (if (< topsint topeint)
    (command "line" topsp int1 "")
    (progn
      (command "line" topsp int2 "")
      (setq flg T)
    )
  )                                                ;if
  (command "line" topep int1 "")
  (command "line" topep int2 "")
)                                                  ;if
(if (= (+ dt1spsel selint1) dt1spint)
  (command "line" int1 tee1sp "")
  (command "line" int1 tee1ep "")
)
(if (= (+ dt2spsel selint2) dt2spint)
  (command "line" int2 tee2sp "")
  (command "line" int2 tee2ep "")
```

```
  )
  (command "layer" "s" cla "")
  (setvar "cmdecho" 1)
  (setvar "highlight" 1)
)                                   ;Kết thúc WALLTEE.LSP
```

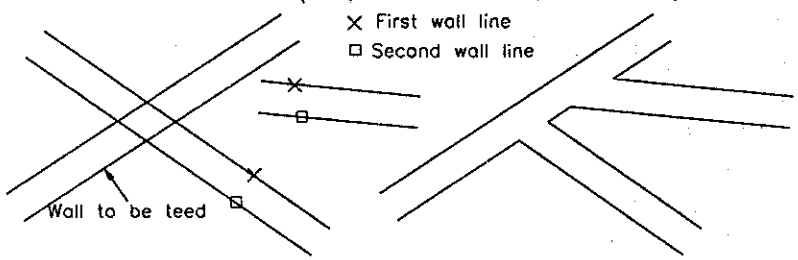
Khi tải chương trình WALLTEE.LSP bằng lệnh **Appload** ta thực hiện như sau:

Command: **Walltee** ↵

Select wall to be teed to: (Chọn line là bức tường)

Select first wall line: (Chọn line thứ nhất cần xén hoặc kéo dài)

Select second wall line: (Chọn line thứ hai cần xén hoặc kéo dài)



a) Trước **Walltee** b) Sau **Walltee**

8.6 Bài tập

1. Viết chương trình DIV.LSP tạo một hàm C:DIV yêu cầu nhập vào số chia và số bị chia. Nếu số chia khác không, hàm này trả về kết quả phép chia, ngược lại, hiện thông báo "ERROR: Cannot divide by zero!".
2. Viết chương trình PTS.LSP tạo một hàm C:PTS yêu cầu nhập vào hai điểm. Dùng hàm **Equal** với tham số FUZZ kiểm tra khoảng cách giữa hai điểm này có nhỏ hơn 0.5 đơn vị hay không. Nếu không thỏa mãn điều kiện, hiện thông báo hai điểm này không trùng nhau.
3. Trong **AutoCAD**, lệnh **Ddatte** hiển thị hộp thoại hiệu chỉnh thuộc tính của block "Edit Attributes". Lệnh **Ddedit** hiển thị hộp thoại hiệu chỉnh chữ số kích thước "Edit text". Hãy viết chương trình sử dụng hàm **If**, cho phép người sử dụng lựa chọn hiển thị một trong hai hộp thoại trên.
4. Hãy bổ sung vào chương trình C:CHECKPLZ việc kiểm tra chắc chắn có điểm nhập vào. Nếu điểm không được chọn, chương trình sẽ hiện thông báo tương ứng. Đặt tên chương trình là CHKPT.LSP
5. Viết chương trình SF.LSP, dùng hàm **If** thực hiện các công việc sau:

a. Dùng lệnh **Save** của **AutoCAD** để lưu file bản vẽ tại thư mục C:\

b. Dùng hàm **If** hỏi người sử dụng có muốn lưu file với cùng tên này ra đĩa mềm không (A: hoặc B:) (Gợi ý: dùng hàm **Strcat** và biến hệ thống DWGNAME).

6. Viết chương trình dùng hàm **Cond** để thực hiện các yêu cầu sau:

a. Kiểm tra các dữ liệu nhập, và trả về các thông báo lỗi nếu dữ liệu nhập sai.

b. Cho phép người sử dụng lựa chọn vẽ hình chữ nhật, hình vuông hoặc tam giác.

c. Dùng polyline để vẽ các đối tượng trên, kết hợp với các hàm cần thiết như **Polar** để tìm tọa độ các điểm cần thiết.

d. Đặt tên chương trình là P8-2.LSP

8.7 Lời giải

1.

```
;Tên file DIV.LSP
(defun C:DIV (/ NUM1 NUM2)
  (setq NUM1 (getreal "\nEnter divisor: ")
        NUM2 (getreal "\nEnter divident: ")
  )
  (if (zerop NUM1)
     (print "ERROR: Cannot divide by zero!")
     (print (strcat "The quotient is: " (rtos (/ NUM2 NUM1))))
  )
  (princ)
) ;Kết thúc chương trình
```

2.

```
;Tên file: C:PTS.LSP
;
(defun C:PTS (/ PT1 PT2)
  (setq PT1 (getpoint "\nFirst point: ")
       PT2 (getpoint "\nSecond point: ")
  )
  (if (EQUAL (distance PT1 PT2) 0 0.5)
     (princ "The points are the same.")
     (princ "The points are not the same.")
  )
  (princ)
) ;Kết thúc chương trình
```

3.

```
;Tên file: EX8-3.LSP
(defun C:ex8 ()
  (setvar "cmdecho" 0)
  (initget 1 "A T")
  (if
    (= "A" (getkword "\Attributes or Text Edit? <A/T>: "))
    (command ".DDATTE")
    (command ".ddedit")
  )
  (princ)
)
```

4.

```
;Tên file:  CHKPT.LSP
(defun C:CHECKPLZ (/ CHKPT LM)
  (setvar "cmdecho" 0)
  (setq CHKPT (getpoint "\nSelect point to check: "))
  (setq LM (getvar "LIMMAX"))
  (if (= 'LIST (Type CHKPT))
  (cond                                  ;Bắt đầu cond
    (                                    ;Bắt đầu first list
      (and                               ;Bắt đầu AND- TEST1
        (> (car CHKPT) (car LM))         ;So sánh với giá trị X
        (> (cadr CHKPT) (cadr LM))       ;So sánh với giá trị Y
      )                                  ;Đóng AND
      (prompt "\nPoint exceeds upper limits on both axes.\n")
       ;RESULT1
    )                                    ;Đóng first list
    (                                    ;Bắt đầu second list
      (> (car CHKPT) (car LM))           ;TEST2
      (prompt "\nPoint exceeds upper limits on X-axis.\n")   ;RESULT2
    )                                    ;Đóng second list
    (                                    ;Bắt đầu third list
      (> (cadr CHKPT) (cadr LM))         ;TEST3
      (prompt "\nPoint exceeds upper limits on Y-axis.\n")   ;RESULT3
    )                                    ;Đóng third list
    (T                                   ;Bắt đầu last list - First item là T
(True)
      (prompt "\nSelected point does not exceed upper limits.")   ;RESULT4
    )                                    ;Đóng last list
  )                                      ;Đóng COND
  (princ "\nPoint not selected")         ;ELSE
  )                                      ;Đóng IF
  (princ)
)
```

5.

```
;Tên file: SF.LSP
(defun C:SF ()
  (prompt "\nSave file on C.")
  (command ".qsave")
  (prompt "\nFile Saved")
  ;
  (initget 1 "Yes No")
  (if
    (= "Yes" (getkword "\nSave on floppy disk? <Y/N>: "))
    (progn      ;THEN
      (initget 1 "A B")
      (if
        (= "A" (getkword "\nSave on A: or B:? <A/B> . "))
        (command ".save" (strcat "A:" (getvar "dwgname")))
        (command ".save" (strcat "B:" (getvar "dwgname")))
      )
    )                                    ;Đóng progn
    (prompt "\nFile no save on floppy disk")  ;ELSE
  )
)
```

6.

```
;Tên file: P8-2.LSP
;Vẽ hình chữ nhật (rectangle), hình vuông (square) hoặc tam giác (triangle)
(defun C:p8-2 ()
  (setvar "CMDECHO" 0)
  ;
  (defun Triangle (/ PT1 PT2 PT3)
    (setq PT1 (getpoint "\n Enter the first point: "))       ; get PT1 location
    (setq PT2 (getpoint  "\n Enter the second point: "))  ; get PT2 location
    (setq PT3 (getpoint  "\n Enter the third point: "))      ; get PT3 location
    (If (or (null PT1) (null PT2) (null PT3))
      (prompt "\n 3 points must entered.")
      (command "PLINE" PT1 PT2 PT3 "close")    ;Vẽ triangle
    )
```

```
)                                                    ;Đóng defun triangle
,
(defun rectangl (/ CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: ") ;CPT1 is first
                                                       ;pick point
        CPT3 (getcorner CPT1 "\nSecond corner point: ")  ;CPT3 completes
                                                  ;the box
        CPT2 (list (car CPT3) (cadr CPT1))       ;CPT2 is constructed
        CPT4 (list (car CPT1) (cadr CPT3))       ;CPT4 is constructed
  )
  (If (or (null CPT1) (null CPT2) (null CPT3) (null CPT4))
    (prompt "\n 4 points must entered.")
    (command ".line" CPT1 CPT2 CPT3 CPT4 "Close") ;Vẽ rectangle
  )
)                                                    ;Đóng  defun rectangle

(defun square (/ LEN CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: "))  ;CPT1 is first pick
                                                          ;point
  (setq LEN (getdist CPT1 "\nEnter side of square: "))
  (if (or (null CPT1) (null LEN))
    (prompt "\n Must specified a point and side.")
    (progn
      (setq CPT2 (polar CPT1 0 LEN)
            CPT3 (polar CPT2 (/ pi 2) LEN)
            CPT4 (polar CPT3 pi LEN)
      )
      (command ".line" CPT1 CPT2 CPT3 CPT4 "Close") ;Vẽ square
    )                                          ;Đóng progn
  )                                            ;Đóng if
)                                              ;Đóng defun square
(initget 1 "Rectangle Square Triangle")
(setq ANS (getkword "\nDraw a Rectangle, Square or Triangle? <R/S/T>:
"))
(cond
  ((= "Triangle" ANS)(Triangle))
  ((= "Rectangle" ANS)(Rectangl))
  ((= "Square" ANS)(Square))
)                                              ;Đóng cond
(princ)
)                                              ;Kết thúc chương trình
```

Chương 9

CÁC VÒNG LẶP CHƯƠNG TRÌNH

Nội dung chương

1. Tạo các vòng lặp chương trình bằng các hàm: **Repeat, While, Append, 1+, 1-.**
2. Tạo và truy xuất danh sách bằng vòng lặp chương trình bởi các hàm: **Foreach, Set, Eval.**

Vòng lặp chương trình cho phép một hoặc nhiều biểu thức được thi hành nhiều lần. Vòng lặp được thực hiện theo một số lần nhất định hoặc được thực hiện cho đến khi điều kiện lặp không còn thỏa mãn nữa.

9.1 Các vòng lặp cơ bản

Các vòng lặp cơ bản trong chương trình được tạo bằng các hàm: **Repeat**, **While**, **Append**, **1+**, **1-**

Hàm REPEAT

Hàm **Repeat** tạo ra vòng lặp với một số lần nhất định.

(**Repeat** NUMBER EXPR ...)

Tham số NUMBER là một số *nguyên dương*, xác định số lần định giá trị các biểu thức EXPR. Hàm trả về giá trị của biểu thức cuối cùng trong vòng lặp.

Nếu tham số NUMBER bằng 0 hoặc là số âm, hàm REPEAT sẽ không định giá trị các biểu thức EXPR và trả về nil.

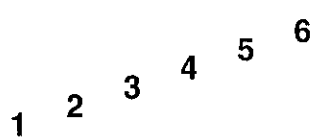
Ví dụ:

```
(setq M 10   N 4)
(repeat 10
  (setq M (+ 2 M))
  (setq N (+ 5 N))
)                          ; hàm REPEAT trả về 54
```

Trong ví dụ trên, hàm **Repeat** định giá trị các biểu thức 10 lần. Sau 10 lần, M=30, N=54. Do đó, hàm **Repeat** trả về giá trị 54.

Ví dụ:

Chương trình viết lên màn hình một chuỗi các số cách đều nhau, và trên một đường thẳng nghiêng so với phương ngang.




```
;NUMROW.LSP
;Mục đích: Tạo một chuỗi các số tăng dần bắt đầu từ số 0, xếp cách đều
;nhau trên một đường thẳng nghiêng một góc bất kỳ.
(defun C:NUMROW (/ PT1 DST DIR NN HN)
  (setvar "cmdecho" 0) (setvar "BLIPMODE" 0)
  (princ "\nText will be CENTER justified.")
  (setq PT1 (getpoint "\nStart point of number row: ")
  DST (getdist "\nDistance between numbers: ");Khoảng cách giữa các
                                              ;chữ số
  DIR (getorient "\nAngle: ")     ;Góc nghiêng của dòng chữ
  NN 0
  HN (getint "\nNumber sequence to be 0 through: ")
 )
 (command "Text" "c" PT1 "" "" "0")                ;Chữ số đầu tiên
 (repeat HN                                        ;Lặp HN lần
   (setq PT1 (polar PT1 DIR DST))                  ;Khoảng cách
   (command "TEXT" "C" PT1 "" "" (itoa (setq NN (+ NN 1))))
 )
)                                                  ;Kết thúc defun
```

Hàm WHILE

Hàm **While** tạo ra một vòng lặp có điều kiện. Vòng lặp này sẽ kết thúc khi điều kiện lặp TESTEXPR không còn thỏa mãn.

(**While** TESTEXPR EXPR ...)

Khi TESTEXPR vẫn còn trả về giá trị khác nil, các biểu thức EXPR sẽ tiếp tục được định giá trị. Quá trình này được thực hiện cho đến khi TESTEXPR trả về giá trị nil.

Ví dụ:

Vòng lặp sau đây sẽ chấm dứt khi nhập vào một số âm:

```
(setq N 0)
(while (not (minusp N))              ;Trong khi N vẫn chưa phải là số âm
 (setq N (getint "\nInput a negative integer: "));yêu cầu nhập giá trị N mới.
)
```

Vòng lặp **While** có thể sử dụng để cho phép người sử dụng chọn nhiều lựa chọn trong chương trình, cho đến khi chọn kết thúc chương trình.

Ví dụ:

Chương trình sau đây cho phép người sử dụng lựa chọn lần lượt các chức năng hiệu chỉnh Text, cho đến khi muốn kết thúc chương trình thì chọn Exit (tương tự lệnh **CHT** trong phần *Bonus* của **AutoCAD 14**).

```
;Tên file: CHT.LSP
(defun C:CHT (/ ANS NT)
```

```
(setvar "cmdecho" 0)
(prompt "\nSelect TEXT entity ...")
(command "SELECT" Pause)
(while
  (or
      (initget 1 "Position Style Height Rotation Text Exit")
      (/= "Exit"
        (setq ANS
          (getkword "\nChange Position/Style/Height/Rotation/Text/Exit: ")
        )                                ;Kết thúc SETQ
      )                                  ;Kết thúc /=
  )                                      ;Kết thúc OR
  (cond
      ((= "Position" ANS)
        (prompt "\nNew text location: ")
        (command "Change" "P" "" "" pause "" "" "" "")
      )
      ((= "Style" ANS)
        (prompt "\nNew text style: ")
        (command "Change" "P" "" "" "" pause "" "" "")
      )
      ((= "Height" ANS)
        (prompt "\nNew text height: ")
        (command "Change" "P" "" "" "" "" pause "" "")
      )
      ((= "Rotation" ANS)
        (prompt "\nRotation angle: ")
        (command "Change" "P" "" "" "" "" "" pause "")
      )
      ((= "Text" ANS)
        (setq NT (getstring T "\nNew text: "))
        (command "Change" "P" "" "" "" "" "" "" NT)
      )
  )                                      ;Kết thúc  COND
))                                       ;Kết thúc  WHILE và DEFUN
```

Dưới đây thêm một ví dụ sử dụng vòng lặp **While**:

Ví dụ:

Chương trình yêu cầu nhập các điểm chèn block TD. Khi nhấn ENTER thì chương trình kết thúc.

```
;Tên file: TD.LSP
; Yêu cầu phải có sẵn block "TD" bất kỳ
;
(defun C:TD (/ TDPT)
  (setq TDPT (getpoint "\nTD insertion point: "))
  (command ".insert" "TD" TDPT 1 1 0)
  (while
    (setq TDPT
      (getpoint "\nNext TD insertion point (or ENTER when done): ")
    )
    (command ".insert" "TD" TDPT 1 1 0)
  )                                      ;Đóng WHILE
)                                        ;Kết thúc chương trình
```

Trong ví dụ trên, hàm **Getpoint** có hai nhiệm vụ: yêu cầu nhập điểm và là biểu thức điều kiện cho hàm **While**. Khi nhập ENTER, hàm **Getpoint** trả về nil, vòng lặp **While** kết thuc va khi đó sẽ kết thúc chương trình.

Hàm APPEND

Hàm **Append** gộp nhiều danh sách thành một danh sách duy nhất. Hàm này có ý nghĩa là thêm các phần tử vào vị trí cuối cùng của một danh sách có sẵn.

(**Append** EXPR ...)

Ví dụ:

(append '(25 33.12) '(10.0))	trả về (25 33.12 10.0)
(append (list "A" "B") (list "C"))	trả về ("A" "B" "C")
(append (list 10 20) 30)	trả về lỗi vì không phải kiểu danh sách
(append (list 10 20) (list 30))	trả về (10 20 30)

Sử dụng hàm **Append** trong vòng lặp **While**, ta có thể bổ sung thêm phần tử vào một danh sách có sẵn.

Ví dụ: Các số thực do người sử dụng nhập vào được bổ sung liên tục vào một danh sách cho đến khi nhập ENTER.

```
(setq RNLST nil)                ;Bảo đảm RNLST là danh sách rỗng
(while (setq RN (getreal "\nType a number (or ENTER when done): "))
  (setq RNLST (append RNLST (list RN)))
)
```

Thực hiện chương trình:

Type a number (or ENTER when done): **10** ↵

Type a number (or ENTER when done): **20** ↵
Type a number (or ENTER when done): **23.4** ↵
Type a number (or ENTER when done): **123** ↵
Type a number (or ENTER when done): ↵
(10.0 20.0 23.4 123.0) ;Danh sách kết quả RNLST

✌ **Ví dụ:** Tạo danh sách gồm các điểm nhập vào

```
(setq PLST nil)
(while (setq P (getpoint "\nPick a point (or ENTER when  done): "))
   (setq PLST (append PLST (list P)))
)
```

Ta có thể dùng phương pháp này cho các hàm **GETxxxx** khác.

Các hàm đếm

Có hai hàm đơn giản thường dùng trong vòng lặp là cộng một số thêm 1 (hàm **1+**)và trừ một số đi 1 (hàm **1-**).

Hàm 1+

Hàm **1+** cộng tham số NUMBER thêm 1 và trả về kết quả này. Tham số NUMBER là số thực hoặc số nguyên.

(**1+** NUMBER)

✌ **Ví dụ:**

(1+ 4)	trả về 5
(1+ -10.8)	trả về –9.8

Hàm 1-

Hàm **1-** trừ tham số NUMBER bớt 1 và trả về kết quả này. Tham số NUMBER là số thực hoặc số nguyên.

(**1-** NUMBER)

✌ **Ví dụ:**

(1- 6)	trả về 5
(1- -10.8)	trả về -11.8

Các hàm đếm thường được dùng trong vòng lặp **While**.

✌ **Ví dụ:**

Số N sẽ lần lượt được cộng thêm 1 cho đến khi bằng 10 thì vòng lặp kết thúc.

```
(setq N 0)
(while (> 10 N)
  (setq N (1+  N))
)
```

Các hàm đếm được dùng khi cần định giá trị từng phần tử của chuỗi, danh sách, hoặc khi đếm số dòng trong một file văn bản.

✌ **Ví dụ:** Chương trình tách từ đầu tiên trong một chuỗi.

```
(setq N 1  TXT "Lap trinh AutoLISP")
(while
  (and
    (/= " " (substr TXT N 1))  ;Khi chưa tìm thấy khoảng trắng
    (/= (strlen TXT) N)        ;và chưa đến hết chuỗi
  )
  (setq N (1+  N))             ;tăng vị trí tìm kiếm thêm 1
)                              ;kết thúc hàm While. (N-1) sẽ là số lương
                               ;các ký tự của từ đầu tiên
(setq WORD1 (substr TXT 1 (1-  N))
)
```

Khi đó WORD1 sẽ được gán bằng từ **"Lap"**.

Tóm tắt:

1. Vòng lặp chương trình cho phep một hoặc nhiều biểu thức được thi hành nhiều lần.
2. Hàm **Repeat** được dùng khi số lần lặp được biết trước.
3. Hàm **While** cho phép vòng lặp tiếp tục cho đến khi biểu thức điều kiện trả về nil.
4. Hàm **Append** dùng để bổ sung các phần tử vao vị trí cuối cùng trong danh sách có sẵn.
5. Các hàm đếm **1+** và **1-** được dùng trong vòng lặp để tăng giảm các số đếm.

9.2 Truy xuất từng phần tử trong danh sách

Tạo và truy xuất danh sách bằng vòng lặp chương trình bởi các hàm: **Foreach, Set, Eval**

Hàm FOREACH

Hàm **Foreach** có cú pháp như sau:

(**Foreach** NAME LIST EXPR ...)

Hàm **Foreach** duyệt từng phần tử trong danh sách LIST. Tại mỗi thời điểm, giá trị của từng phần tử trong danh sách được gán tạm thời cho biến NAME. Sau đó các biểu thức EXPR được định giá trị. Nếu trong các biểu thức EXPR có chứa biến NAME, thì giá trị biến NAME trong từng thời điểm sẽ khác nhau. Nếu danh sách LIST rỗng, hàm **Foreach** sẽ trả về giá trị nil.

Ví dụ:

```
(foreach LTR (list "A" "B" "C" "D" "E" "F" "G") ;lần lượt các chữ cái được
                                                ;gán cho biến LTR
        (princ (strcat "\n" LTR))               ;và được in ra trên màn hình
)
(princ)
```

Hàm SET

Hàm **Set** rất có ích khi làm việc với các biến và tên biến trong hàm **Foreach**.

(**Set** SYM EXPR)

Tham số SYM phải là kiểu biến không định giá trị (dùng dấu ' trước tên biến hoặc hàm **Quote**). Hàm **Set** gán giá trị của biểu thức EXPR cho biến chứa trong tham số SYM.

Ví dụ:

```
(set 'a 10)          ;biến a được gán giá trị bằng 10
(set (quote b) 20)   ;biến b được gán giá trị bằng 20
```

So sánh hàm SETQ và hàm SET:

- Hàm **Setq** gán giá trị cho một biến mà không định giá trị cho biến này.
- Hàm **Set** trước tiên định giá trị của biến SYM. Sau đó mới gán giá trị cho biến tìm được.
- Do đó **Setq** có ý nghĩa là **Setq** = **Set** + **Quote**

Ví dụ:

```
(setq a 10)              a được gán bằng 10
(set 'a 10)              'a được định giá trị, trả về kết quả là biến a.
                         Sau đó a được gán giá trị bằng 10
(setq b 20)              B được gán bằng 20
(set (QUOTE b) 20)       (QUOTE b) được định giá trị, trả về kết quả là
                         biến b. Sau đó b được gán giá trị bằng 20.
(setq X 'a)              X được gán giá trị là biến 'a không định giá trị
(set X 30)               X được định giá trị là a. Sau đó a được gán
                         giá trị bằng 30.
```

Vì hàm **Foreach** duyệt từng phần tử của một danh sách, nên hàm **Set** rất có ích khi dùng chung với hàm này để gán giá trị cho nhiều biến.

Ví dụ:

```
(foreach S '(a b c)                    ;S lần lượt được gán các giá trị là
                                       ; 'a, 'b, 'c.
 (set S (getint "\nEnter an integer: ")) ; lần lượt thực hiện 3 biểu thức
                                       ;(set 'a (getint "\nEnter an integer: "))
                                       ;(set 'b (getint "\nEnter an integer: "))
                                       ;(set 'c (getint "\nEnter an integer: "))
)                                      ;do đó, các biến a, b, c được gán các
                                       ;số nguyên.
(princ list (a b c))                   ;In danh sách các số nguyên nhập
                                       ;vào.
```

Hàm EVAL

Hàm **Eval** trả về kết quả của việc định giá trị biểu thức EXPR.

(**Eval** EXPR)

Biểu thức EXPR được định giá trị cho đến mức cuối cùng, loại bỏ các kết quả ở các mức trung gian.

Ví dụ:

```
(setq X "YES"  Y 15  Z 'Y  W Z)
(eval X)          ;trả về "YES"
(eval Y)          ;trả về 15
(eval Z)          ;trả về 15: Z có giá trị là Y. Y được định giá trị tiếp
                  ;tục, kết quả trả về là 15
(eval W)          ;trả về 15: W có giá trị là Z. Z được định giá trị là Y.
```

```
                    ;Y được đinh giá trị là 15. Kết quả trả về là 15
(setq M (+ W 1)     ;Trả về lỗi vì W không phải kiểu số
(setq M (+ (eval W) 1) ;Trả về kết quả là 16.
```

Ví dụ:

Danh sách L sau đây được tạo ra bằng cách lần lượt đưa vào các phần tử là tên biến và giá trị của biến.

```
(setq L '())                          ;Khởi tạo danh sách L
(foreach S '(a b c)                   ;Duyệt từng phần tử của danh sách
  (set S (getint "\nEnter an integer: "));Nhập giá trị cho các biến a, b, c.
  (setq L (append L (list S)))        ;Đưa tên biến vào danh sách L
  (setq L (append L (list (eval S)))) ;Đưa giá trị của biến vào danh sách L
)
```

Thực hiện chương trình:

Enter an integer: **2** ↵
Enter an integer: **3** ↵
Enter an integer: **4** ↵
Command:**(list L)**↵
((A 2 B 3 C 4))

Ví dụ:

Đoạn chương trình sau (có thể được dùng cho bài tập 2) giới thiệu một phương pháp gán giá trị tham số cho các biến từ một chuỗi bằng cách dùng hàm **Read**.

```
(setq LTRLST (list "A" "B" "C" "D" "E" "F"))
(foreach VN LTRLST
  (set
    (read VN)
    (getreal
      (strcat "\nValue for dimension "  VN  ": ")
    )
  )
)
```

Ví dụ:

Chương trình sau yêu cầu người sử dụng nhập vào tên biến, sau đó nhập giá trị để gán cho biến này. Hàm **While** dùng để thực hiện công việc này nhiều lần. Hàm **Read** dùng để lấy ra tên biến từ chuỗi do người sử dụng nhập vào. Hàm **Set** dùng để trả về tên biến. Hàm **Eval** dùng để tính giá trị các biến để in ra trên màn hình.


```
;Tên file: VAR.LSP
(defun C:VAR (/ VNS VNL DT )
  (while (/= "" (setq VNS (getstring "\nEnter the variables name
          to assign: ")))
    (setq VNL (append VNL (list VNS)))      ;Hàm Append đòi hỏi tham
                                            ;số phải có kiểu danh sách
    (initget 1 "Integer Real String Point")  ;Các từ khóa hợp lệ
    (setq DT  (getkword "\nData type (Integer/Real/String/Point): "))
    (cond
      ((= DT  "Integer")
        (set (read VNS)                     ;Dùng hàm SET để VNS được
                                            ;đinh giá trị
          (getint
            (strcat "\nInteger value for " VNS "  ")
          )                                 ;Kết thúc GETINT
        )                                   ;Kết thúc SET
      )                                     ;Kết thúc đối số COND thứ nhất
      ((= DT  "Real")
        (set (read VNS) (getreal (strcat "\nReal value for " VNS ": ")))
      )                                     ;Kết thúc đối số COND thứ nhất
      ((= DT  "String")
        (set (read VNS) (getstring T (strcat
                              "\nString value for " VNS ": ")))
      )                                     ;Kết thúc đối số COND thứ ba
      ((= DT  "Point") argument
        (set (read VNS) (getpoint (strcat "\nPoint value for " VNS ": ")))
      )                                     ;Kết thúc đối số COND cuối cùng
    )                                       ;Kết thúc hàm COND
  )                                         ;Kết thúc WHILE
  (princ "\nVariables assignments made: ")
  (foreach VN VNL
    (princ (strcat "\n" VN " = "))          ;In tên biến và dấu =
    (princ                                  ;In giá trị của biến
      (eval
        (read VN)
      )                                     ;Kết thúc EVAL
    )                                       ;Kết thúc PRINC
  )                                         ;Kết thúc FOREACH
  (princ)
)                                           ;Kết thúc chương trình
```

Tóm tắt:

1. Có thể truy xuất các danh sách một cách hiệu quả bằng hàm **Foreach**.
2. Hàm **Set** là một phương phap khác dùng để tạo ra các biến. Nếu được dùng chung với hàm **Foreach**, có thể rút ngắn trình tự gán giá trị cho các biến.
3. Hàm **Eval** đùng để định giá trị các biểu thức **AutoLISP**. Nó trả về giá trị ở mức cuối cùng.
4. Vòng lặp là một trong những điểm mạnh nhất của ngôn ngữ lập trình **AutoLISP**.

9.3 Các ví dụ mẫu

Ví dụ 1: Chương trình tính giai thừa sử dụng hàm **Repeat**

```
;Tên file: GiAITHUA1.LSP
(defun C:GiaiThua (/ i n S)
  (setq n (getint "\nNhap so nguyen: "))
  (setq i 1)
  (setq S 1)
  (Repeat n                                    ;Lặp n lần
        (setq S (* S i))
        (setq i (1+ i))
  )                                            ;Đóng Repeat
  (princ "\nGiai Thua: ")
  (princ S)
  (princ)
)
```

Ví dụ 2: Chương trình tính giai thừa (dùng hàm **While**)

```
; Tên file: GiAITHUA2.LSP
(defun C:GIAITHUA (/ i n S)
  (setq n (getint "\nNhap so nguyen: "))
  (setq i 1)
  (setq S 1)
  (While (<= i n)                              ;Lặp cho đến khi lớn hơn n
         (setq S (* S i)
         (setq i (1+ i))
  )                                            ;Kết thúc While
  (princ "\nGiai Thua: ")
```



```
  (princ S)
  (princ)
)
```

Ví dụ 3: Chương trình vẽ mặt cắt cho miền tự định đường biên.

```
;Tên file: DWGHATCH.LSP
(defun C:DRWHATCH (/ OLD_CMDECHO PT HATCH_NAME
                     HATCH_SCALE HATCH_ANGLE)
  (prompt "\nDRWHATCH - Draw hatch by picking points.")
  (setq OLD_CMDECHO (getvar "CMDECHO"))
  (setvar "CMDECHO" 0)
  (setq HATCH_NAME (getstring "\nHatch pattern name: "))
  (setq HATCH_SCALE (getdist "\nHatch scale: "))
  (setq HATCH_ANGLE (getreal "\nHatch angle: "))
  (setq PT (getpoint "\nFirst point: "))
  ;
  (command ".pline" PT)
  (while (setq PT (getpoint PT "\nNext point: "))
         (command PT)
  )
  (command "C")
  (command ".Select" "L")
  (command ".HATCH" HATCH_NAME
                    HATCH_SCALE HATCH_ANGLE  "L" "")
  (command ".erase" "P" "")
  (setvar "CMDECHO" OLD_CMDECHO)
  (prompt "\n\nProgram complete. ")
  (princ)
)
```

Ví dụ 4: Tạo các hình dạng đám mây (CLOUD.LSP) sử dụng hàm **While**. Chương trình sử dụng để vẽ hình dạng đám mây. Hình dạng đám mây là tập hợp các cung tròn. Tùy theo chiều định điểm (ngược chiều hoặc cùng chiều kim đồng hồ ta có các hình dạng đám mây khác nhau).

```
;Tên file: CLOUD.LSP
(defun C:CLOUD()
  (setvar"cmdecho"0)(setvar"orthomode"0)
  (prompt"\nThis utility cannot close the pline, and stops only when
         canceled.")
```

Tóm tắt:

1. Có thể truy xuất các danh sách một cách hiệu quả bằng hàm **Foreach**.
2. Hàm **Set** là một phương pháp khác dùng để tạo ra các biến. Nếu được dùng chung với hàm **Foreach**, có thể rút ngắn trình tự gán giá trị cho các biến.
3. Hàm **Eval** dùng để đinh giá trị các biểu thức **AutoLISP**. Nó trả về giá trị ở mức cuối cùng.
4. Vòng lặp là một trong những điểm mạnh nhất của ngôn ngữ lập trình **AutoLISP**.

9.3 Các ví dụ mẫu

Ví dụ 1: Chương trình tính giai thừa sử dụng hàm **Repeat**

```
;Tên file: GiAITHUA1.LSP
(defun C:GiaiThua (/ i n S)
  (setq n (getint "\nNhap so nguyen: "))
  (setq i 1)
  (setq S 1)
  (Repeat n                                    ;Lặp n lần
        (setq S (* S i))
        (setq i (1+ i))
   )                                           ;Đóng Repeat
  (princ "\nGiai Thua: ")
  (princ S)
  (princ)
)
```

Ví dụ 2: Chương trình tính giai thừa (dùng hàm **While**)

```
Tên file: GiAITHUA2.LSP
(defun C:GIAITHUA (/ i n S)
  (setq n (getint "\nNhap so nguyen: "))
  (setq i 1)
  (setq S 1)
  (While (<= i n)                              ;Lặp cho đến khi lớn hơn n
         (setq S (* S i)
         (setq i (1+ i))
  )                                            ;Kết thúc While
  (princ "\nGiai Thua: ")
```



```
  (princ S)
  (princ)
)
```

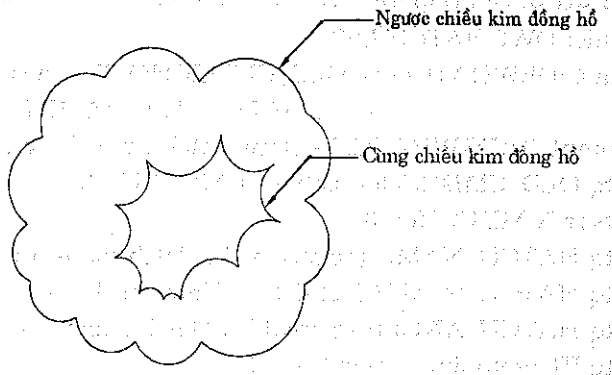
Ví dụ 3: Chương trình vẽ mặt cắt cho miền tự đinh đường biên.

```
;Tên file: DWGHATCH.LSP
(defun C:DRWHATCH (/ OLD_CMDECHO PT HATCH_NAME
                     HATCH_SCALE HATCH_ANGLE)
  (prompt "\nDRWHATCH - Draw hatch by picking points.")
  (setq OLD_CMDECHO (getvar "CMDECHO"))
  (setvar "CMDECHO" 0)
  (setq HATCH_NAME (getstring "\nHatch pattern name: "))
  (setq HATCH_SCALE (getdist "\nHatch scale: "))
  (setq HATCH_ANGLE (getreal "\nHatch angle: "))
  (setq PT (getpoint "\nFirst point: "))
  ;
  (command ".pline" PT)
  (while (setq PT (getpoint PT "\nNext point: "))
         (command PT)
  )
  (command "C")
  (command ".Select" "L")
  (command ".HATCH" HATCH_NAME
                    HATCH_SCALE HATCH_ANGLE  "L" "")
  (command ".erase" "P" "")
  (setvar "CMDECHO" OLD_CMDECHO)
  (prompt "\n\nProgram complete. ")
  (princ)
)
```

Ví dụ 4: Tạo các hình dạng đám mây (CLOUD.LSP) sử dụng hàm **While**. Chương trình sử dụng để vẽ hình dạng đám mây. Hình dạng đám mây là tập hợp các cung tròn. Tùy theo chiều định điểm (ngược chiều hoặc cùng chiều kim đồng hồ ta có các hình dạng đám mây khác nhau).

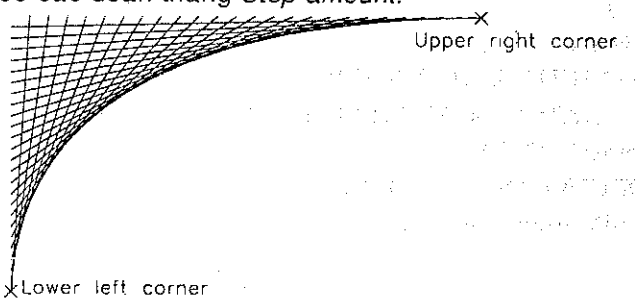
```
;Tên file: CLOUD.LSP
(defun C:CLOUD()
  (setvar"cmdecho"0)(setvar"orthomode"0)
  (prompt"\nThis utility cannot close the pline, and stops only when
```

```
  (setq a(getpoint"\nFirst point: "))
  (prompt"\r                \rTo point: ")
  (command"pline"a"w""0""""a")
  (while t(command"a""130"pause))
)       ;Kết thúc file CLOUD.LSP
```



Ví dụ 5: Vẽ các đường thẳng (WARP.LSP)

Chương trình này có sử dụng hàm **Repeat** để tạo các đoạn thẳng có các đỉnh nằm cách đều nhau trên hai cạnh kế tiếp của hình chữ nhật được định bởi hai điểm *Upper left corner*, *Lower right corner* và số các đoạn thẳng *Step amount*.




```
;Tên file: WARP.LSP
(defun C:WARP (/ p1 p2 st x1 y1 x2 y2 sx sy b x5 y5 p3 p4)
  (setq p1 (getpoint "\nUpper left corner: "))
  (setq p2 (getpoint "\nLower right corner: "))
  (setq st (getint "\nStep amount: "))
  (setq x1 (car p1))
  (setq y1 (cadr p1))
  (setq x2 (car p2))
  (setq y2 (cadr p2))
  (setq sx (/ (- x2 x1) st))
  (setq sy (/ (- y1 y2) st))
  (setq b 0)
```



```
  (repeat st
    (setq b (+ b 1))
    (setq x5 (+ x1 (* b sx)))
    (setq y5 (- y1 (* sy (- b 1))))
    (setq p3 (list x1 y5))
    (setq p4 (list x5 y2))
    (command "line" p3 p4 "")
  )
); Kết thúc file WARP.LSP
```

Ví dụ 6: Tạo các dãy gạch xếp cách đều (BRICKCAP.LSP). Chương trình này sử dụng để vẽ dãy gạch có dạng hình chữ nhật. Ta cần phải nhập chiều rộng (width) và chiều dài (length) của viên gạch và khe hở giữa các viên gạch.

```
;Tên file BRICKCAP.LSP
(defun newerr (ne)
  (if (/= ne "Function cancelled")
    (princ (strcat "\nError: " ne))
  )
  (setvar "OSMODE" om)
  (setvar "CMDECHO" ce)
  (setvar "BLIPMODE" bm)
  (setq *error* olderr)
  (princ)
)
(defun c:BRICKCAP (/ sp ep ds an wd lg bs om ce bm nx nu p1 p2 p3 p4)
  (setq olderr *error*)
  (setq *error* newerr)
  (while (= sp nil) (setq sp (getpoint "\nStart of first brick: ")))
  (while (= ep nil) (setq ep (getpoint "\nFinish point: ")))
  (setq ds (distance sp ep))
  (setq an (angle sp ep))
  (setq wd (getdist "\nPick or enter width of bricks <4.0>: "))
  (if (= wd nil) (setq wd 4.0))
  (setq lg (getdist "\nPick or enter length of bricks <8.0>: "))
  (if (= lg nil) (setq lg 8.0))
  (setq bs (getdist "\nPick or enter distance between bricks <W + L / 20>: "))
  (if (= bs nil) (setq bs (/ (+ wd lg) 20)))
  (setq om (getvar "OSMODE")) (setvar "OSMODE" 0)
  (setq ce (getvar "CMDECHO")) (setvar "CMDECHO" 0)
  (setq bm (getvar "BLIPMODE")) (setvar "BLIPMODE" 0)
```

```
(setq nx (+ bs lg)
  nu (/ ds nx)
  p1 sp
  p2 (polar p1 an lg)
  p3 (polar p2 (+ an (/ pi 2)) wd)
  p4 (polar p1 (+ an (/ pi 2)) wd)
)
(repeat (+ (fix nu) 1)
  (command "LINE" p1 p2 p3 p4 "C")
  (setq p1 (polar p1 an nx)
    p2 (polar p2 an nx)
    p3 (polar p3 an nx)
    p4 (polar p4 an nx)
  )
)
(setvar "OSMODE" om)
(setvar "CMDECHO" cm)
(setvar "BLIPMODE" bm)
(setq *error* olderr)
(princ)
)        ;Kết thúc file BRICKCAP.LSP
```

Sau khi tải chương trình vào **AutoCAD** bằng lệnh **Appload** ta nhập tên **Brickcap** vào dòng lệnh.

Command:**Brickcap**↵

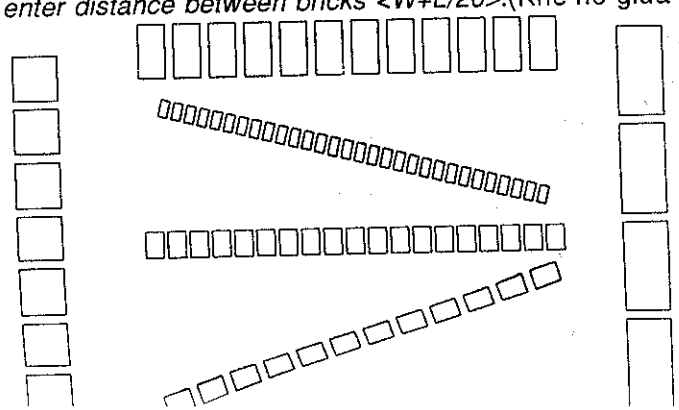
Start of first brick: (Điểm đầu tiên của hàng gạch)

Finish point: (Điểm cuối của hàng gạch)

Pick or enter width of bricks <4.0>: (Chiều rộng của viên gạch)

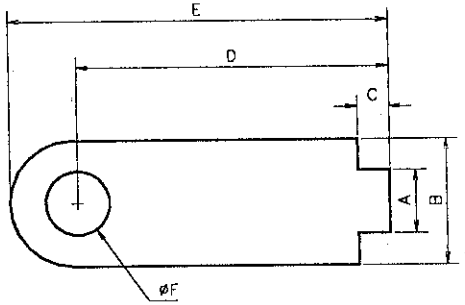
Pick or enter length of bricks <8.0>:(Chiều dài của viên gạch)

Pick or enter distance between bricks <W+L/20>:(Khe hở giữa các viên gạch)

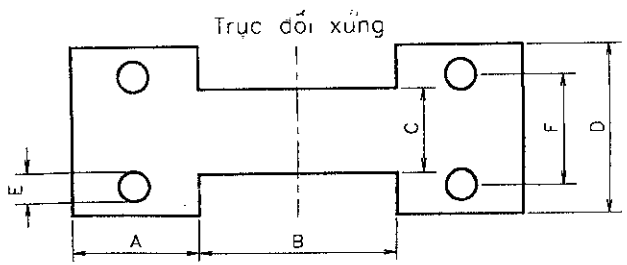


9.4 Bài tập

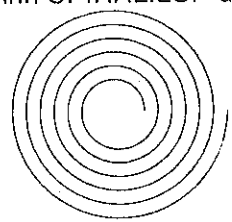
1. Viết chương trình LASTWORD.LSP dùng vòng lặp **While** trả về từ cuối cùng trong một chuỗi. Để không gây ra lỗi, vòng lặp phải dừng lại khi đi đến đầu hoặc cuối chuỗi.
2. Viết chương trình PART1.LSP yêu cầu người sử dụng nhập các kích thước để vẽ chi tiết sau:



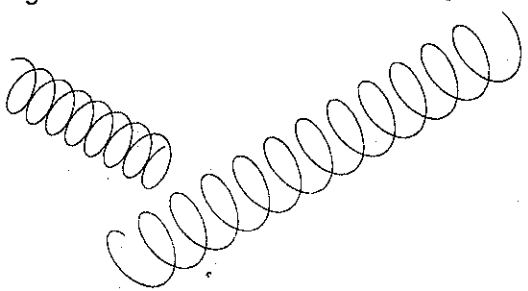
   - Cho phép người sử dụng lựa chọn hiện hoặc không hiện file slide (.SLD) biểu diễn hình vẽ chi tiết.
   - Sau đó dùng hàm **Foreach**, yêu cầu nhập 6 kích thước từ A đến F.
   - Dùng tham số LIST của hàm **Foreach** để chứa các kích thước.
   - Hiện các kích thước để người sử dụng kiểm tra. Sau đó cho phép người sử dụng lựa chọn tiếp tục chương trình hoặc kết thúc.
   - Chương trình sẽ kiểm tra sự hợp lệ của các kích thước. Ví dụ: Kích thước B phải lớn hơn kích thước A. Nếu các kích thước không hợp lệ, chương trình sẽ hiện thông báo và kết thúc.
   - Vẽ chi tiết với các kích thước trên.
3. Viết chương trình dùng vòng lặp **While** cho phép người sử dụng lựa chọn vẽ hình chữ nhật, vẽ hình chữ nhật tô, vẽ tam giác, hoặc kết thúc chương trình.
4. Viết chương trình chèn khối nhiều lần. Chương trình yêu cầu người sử dụng nhập tên khối, số lượng sẽ chèn và hệ số tỉ lệ. Sau đó dùng vòng lặp **Repeat** yêu cầu nhập các điểm chèn và góc quay.
5. Viết chương trình PART2.LSP vẽ chi tiết sau. Kiểm tra các kích thước hợp lệ.



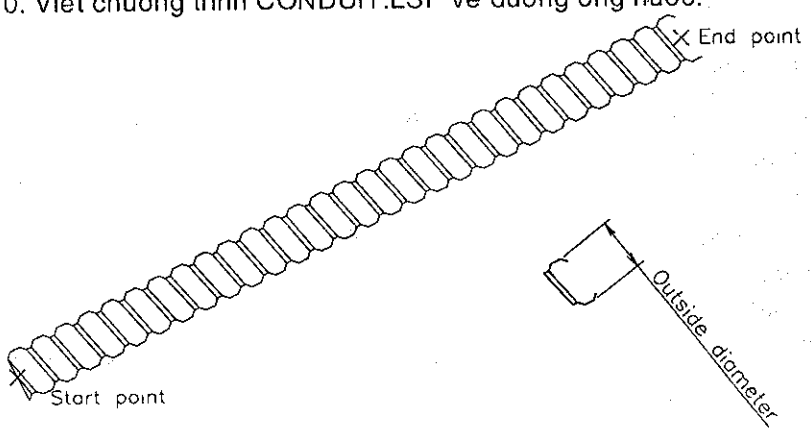
6. Viết chương trình TD2.LSP tương tự như chương trình TD.LSP ở chương này. TD2.LSP yêu cầu nhập số lượng block TD cần chèn. Sau khi chèn xong thì chương trình kết thúc.
7. Viết chương trình WORDCT.LSP yêu cầu nhập một chuỗi, sau đó đếm số lượng các từ trong chuỗi này. (Gợi ý: Đếm các khoảng trắng giữa các từ).
8. Viết chương trình SPIRAL.LSP để vẽ đường xoắn ốc 2D.



9. Viết chương trình HELIX3D.LSP để vẽ đường xoắn ốc 3D.



10. Viết chương trình CONDUIT.LSP vẽ đường ống nước.

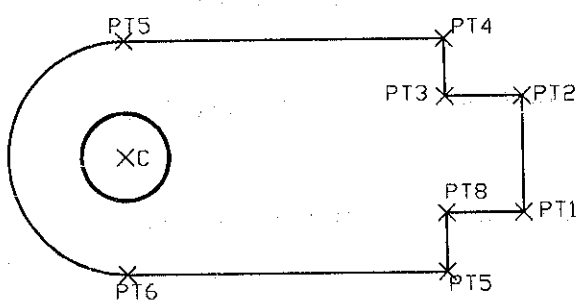


9.5 Lời giải

1.

```
;Tên file: LASTWORD.LSP
(defun C:lastword (/ LEN TXT N LWORD)
  (setq TXT (getstring T "\nEnter a string:  ")
        LEN (strlen TXT)
        N LEN                    ;tìm ngược từ cuối chuỗi về đầu chuỗi
  )
  (while
    (and
      (/= N 0)                   ;khi chưa đến đầu chuỗi
      (/= " " (substr TXT N 1)) ; và chưa tìm thấy khoảng trắng
    )
    (setq N (1- N))              ;giảm vị trí tìm kiếm đi 1
  )                              ;kết thúc hàm While. (N+1) sẽ là vị trí bắt
                                 ;đầu từ cuối cùng, (LEN-N) là số lương
                                 ;các ký tư của từ cuối cùng
  (setq LWORD (substr TXT (+ 1 N) (- LEN N)))
  (princ (strcat "Last word is: " LWORD))
  (princ)
)                                ;Kết thúc  defun
```


2.



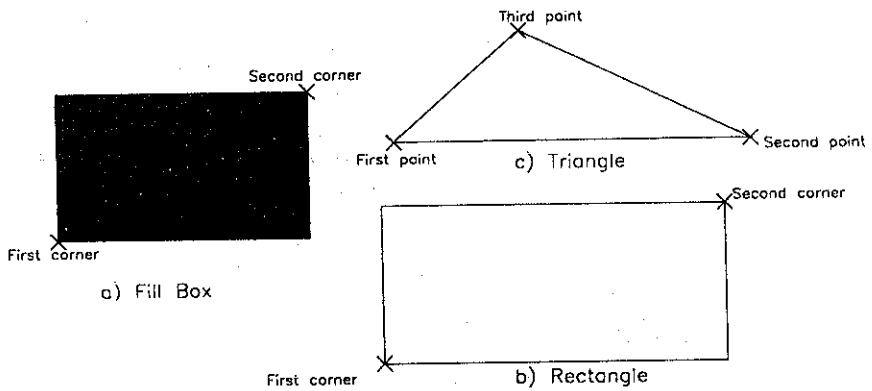

```
;Tên file: PART1.LSP
(defun C:Part1 ()
;
(setq LTRLST (list "A" "B" "C" "D" "E" "F"))
(foreach VN LTRLST
  (set
    (read VN)
    (getreal
      (strcat "\nValue for dimesion "  VN  ": ")
    )
  )
)
;
(princ "Dimensions of part are: ")
  (foreach VN LTRLST
    (princ (strcat "\n" VN " = "))
```

```
    (princ
      (eval
        (read VN)
      )                                ;Kết thúc EVAL
    )                                  ;Kết thúc PRINC
  )                                    ;Kết thúc FOREACH
;
  (initget 1 "Yes No")
  (if (= "Yes" (getkword "\nDraw the part? <Y/N>: "))
  (progn                               ;progn1
;
  (setq Msg "")                        ;Bắt đầu các dòng nhắc lỗi
  (if (>= A B)
     (setq Msg "A must less than B")
  )
  (if (>= C D)
       (setq Msg (strcat Msg "\nC must less than D"))
  )
  (if (>= D E)
       (setq Msg (strcat Msg "\nD must less than E"))
  )
  (if (>= F B)
       (setq Msg (strcat Msg "\nF must less than B"))
  )
;
  (if (/= Msg "")
      (princ (strcat Msg "\nCan't draw a part"))
        (progn
        (setq PT1 (getpoint "\nPick a point: ")
              PT2 (polar PT1 (/ Pi 2) A)
              PT3 (polar PT2 Pi C)
             PT4 (polar PT3 (/ Pi 2) (/ (- B A) 2))
              PT5 (polar PT4 Pi (- E (/ B 2) C))
              PT6 (polar PT5 (- (/ Pi 2)) B)
              PT7 (polar PT6 0 (- E (/ B 2) C))
             PT8 (polar PT7 (/ Pi 2) (/ (- B A) 2))
        )
        (command ".redraw"
                 ".pline" PT1 PT2 PT3 PT4 PT5 "a" PT6 "L" PT7 PT8 "c"
                  ".circle" (list (- (car PT1) D) (+ (cadr PT1) (/ A 2))) "d" F
        )
```



```
      )                                ;Kết thúc PROGN
    )                                  ;Kết thúc IF
    )                                  ;Kết thúc PROGN1
  )                                    ;Kết thúc IF
(princ)
)                                      ;Kết thúc DEFUN
```

3.




```
;Tên file: DRAW.LSP
(defun C:Draw (/ ANS NT)
  (setvar "cmdecho" 0)
;
(defun Rectangle (/ CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: ") ;CPT1 là điểm chọn đầu
   CPT3 (getcorner CPT1 "\nSecond corner point: ") ;CPT3 là điểm chọn cuối
      CPT2 (list (car CPT3) (cadr CPT1))          ;CPT2 được vẽ
      CPT4 (list (car CPT1) (cadr CPT3))          ;CPT4 được vẽ
)
(command ".line" CPT1 CPT2 CPT3 CPT4 "Close")    ;Vẽ hình chữ nhật
)
;
(defun FillBox (/ CPT1 CPT2 CPT3 CPT4)
  (setq CPT1 (getpoint "\nSelect first corner point: ")
   CPT3 (getcorner CPT1 "\nSecond corner point: ")
      CPT2 (list (car CPT3) (cadr CPT1))
      CPT4 (list (car CPT1) (cadr CPT3))
)
(command ".Solid" CPT1 CPT2 CPT4 CPT3 "")
)
;
(defun Triangle (/ PT1 PT2 PT3)
```

```
  (setq PT1 (getpoint "\n Enter the first point: "))          ;Gán vị trí PT1
  (setq PT2 (getpoint PT1 "\n Enter the second point: "))     ;Gán vị trí PT2
  (setq PT3 (getpoint PT2 "\n Enter the third point: "))      ;Gán vị trí PT3
  (command "PLINE" PT1 PT2 PT3 "close")                       ;Vẽ tam giác
)
;
;Main program
 (while
    (or
       (initget 1 "Rectangle Fill Triangle Exit")
       (/= "Exit"
         (setq ANS
           (getkword "\nDraw a Rectangle/Fill box/Triangle/Exit: ")
         )                              ;Kết thúc SETQ
       )                                ;Kết thúc /=
    )                                   ;Kết thúc OR
    (cond
       ((= "Rectangle" ANS)
         (prompt "\nDraw a rectangle: ")
         (Rectangle)
       )
       ((= "Fill" ANS)
         (prompt "\nDraw a box: ")
       (FillBox)
       )
       ((= "Triangle" ANS)
         (prompt "\nDraw a triangle: ")
         (Triangle)
       )
    )                   ;Kết thúc COND
)
;
  (princ)
)                       ;Kết thúc WHILE và DEFUN
```

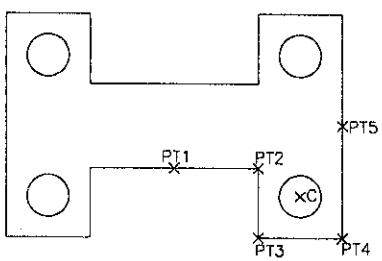
4.

```
;Tên file: MBI.LSP
(defun C:MBI (/ BNAME NUM SFACTOR INSPT ANG CNT)
;Multiple Insert block
;
  (setq BNAME (getstring "\nBlock name:")
```



```
      NUM    (getint  "Number of insertion: ")
       SFACTOR (getreal "Insertion scale factor: ")
)                                       ;Kết thúc SETQ
;
(setq CNT 0)
(repeat NUM
  (setq CNT(1+  CNT))
  (setq INSPT (getpoint (strcat "\nInsert point " (itoA CNT)  ": " ))
        ANG    (getreal "\nRotation angle: ")
  )
  (command ".insert" BNAME INSPT SFACTOR SFACTOR ANG)
)                               ; Kết thúc REPEAT
;
)                               ;Kết thúc defun
```

5.




```
;Tên file: PART2.LSP
(defun C:Part2 (/ VN LTRLST PT1 PT2 PT3 PT4 PT5)
(setq LTRLST (list "A" "B" "C" "D" "E" "F"))
(foreach VN LTRLST
  (set
    (read VN)
    (getreal
       (strcat "\nValue for dimesion "   VN   ": ")
    )
  )
)
;
;
(princ "Dimensions of part are: ")
  (foreach VN LTRLST
    (princ (strcat "\n" VN " = "))
(princ
       (eval
```

```
        (read VN)                    ;Tên biến.
      )                              ;Kết thúc EVAL
    )                                ;Kết thúc PRINC
  )                                  ;Kết thúc FOREACH


(initget 1 "Yes No")
(if (= "Yes" (getkword "\nDraw the part? <Y/N>: "))
(progn        ;progn1

(setq Msg "")                        ;Thực hiên lời nhắc lỗi
(if (>= C D)
      (setq Msg (strcat Msg "\nC must less than D"))
)
(if (>= E A)
      (setq Msg (strcat Msg "\nE must less than A"))
)
(if (>= E C)
      (setq Msg (strcat Msg "\nE must less than C"))
)
(if (>= F D)
      (setq Msg (strcat Msg "\nE must less than C"))
)


(if (/= Msg "")
    (princ (strcat Msg "\nCan't draw a part"))
      (progn
      (setq PT1 (getpoint "\nPick a point: ")
          PT2 (polar PT1 0 (/ B 2))
          PT3 (polar PT2 (- (/ Pi 2)) (/ (- D C) 2))
        PT4 (polar PT3 0 A)
        PT5 (polar PT4 (/ Pi 2) (/ D 2))
      )
      (command "redraw"
              "Pline" PT1 PT2 PT3 PT4 PT5 ""
              "Select" "L" ""
              "Mirror" "P" "" PT1 (polar PT1 (/ Pi 2) 1) "N"
              "Mirror" "P" "L" "" PT5 (polar PT5 0 1) "N"
              "Circle" (list (- (car PT5) (/ A 2)) (- (cadr PT5) (/ F 2))) "d" E
              "Array" "L" "" "R" 2 2 F (-(+ A B))
```



```
      )
      )                              ;Kết thúc PROGN
  )                                  ;Kết thúc IF
  )                                  ;Kết thúc PROGN
 )                                   ;Kết thúc IF (draw the part?)
(princ)
)                                    ;Kết thúc DEFUN
```

6.

```
;Tên file: TD2.LSP
;Multiple Tie-Dot creation routine.
;Requires that a block named "TD" exists, which represents the proper size
;dot.
;
(defun C:TD2 (/ TDPT)
  (setq NUM (getint "\nNumber of insertion point: "))
  (repeat NUM
    (setq TDPT (getpoint "\nTie-Dot insertion point: "))
    (command ".insert" "TD" TDPT 1 1 0)
  );end REPEAT
  (princ)
)                                    ;Kết thúc DEFUN
```

7.

```
;Tên file: WORDCT.LSP
;Count the number of words in the string
;
(defun C:WORDCT (/ TXT N COUNT)
;
(defun GETWORD ()
 (while
  (and
    (/= " " (substr TXT N 1))  ;Khi chưa tìm thấy khoảng trắng
    (/= (strlen TXT) N)        ;và chưa đến hết chuỗi
  )
  (setq N (1+  N))             ;tăng vi trí tìm kiếm thêm 1
  )                            ;kết thúc hàm While.

 )                             ;Kết thúc DEFUN GETWORD
;
```

```
(defun GETSPACE()
  (while
   (and
    (= " " (substr TXT N 1))    ;Khi vẫn còn khoảng trắng
    (/= (strlen TXT) N)          ;và chưa đến hết chuỗi
   )
   (setq N (1+  N))
  )                              ;Kết thúc while
  (princ)                        ;Để hàm getspace luôn trả về True
)                                ;Kết thúc GETSPACE

;Main program
  (setq TXT (getstring T "\nEnter a string: ")
        N 1
      COUNT 0
  )
  (While                ;Khi vẫn còn tìm thấy từ thì còn tiếp tuc
   (and
    (GETSPACE)          ;bỏ qua các khoảng trắng ở phía trước mêt từ
    (getword)           ;phát hiện được một từ ?
   )
   (setq COUNT (1+ COUNT))          ;Vậy thì tăng số từ tìm được lên 1
  )                                 ;Kết thúc while

  (princ (strcat "\nNumber of words is: " (itoa COUNT)))
  (princ)
)                                   ;Kết thúc defun
```

9. Chương trình vẽ đường xoắn ốc:

```
;Tên file: SPIRAL.LSP
(defun C:SPIRAL (/ NUM BLI CMD DIS DD CEN PTS ANG DA GROW)
(Setq BLI (GETVAR "BLIPMODE"))
(Setq CMD (GETVAR "CMDECHO"))
(Setvar "CMDECHO" 0)
(Initget 1)
(Setq CEN (getpoint "\nCenter point: "))
(Initget 1)
(Setq ANG (getangle CEN "\nAngle: "))
(setq NUM (getint "\n Number of rotations: "))
(setq GROW (getdist CEN "\nGrow per rotations: "))
```



```
(if (null PTS) (setq PTS 20))
 (Initget "Left right")
 (if
  (= (getkword "Direction: <Left>/Right") "Right")
  (setq DA (- (/ (* 2 PI) PTS)))
  (setq DA (/ (* 2 PI) PTS))
 )
 (Setvar "BLIPMODE" 0)
 (Setq DIS 0.0)
 (Setq DD (/ GROW PTS))
 (Command "SPLINE" CEN)
 (Repeat NUM
  (Repeat PTS
   (Command
   (polar CEN
   (setq ANG (+ ANG DA))
   (Setq DIS (+ DIS DD))
  )
 )
)
)
(Command "")
(Setvar "CMDECHO" CMD)
(Setvar "BLIPMODE" BLI)
(princ)
)
```

9.

```
;Tên file: HELIX3D.LSP
(defun radi (RADIAN)
  (/ (* 180.0 RADIAN) pi)
)
(defun C:HELIX3D ()
  (setq CMD (getvar "CMDECHO"))
  (setvar "CMDECHO" 0)
  (setq BLIP (getvar "BLIPMODE"))
  (setvar "BLIPMODE" 0)
  ;
  ;Data
  ;----
```

```
(setq PITCH (getdist "\nPitch                    = "))
(setq PITCH2 (getint "\nNumber of pitches <2>  = "))
(cond ((null PITCH2) (setq PITCH2 2)))
(setq RAD (getdist "\nRadius                   = "))
(setq PREC (getint "\nDivisions points <100> = "))
(cond ((null PREC) (setq PREC 100)))
;
;Execution
;---------
(setq TOUR 0)
(setq ANG1 (* 0.5 pi))          ;start angle
(setq PT0 (getpoint "Start point (Center line) "))
(setq X0 (car PT0) Y0 (cadr PT0) Z0 (caddr PT0))   ;* add
(setq ANG2 (/ (* 2.0 pi) PREC))
(setq DISTY (/ PITCH PREC))
(command "3dpoly" (list X0 Y0 (+ Z0 RAD)))        ;* mod
(repeat PITCH2
  (repeat PREC
    (setq TOUR (+ 1 TOUR))
    (setq ANG3 (+ (* ANG2 TOUR) ANG1))
    (setq DISTY2 (* DISTY TOUR))
    (setq DISTX (* (cos ANG3) RAD))
    (setq DISTZ (* (sin ANG3) RAD))            ;* mod
    (setq PTX (list (+ DISTX X0)
       (+ DISTY2 Y0)
    (+ DISTZ Z0)))              ;* mod
    (command PTX)
  )
)
(command "")
;
;Rotation
;--------
(setq ANG (getangle PT0 "\nRotation angle <90> = "))
(cond ((null ANG) (setq ANG (/ pi 2.0))))
(setq ANG (RADI ANG))
(setq ANG (- ANG 90.0))
(setq SS (ssget "L"))
(command "ROTATE" SS "" PT0 ANG)
;
(setvar "CMDECHO" CMD)
```



```
(setvar "BLIPMODE" BLIP)
(prin1)
);Kết thúc chương trình helix3d.lsp
```

10.

```
;Tên file: CONDUIT.LSP
(defun c:CONDUIT
  (/ ce bm om sp ep od dx an nx nu pa pb pc p1 p2 p3 p4 p5 p6 p7 p8 p9 p0
xt)
  (setq ce (getvar "CMDECHO"))
  (setq bm (getvar "BLIPMODE"))
  (setq om (getvar "OSMODE"))
  (defun ne (ne)
    (setvar "CMDECHO" ce)
    (setvar "BLIPMODE" bm)
    (setvar "OSMODE" om)
    (princ "Function cancelled ")
    (princ)
  )
  (setq oe *error* *error* ne)
  (setvar "CMDECHO" 0)
  (while (= sp nil) (setq sp (getpoint "\nPick start point of conduit: ")))
  (while (= ep nil) (setq ep (getpoint "\nPick end point of conduit: ")))
  (setq od (getdist "\nPick or enter outside diameter of conduit <0.5>: "))
  (if (= od nil)
    (setq od 0.5)
  )
  (setvar "BLIPMODE" 0)
  (setvar "OSMODE" 0)
  (setq dx (distance sp ep)
    an (angle sp ep)
    nx (/ od 2)
    nu (/ dx nx)
    pa (polar sp an (/ od 10))
    pb (polar pa an (/ od 5))
    pc (polar sp (+ an (* pi 1.5)) (/ od 2))
    p1 (polar pa (+ an (* pi 1.5)) (/ od 2.5))
    p2 (polar p1 an (/ od 10))
    p3 (polar pb (+ an (* pi 1.5)) (/ od 2))
    p4 (polar p3 an (/ od 5))
    p5 (polar p2 an (/ od 2.5))
```

```
  p9 (polar sp (+ an (/ pi 2)) (/ od 2.5))
  p6 (polar p9 an (/ od 2.5))
  p7 (polar pb (+ an (/ pi 2)) (/ od 2))
  p8 (polar p7 (+ an pi) (/ od 5))
  p0 (polar p6 an (/ od 10))
)
(command "LINE" sp p1 pc p9 "")
(repeat (fix nu)
  (command ".LINE" p1 p2 p3 p4 p5 p6 p7 p8 p9 p2 "" ".LINE" p6 p0 "")
  (setq sp (polar sp an nx)
    p1 (polar p1 an nx)
    p2 (polar p2 an nx)
    p3 (polar p3 an nx)
    p4 (polar p4 an nx)
    p5 (polar p5 an nx)
    p6 (polar p6 an nx)
    p7 (polar p7 an nx)
    p8 (polar p8 an nx)
    p9 (polar p9 an nx)
    p0 (polar p0 an nx)
  )
)
(setq xt (distance sp ep))
(if (>= xt (/ od 5))
  (command ".LINE" p6 p7 p8 p9 p2 p1 "" ".LINE" p2 p3 p4 "")
)
(setvar "OSMODE" om)
(setvar "CMDECHO" ce)
(setvar "BLIPMODE" bm)
(setq *error* oe oe nil)
(princ)
); Kết thúc chương trình CONDUIT.LSP
```

Chương 10

XỬ LÝ DANH SÁCH (NÂNG CAO)

Nội dung chương

1. Tạo, sửa và xử lý danh sách bằng các hàm nâng cao: **Assoc**, **Cons**, **Member**, **Reverse**, **NTN**.
2. Sử dụng các tham số kiểu danh sách cho các hàm có tham số không phải kiểu danh sách: **Apply**, **Mapcar**.
3. Các hàm không tên: **Lambda**.

Khi viết chương trình với khối lượng dữ liệu lớn, ta nên sử dụng danh sách để lưu trữ dữ liệu. Do đó, ta phải sử dụng thành thạo các hàm xử lý dữ liệu.

10.1 Các hàm xử lý danh sách nâng cao

Các hàm xử lý dữ liệu đã trình bày ở các chương trước:

(**List** EXPR ...)	Tạo danh sách định giá trị.
(**Quote** EXPR)	Tạo danh sách không định giá trị.
(**CxxxxR** LIST)	Các hàm truy xuất phần tử trong danh sách.
(**Last** LIST)	Truy xuất phần tử cuối cùng trong danh sách.
(**Length** LIST)	Trả về số lượng các phần tử trong danh sách.
(**Append** EXPR ...)	Thêm các phần tử vào vị trí cuối cùng trong một danh sách.

Trong mục này ta sẽ khảo sát thêm các hàm:**Assoc**, **Cons**, **Member**, **Reverse**, **NTN**...

Phân loại danh sách kho dữ liệu (data storage list):

- *Danh sách đơn (simple data list)*: Chứa các dữ liệu đơn. Thứ tự các phần tử trong danh sách không quan trọng.
- *Danh sách phức hợp (association list)*: Chứa các danh sách con (*sublist*). Phần tử đầu tiên của danh sách con gọi là *header*.

Hàm ASSOC

Hàm **Assoc** *(ASSOCiation)* trả về một danh sách con trong danh sách phức hợp.

(**Assoc** ITEM ALIST)

Tham số ALIST phải là danh sách phức hợp. Tham số ITEM là phần tử đầu tiên (*header*) của danh sách con trả về. Nếu không tìm thấy danh sách con nào chứa phần tử đầu tiên là ITEM, hàm **Assoc** sẽ trả về giá trị nil.

✌ **Ví dụ:**

```
(setq ALIST '((1 "ONE" "UNIX") (2 "TWO" "DOS") (3 "THREE" "LINUX") ))
                ;1 là header của danh sách con (1 "ONE" "UNIX")
                ;2 là header của danh sách con (2 "TWO" "DOS")
                ;3 là header của danh sách con (3 "THREE" "LINUX")
(assoc 1 ALIST)      ;trả về (1 "ONE" "UNIX")
(assoc 2 ALIST)      ;trả về (2 "TWO" "DOS")
(assoc 3 ALIST)      ;trả về (3 "THREE" "LINUX")
(assoc 4 ALIST)      ;trả về nil
```


Ta dùng hàm **Cdr** để loại bỏ *header* khỏi danh sách con, lấy ra thông tin cần thiết.

✌ **Ví dụ:**

```
(cdr (assoc 1 ALIST))      trả về ("ONE" "UNIX")
(cdr (assoc 2 ALIST))      trả về ("TWO" "DOS")
(cdr (assoc 3 ALIST))      trả về ("THREE" "LINUX")
```

Hàm **Assoc** là một phương pháp hiệu quả để truy xuất thông tin lưu trữ.

✌ **Ví dụ:**

Danh sách FORM chứa chiều dài, chiều rộng, chiều cao và thể tích một khối chữ nhật.

```
(setq FORM '((length 185.0) (width 75.0) (height 45.0) (volume 624375.0)))
(cadr (assoc 'length FORM))      trả về 185.0
(cadr (assoc 'width FORM))       trả về 75.0
(cadr (assoc 'height FORM))      trả về 45.0
(cadr (assoc 'volume FORM))      trả về 624375.0
```

Hàm CONS

Các danh sách con trong danh sách phức hợp có các dạng khác nhau và có thể chứa nhiều phần tử. Một dạng thường gặp của danh sách con là *dotted pair*. Đây là một danh sách chứa hai phần tử phân cách nhau bằng dấu chấm. **AutoLISP** thường dùng dạng danh sách này để lưu trữ và truy xuất dữ liệu lấy từ cơ sở dữ liệu các đối tượng của **AutoCAD**.

Hàm **Cons** (*CONStruct*) có hai tham số. Nếu tham số thứ hai là một danh sách, hàm này sẽ bổ sung tham số thứ nhất vào vị trí đầu tiên trong danh sách này.

(**Cons** NEW-FIRST-ELEMENT LIST)

✌ **Ví dụ:**

```
(setq a 100)
(cons 10 '(50 40 60))                  trả về (10 50 40 60)
(cons 'a '(b c))                       trả về (a b c)
(cons a '(b c))                        trả về (100 b c)
(cons '("RED") '("BLUE" "GREEN"))      trả về ("RED" "BLUE" "GREEN")
```

```
(cons "RED '("BLUE" "GREEN"))      trả về ("RED" "BLUE" "GREEN")
(cons 100 '("BLUE" "GREEN"))       trả về (100 "BLUE" "GREEN")
```

Nếu tham số thứ hai là một nguyên tố (bất kỳ kiểu dữ liệu nào không phải là danh sách), hàm này sẽ trả về danh sách dạng *dotted pair*.

Ví dụ:

```
(setq A 100 B "ONE HUNDRED")
(cons 1 "ONE")                     trả về (1 . "ONE")
(cons A B)                         trả về (100 . "ONE HUNDRED")
(cons '(100 200) A)                trả về ((100 200) . 100)
```

Thông thường ta dùng hàm **Cons** để tạo ra danh sách con chứa một *header* và một giá trị.

Ví dụ:

```
(setq AL (list (cons 'COLOR 4) (cons 'LAYER "0")
               (cons 'PT1 '(1.0 1.0 0.0))))
```

Kết quả trả về là danh sách:

```
((COLOR . 4) (LAYER . "0") (PT1 1.0 1.0 0.0))
```

Trích dữ liệu từ danh sách AL:

```
(cdr (assoc 'COLOR AL))            trả về 4
(cdr (assoc 'LAYER AL))            trả về "0"
(cdr (assoc 'PT1 AL))              trả về (1.0 1.0 0.0)
```

Hàm MEMBER

Hàm **Member** duyệt danh sách LIST để tìm xem EXPR có là phần tử của danh sách này hay không. Nếu tìm thấy, kết quả trả về là phần còn lại của danh sách LIST, bắt đầu từ phần tử EXPR.

(**Member** EXPR LIST)

Ví dụ:

```
(setq L '(1 2 3 4 5 6))
(member 2 L)                       trả về (2 3 4 5 6)
(member 3 L)                       trả về (3 4 5 6)
(member 6 L)                       trả về (6)
(member 7 L)                       trả về nil
```

Trong các chương trước, ta đã biết cách dùng các biểu thức điều kiện **If**, **Cond** để cung cấp các lựa chọn cho người sử dụng. Tuy nhiên nếu

cần có nhiều lựa chọn, thì cần phải có nhiều biểu thức điều kiện. Ta có thể dùng phối hợp các hàm **Member**, **Assoc**, **CxxxxR** để giảm bớt các biểu thức điều kiện.

Ví dụ:

Chương trình thiết lập bản vẽ. Giả sử đã có các khối hoặc bản vẽ chứa khung tên cần chèn vào khi thiết lập bản vẽ là "TITLE-A0", "TITLE-A1", "TITLE-A2", "TITLE-A3", "TITLE-A4".

```
;Tên file: SETUP.LSP
;Chương trình này sử dụng các tham số chứa trong tham số phức hợp để
;thiết lập bản vẽ
;
;Nhập dữ liệu
;
;Sử dụng hàm Member để kiểm tra dữ liệu nhập có hợp lệ hay không.
;
(setq OPT (strcase (getstring "\nPaper size for drawing
                              A0/A1/A2/A3/A4>: ")))
(while
  (not (member OPT '("A0" "A1" "A2" "A3" "A4")))  ;Nếu OPT không có
                                                  ;trong danh sách
  (princ "\nInvalid paper size. Try again ...")   ;thì hiện thông báo và
                                                  ;yêu cầu nhập lại
  (setq OPT (strcase (getstring "\n Paper size for drawing
                                <A0/A1/A2/A3/A4>: ")))
)                                                 ;Đóng WHILE
;
;Tạo danh sách phức hợp chứa các tham số
;
;
  (setq ALIST '(                                  ;Tao danh sách phức hợp
                ("A0" 2376 1188 "TITLE-A0")
                ("A1" 1188 594 "TITLE-A1")
                ("A2" 594 420 "TITLE-A2")
                ("A3" 420 297 "TITLE-A3")
                ("A4" 297 210 "TITLE-A4")
               )                                  ;Đóng QUOTE
  )                                               ;Đóng SETQ
;
;Thiết lập bản vẽ
```

```
;Sử dung các tham số trong danh sách tương ứng với khổ giấy do người sử
dung nhập vào để thiết lập bản vẽ.
;
(setq PARALIST (assoc OPT ALIST))          ;Lấy giá trị tương ứng với OPT
                                           ;gán cho biến PARALIST
(setvar "LIMMIN" '(0 0))                   ;Góc trái dưới tại điểm 0,0
(setvar "LIMMAX" (list (cadr PARALIST)     ;Phần tử thứ hai, thứ ba trong
                       (caddr PARALIST))   ;PARALIST là góc phải trên
)
(command "insert" (cadddr PARALIST)
                  '(0 0) 1 1 0)            ;Phần tử thứ tư của PARALIST
(command "zoom" "A")                       ;Zoom All
); Kết thúc chương trình SETUP.LSP
```

Trong chương trình trên, nếu người sử dụng nhập vào khổ bản vẽ không có trong danh sách OPT, hàm **Member** sẽ trả về nil và vòng lặp **While** sẽ lặp lại. Sau đó hàm **Assoc** sẽ lấy ra các giá trị tương ứng với khổ giấy vẽ được chọn trong danh sách ALIST. Các giá trị nay được dùng để thiết lập bản vẽ. Việc sử dụng danh sách để lưu trữ dữ liệu làm giảm bớt các đoạn chương trình lặp đi lặp lại, giúp cho việc quản lý dữ liệu dễ dàng hơn.

Hàm REVERSE

Hàm **Reverse** trả về danh sách theo thứ tự ngược lại.

(**Reverse** LIST)

✌ **Ví dụ:**

```
(setq L1 '(A B C D E) L2 (list (cons "LENGTH" 40)(cons "WIDTH" 25)))
(reverse L1)                    trả về (E D C B A)
(reverse L2)                    trả về (("WIDTH" . 25)("LENGTH" . 40))
```

✌ **Ví dụ:**

Loại bỏ một phần tử khỏi danh sách bằng các hàm **Member**, **Reverse**

```
;Tên file REMFMLST.LST
;Mục đích: Loại bó một phần tử khỏi danh sách.
;
(defun RFL (I L / L1 L2)
  (setq L1                                  ; [5]
   (reverse                                 ; [4]
     (cdr                                   ; [3]
       (member I                            ; [2]
          (reverse L)                       ; [1]
       )
     )
   )
  )
  (setq L2                                  ; [8]
     (cdr                                   ; [7]
       (member I L)                         ; [6]
     )
   )
   (append L1 L2)                           ; [9]
)
```


Xét danh sách LST như sau:

Command:**(setq LST '(A B C D E F G H)**↵

(A B C D E F G H)

Command: **(setq LST (RFL 'E LST)** ↵

(A B C D F G H) ;loại bỏ phần tử 'E khỏi danh sách LST

Giải thích:

[1] **(reverse L)** Đảo ngược danh sách LST. Kết quả: (H G F E D C B A)

[2] **(member I** Tìm các phần tử sau phần tử I trong danh sách. Kết quả: (E D C B A)

[3] **(cdr** Loại bỏ phần tử đầu tiên là E. Kết quả: (D C B A)

[4] **(reverse** Đảo ngược danh sách để trả lại thứ tự ban đầu. Kết quả: (A B C D)

[5] **(setq L1** Lưu trữ danh sách trên vào biến L1.

[6] **(member I L)** Bắt đầu tao phần thứ hai của danh sách. Kết quả: (E F G H)

[7] **(cdr** Loại bỏ phần tử đầu tiên là E. Kết quả: (F G H)

[8] **(setq L2** Lưu trữ danh sách trên vào biến L2

[9] **(append L1 L2)** Kết nối hai danh sách tạo ra danh sách kết quả. Đây là biểu thức cuối cùng nên hàm RFL cũng trả về kết quả nay. Kết quả: (A B C D F G H)

Đây là một hàm rất có ích cho nhiều chương trình. Có thể viết lại ngắn gọn như sau:

```
(defun RFL (I L)
   (append (reverse (cdr (member I (reverse L))))(cdr (member I L)))
)
```

Hàm NTH

Các hàm **CxxxxR** chỉ dùng được với các danh sách có ít phần tử. Với các danh sách có nhiều phần tử, ta có thể dùng hàm **Nth**.

Hàm **Nth** cho phép lấy ra phần tử ở vị trí N trong danh sách LIST.

(**Nth** N LIST)

Chú ý:

Vị trí các phần tử trong danh sách được đánh số bắt đầu từ 0.
- Phần tử thứ 1 có vị trí 0.
- Phần tử thứ 2 có vị trí 1.
- Phần tử thứ 3 có vị trí 2...

Ví dụ:

```
(setq LST '(1EP0 2EP1 3EP2 4EP3 5EP4 6EP7)
(nth 0 LST)          trả về 1EP0    (phần tử 1, vị trí 0)
(nth 3 LST)          trả về 4EP3    (phần tử 4, vị trí 3)
(nth 5 LST)          trả về 6EP5    (phần tử 6, vị trí 5)
```

Ví dụ:

Chương trình thiết lập không gian giấy vẽ và tạo các cửa sổ động.

```
;Tên file: PAPER.LSP
;Mục đích:  Thiết lập không gian giấy vẽ (Paper space). Gán các giá trị snap
;và grid,
;Chọn khổ giấy vẽ A4, A3, A2, số lượng các cửa sổ động
;
(defun C:PAPER (/ SIZ ALIST T_R RES ALI VIEWPORTS CTR SCA)
  (setvar "cmdecho" 0)
  (setvar "tilemode" 0)                    ;Chuyển sang không gian giấy vẽ
  (princ "\nSnap setting: ") (command ".snap" pause);Gán giá trị snap
  (princ "\nGrid setting: ") (command ".grid" pause) ;Gán giá trị grid
```



```
;Danh sách kích thước bản vẽ - A4, A3, A2
  (initget 1 "A4 A3 A2")
  (setq SIZ   (getkword "\nPaper Space drawing size – A4/A3/A2: ")
        ALIST '(("A4" 297 210) ("A3" 420 297) ("A2" 594 420))
        T_R   (cdr (assoc SIZ ALIST))                              ; [1]
  )
  (command ".limits" '(0 0) T_R
           ".zoom" "A"
           ".layer" "m" "pslayout" ""        ;Tạo lớp bản vẽ cho không
                                             ;gian giấy vẽ
           ".mview"                          ;Lệnh Mview ...
  )
  (initget 1 "2 3 4")
  (setq RES (getkword "\nMview creation – 2/3/4 viewports: "))     ; [2]
  (cond
    ((= "2" RES)
    (initget "Horizontal Vertical")
    (setq ALI (getkword "\nHorizontal/<Vertical>: ")               ; [3]
          ALI (if (null ALI) "Vertical" ALI)                       ; [4]
    )
    (command RES ALI '(0 0) T_R)                                   ; [5]
    )
    ;
    ((= "3" RES)
    (initget "Horizontal Vertical Above Below Left Right")
    (setq ALI (getkword "\nHorizontal/Vertical/Above/
                         Below/Left/<Right>: ")                    ; [6]
          ALI (if (null ALI) "Right" ALI)                          ; [7]
    )
    (command RES ALI '(0 0) T_R)                                   ; [8]
    )
    (T (command RES '(0 0) T_R))
    ;
  )                                                    ;Đóng COND
  (command ".mspace")
  (setq VP   (vports)                                              ; [9]
        VP  (rfl (assoc 1 VP) VP)                                  ; [10]
        CTR 0                              ; gán biến đếm CTR bằng 0
  )
  (repeat (atoi RES)                    ;Duyệt từng không gian giấy vẽ
    (setvar "cvport" (car (nth CTR VP)))                           ; [11]
```

```
    (princ "\nViewport selected ...Zoom window on detail:")
    (command ".zoom" "w" pause pause)                          ; [12]
    (initget 1 "2 4 8")
    (setq SCA (getkword "\nScaling [i.e. 2 = 1/2 scale] - 2/4/8: ")
          SCA (/ 1.0 (atoi SCA))                               ; [13]
    )
    (command ".zoom" (strcat (rtos SCA 2 8) "xp"))             ; [14]
    (setq CTR (1+ CTR))
  )
  (command ".pspace")
  (princ)
);Kết thúc chương trình PAPER.LSP
```

Giải thích:

[1] **(setq T_R (cdr (assoc SIZ ALIST)))** Hàm **Assoc** lấy ra danh sách con tương ứng, với khổ giấy SIZ là header. Hàm **Cdr** lấy ra giới hạn bản vẽ và nó được gán cho biến T_R.

[2] **(setq RES (getkword "\nMview creation – 2/3/4 viewports: "))** Biến RES chứa số lượng các cửa sổ động sẽ được lệnh **Mview** tạo ra.

[3] **(setq ALI (getkword "\nHorizontal/<Vertical>: ")** Có hai cách sắp xếp hai cửa sổ động là Horizontal và Vertical.

[4] **(setq ALI (if (null ALI) "Vertical" ALI))** Khi người sử dụng nhấn ENTER chấp nhận giá trị mặc định <Vertical>, tức là điều kiện (null ALI) đúng, biến ALI sẽ được gán giá trị "Vertical". Ngược lại, giá trị biến ALI giữ nguyên.

[5] **(command RES ALI '(0 0) T_R)** Thực hiện tiếp tục lệnh **Mview**. RES chứa số lượng cửa sổ động. ALI chứa cách sắp xếp cửa sổ. '(0 0) là tọa độ góc trái dưới. T_R chứa góc trên phải.

[6] **(setq ALI (getkword "\nHorizontal/Vertical/Above/Below/Left/<Right>: "))** Có sáu cách sắp xếp ba cửa sổ động.

[7] **(setq ALI (if (null ALI) "Right" ALI))** Khi người sử dụng nhấn ENTER chấp nhận giá trị mặc định <Right>, tức là điều kiện (null ALI) đúng, biến ALI sẽ được gán giá trị "Right". Ngược lại, giá trị biến ALI giữ nguyên.

[8] **(command RES ALI '(0 0) T_R)** Thực hiện tiếp tục lệnh **Mview**. RES chứa số lượng cửa sổ động. ALI chứa cách sắp xếp cửa sổ. '(0 0) là tọa độ góc trái dưới. T_R chứa góc trên phải.

[9] **(setq VP (vports))** Hàm **Vports** tạo ra danh sách phức hợp, chứa tất cả các cửa sổ động trong không gian giấy vẽ. Biến VIEWPORTS chứa danh sách này.

[10] **(setq VP (rfl (assoc 1 VP) VP))** Trong danh sách phức hợp do hàm **Vports** tạo ra, danh sách con thứ nhất chứa dữ liệu về màn hình. Ta cần loại bỏ danh sách con này bằng hàm **Rfl**. Danh sách này có *header* là 1. Ghi chú: Hàm **Rfl** đã được tải vào chương trình sẵn.

[11] **(setvar "cvport" (car (nth CTR VP)))** Biến hệ thống CVPORT lưu trữ số thứ tự của cửa sổ hiện hành. Hàm **Nth** lấy ra danh sách con trong danh sách VP. Hàm **Car** lấy ra số thứ tự của cửa sổ động. Cửa sổ này được gán cho biến hệ thống CVPORT và trở thành cửa sổ hiện hành.

[12] **(command ".zoom" "w" pause pause)** Cho phép người sử dụng thực hiện lệnh **Zoom** để định vị hình ảnh trên bản vẽ. Sau đó lệnh **Zoom** thứ hai sẽ phóng theo tỉ lệ chính xác.

[13] **(setq SCA (/ 1.0 (atoi SCA)))** Hệ số tỉ lệ của lệnh **Zoom** thứ hai.

[14] **(command ".zoom" (strcat (rtos SCA 2 8) "xp"))** Chuyển số thực lưu trong biến SCA thành chuỗi với kiểu đơn vị hệ thập phân, 8 số lẻ. Chuỗi này được kết nối với "xp" để tạo ra tỉ lệ **Zoom** tương ứng với không gian giấy vẽ.

Tóm tắt:

1. Danh sách phức hợp là phương tiện lưu trữ và truy xuất dữ liệu đơn giản và hiệu quả.
2. Hàm **Assoc** truy tìm danh sách con khi biết phần tử đầu tiên của danh sách con. Hàm này trả về toàn bộ danh sách con.
3. *Dotted pair* là một dạng đặc biệt của danh sách, được tạo bởi hàm **Cons**.
4. Hàm **Member** truy tìm một giá trị có trong danh sách hay không. Nếu tìm thấy sẽ trả về danh sách bắt đầu từ phần tử này cho đến hết danh sách.
5. Hàm **Reverse** đảo ngược một danh sách theo thứ tự ngược lại.
6. Hàm **Nth** thích hợp với các danh sách chứa nhiều phần tử hơn các hàm dạng **CxxxxR**.

10.2 Sử dụng các tham số kiểu danh sách cho các hàm có tham số không phải kiểu danh sách

Xét bài toán tính tổng nhiều số. Ta có thể dùng vòng lặp, nhập vào lần lượt các số và cộng dồn các số này vào một biến.

```
(setq Sum 0)
(Repeat 10
    (setq Sum (+ Sum (getreal "\nEnter a number: ")))
)
```

Một phương pháp hiệu quả hơn là ghép các số thành một danh sách. Sau đó thực hiện phép cộng các phần tử của danh sách này. Một trở ngại ở đây là hàm công chỉ chấp nhận các tham số kiểu chuỗi, chứ không chấp nhận tham số kiểu danh sách.

Hàm APPLY

Hàm **Apply** giúp ta tạo ra một danh sách LIST chứa các tham số cần cung cấp cho hàm FUNCTION.

(**Apply** FUNCTION LIST)

✌ **Ví dụ:**

(apply 'strcat (list "H" "e" "l" "l" "o"))

tương đương với :

(strcat ("H" "e" "l" "l" "o")) trả về "Hello"

Ghi chú:

- Tên hàm phải có dấu ' ở phía trước ('strcat) .
- Các phần tử trong danh sách phải phù hợp với hàm.

✌ **Ví dụ:**

(apply 'princ '("A" "B" "C")) trả về lỗi vì chỉ cần 1 tham số chứ không phải 3

(apply 'princ '("A") trả về "A"

(apply '+ (list 1 2 3 4 5)) trả về 15

✌ **Ví dụ:**

Chương trình tính tổng các số nhập vào.

```
;Tên file: ADDEM.LSP
(defun C:ADDEM (/ NEXTNUM NUMLIST)
(while (setq NEXTNUM (getreal "\nInput number or [ENTER] when done:
       "))
   (setq NUMLIST (append NUMLIST (list NEXTNUM)))
)
(apply '+ NUMLIST)
)
```

Đối với các hàm có số lượng các tham số cố định (như hàm **Princ**), ta thường dùng phương pháp cũ. Dùng ham **Apply** có hiệu quả khi số lượng tham số thay đổi như hàm cộng trong ví dụ trên.

Hàm MAPCAR

Hàm **Mapcar** cho phép thực hiện hàm FUNCTION nhiều lần, mỗi lần sử dụng một danh sách các tham số khác nhau.

(**Mapcar** FUNCTION LIST1 ... LISTN)

Hàm **Mapcar** tạo ra một vòng lặp. Lần đầu tiên, hàm FUNCTION sẽ sử dụng tham số là các phần tử đầu tiên của các danh sách LIST. Lần thứ hai, hàm FUNCTION sử dụng tham số là các phần tử thứ hai của các danh sách LIST... Tất cả các kết quả trả về sẽ được hàm **Mapcar** kết nối thành một danh sách kết quả.

✌ **Ví dụ:**

Command: **(mapcar '+ (list 1 2 3)(list 4 5 6))** ↵

(5 7 9) ;(1+4), (2+5) và (3+6)

Command: **(mapcar 'strcat '("A" "X") '("B" "Y") '("C" "Z"))**↵

("ABC" "XYZ")

Command: **(mapcar '1+ (list 1 2 3 4 5 6))**↵

(2 3 4 5 6 7)

Command: **(mapcar 'setvar '("BLIPMODE" "CMDECHO" "EXPERT")**
'(0 0 1)) ↵

(0 0 1) ;gán giá trị các biến hệ hống BLIPMODE = 0
;CMDECHO = 0 và EXPERT = 1

Command: **(apply 'strcat (mapcar 'substr '("HOME" "END" "SCROLL" "LOCK") '(1 1 5 2) '(1 1 2 1)))** ↵

```
"HELLO"          ;Các chuỗi con trong các chuỗi lần lượt được hàm
                 ;Substr tách ra, sau đó được hàm Strcat ghép lại
                 ;thành chuỗi "HELLO".
```

Các hàm không tên

Các hàm **Apply** và **Mapcar** đòi hỏi phải chỉ rõ tên hàm sử dụng với các danh sách. Các hàm này có thể là hàm **AutoLISP** nhưng phần lớn là các hàm tự tạo. Tuy nhiên, nếu dùng hàm **Defun** để tạo các hàm chỉ dùng một lần duy nhất, thì các hàm này vẫn tiếp tục chiếm bộ nhớ trong khi chúng không cần thiết nữa. Ngoài ra, số lượng các hàm tự tạo tăng lên, gây khó khăn trong việc quản lý chúng. Thay vào đó, ta có thể dùng hàm **Lambda** để tạo các hàm dùng một lần duy nhất ngay tại vị trí cần thiết.

(**Lambda** ARGUMENTS EXPR ...)

Các hàm do **Lambda** tạo ra không có tên nên được gọi là các *hàm không tên.*

Ví dụ:

Trong biểu thức sau, yêu cầu vẽ một đường thẳng đi qua hai điểm, sau đó gán màu cho đường thẳng này. Ta dùng hàm **Lambda** để định nghĩa một hàm thực hiện các công việc này. Sau đó cung cấp cho nó tọa độ hai điểm.

```
(
  (lambda (P1 P2 / CL)          ;định nghĩa hàm không tên
    (command ".line" P1 P2 "")
    (setq CL (getint "\nColor number : "))
    (command ".chprop" "L" "" "C" CL "")
  )                             ;Đóng LAMBDA
  '(2 2) '(8 2)                 ;toa độ hai điểm cung cấp cho hai tham số P1, P2
)
```

Nếu dùng hàm **Defun**, thì biểu thức có dạng như sau:

```
(defun COLIN (P1 P2 / CL)
    (command ".line" P1 P2 "")
    (setq CL (getint "\nColor number : "))
    (command ".chprop" "L" "" "C" CL "")
)
```


Hàm này được gọi thi hành như sau: (COLIN P1 P2)

Hàm **Lambda** đặc biệt có ích khi dùng với các hàm **Apply** và **Mapcar**.

Ví dụ:

Trong ví dụ sau có chứa ba danh sách. Danh sách thứ nhất chứa tên các lớp bản vẽ, danh sách thứ hai chứa tên các dạng đường, danh sách thứ ba chứa tên các màu. Hàm **Mapcar** sẽ truy xuất các danh sách này. Và hàm **Lambda** sẽ lần lượt tạo các lớp bản vẽ mới, gán dạng đường và màu sắc.

```
(setq LNLST '("HID" "CEN" "PHAN" "DIM" "OBJ")
      LTLST '("HIDDEN" "CENTER" "PHANTOM" "CONTINUOUS"
              "CONTINUOUS")
      COLST '("CYAN" "YELLOW" "GREEN" "RED" "WHITE")
)
;
(mapcar
  '(lambda (LN LT CO)           ;LAMBDA dùng làm tên hàm cho MAPCAR
                                ;Biểu thức LAMBDA phải có dấu ' phía trước
    (command ".layer" "new" LN "ltype" LT LN "color" CO LN "")
  )                             ;end LAMBDA
  LNLST                         ;tham số thứ nhất của MAPCAR
  LTLST                         ;tham số thứ hai của MAPCAR
  COLST                         ;tham số thứ ba của MAPCAR
)
```

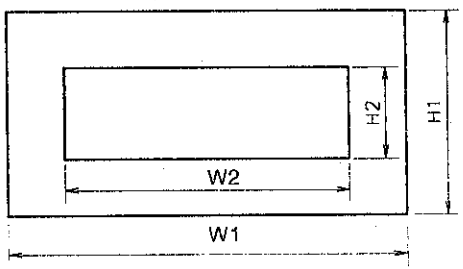
Tóm tắt:

1. Ta nên dùng danh sách để lưu trữ nhiều dữ liệu liên quan với nhau.
2. Danh sách phức hợp giúp ta truy xuất dữ liệu dễ dàng. Nhiều phần tử danh sách có thể được truy xuất nhờ vào một từ khoa chung, chính là *header* của danh sách con. Ngoài ra, đây còn là cách thức **AutoLISP** biểu diễn các dữ liệu lấy từ bản vẽ **AutoCAD**.
3. Các hàm **Apply** và **Mapcar** cho phép sử dụng danh sách làm tham số cho các hàm chỉ chấp nhận tham số ở các kiểu bình thường.

4. Hàm **Lambda** dùng để tạo ra các hàm không tên chỉ sử dụng một lần. Hàm này đặc biệt có ích khi dùng chung với các hàm **Apply** và **Mapcar**.

10.3 Ví dụ mẫu

Ví dụ sau đây sử dụng nhiều khái niệm và các hàm trình bày trong chương này. Chương trình dùng để tính khối lượng (kg) và moment quán tính (kg/cm^3) của một trục rỗng hình chữ nhật được làm bằng những vật liệu khác nhau.




```
;Tên file:   SQTUBE.LSP
;
(defun C:PARA (/ H1 W1 H2 W2 LEN ANS MATS)
  (setvar "cmdecho" 0)
  (setvar "gridmode" 0)

;Tao các thông tin tham số cho tính toán trục chữ nhật rỗng
;
  (mapcar
    '(lambda (PARMS DIMS)
      (initget (+ 1 2 4))
      (set PARMS (getreal DIMS))
    )                                   ;Đóng LAMBDA
    (list 'H1 'W1 'H2 'W2 'LEN)         ;Danh sách tham số thứ nhất cho
                                        ;MAPCAR – tên ký hiệu PARMS cho
                                        ;LAMBD
    '("\nHeight H1 (cm): " "\nWidth W1 (cm): ";Danh sách tham số thứ hai
                                        ;cho MAPCAR – dòng nhắc DIMS
      "\nHeight H2 (cm): " "\nWidth W2 (cm): ";cho LAMBDA
      "\nLength of Beam in cm: "
    )
  )                                     ;Đóng MAPCAR
```



```
;Dựng hình 3D trên màn hình đồ hoa
;
  (defun lp()                           ;Sử dụng hàm truy luc điểm cuối-
                                        ;xếp lồng hàm  DEFUN ở trên
    (getvar "lastpoint")
  )                                     ;Kết thúc hàm xếp lồng DEFUN
  (setq PT1 '(0 0 0)
        PT2 (list (/ (- W1 W2) 2) (/ (- H1 H2) 2) 0)
  )                                     ;Đóng SETQ
  (command ".ucs" 3 "*0,0,0" "*0,1,0"  "*0,0,1"
           ".vpoint" "-1,-1,1"
           ".ucsicon" "or"
           ".thickness" LEN
           ".pline" PT1 (polar PT1 0 W1)
                        (polar (lp) (* pi 0.5) H1)
                        (polar (lp) pi W1)  "c"
           ".pline" PT2 (polar PT2  0 W2)
                        (polar (lp) (* pi 0.5) H2)
                        (polar (lp) pi W2)  "c"
           ".zoom" "e"
           ".hide"
  )                                     ;Đóng  COMMAND
;In các kết quả thống kê lên màn hình khối lượng và mô ment quán tính
;
  (mapcar
    '(lambda (MAT WTS)
      (princ (strcat "\nThe weight for "  MAT  "is: "))
      (princ (* (- (* W1 H1)(* W2 H2)) LEN WTS))
      (princ " kg.")
    )                                   ;Đóng LAMBDA
    '("Steel" "Cooper" "Brass" "Alum" "Iron"); Danh sách tham số thứ nhất
                                        ;MAPCAR, MAT cho LAMBDA
    '(0.00783 0.00889 0.00870 0.00270 0.00790);Danh sách tham số thứ hai
                                        ;MAPCAR WTS cho LAMBDA
  )                                     ;Đóng MAPCAR
;
  (princ "\n\n   The Moment of inetia is: ")
  (princ (/ (- (* W1 (expt H1 3)) (* W2 (expt H2 3))) 12))
  (princ " kg/cm^3.")

;
(initget 1 "Steel Copper Brass Alum Iron")
```

```
  (setq ANS (getkword "\n\nSelect one of the above materials: ")
        MATS '(("Steel" "ansi32") ("Cooper" "ansi33")("Brass" "ansi33")
               ("Alum" "ansi38") ("Iron" "ansi31"))
  )
  (command ".thickness" 0
           ".hatch" (cadr (assoc ANS MATS)) "3" "0" PT1 PT2 ""
  )
  (setvar "cmdecho" 1)
  (princ)
)
;Kết thúc chương trình SQTUBE.LSP
```

10.4 Bài tập

1. Dựa vào các ví dụ trong chương này, viết chương trình DATALIST.LSP có một biến toàn cục tên là DATALIST. Chương trình cung cấp cho người sử dụng các lựa chọn sau:

a. Hiện các dòng nhắc nhập giá trị HEADER và VALUE. Hai giá trị này xem như là một *dotted pair*. Sau đó chèn *dotted pair* này vào DATALIST.

b. Hiển thị DATALIST.

c. Nhập vào giá trị của HEADER. Sau đó loại bỏ *dotted pair* tương ứng với HEADER này ra khỏi DATALIST.

d. Nhập vào giá trị của HEADER. Lấy ra giá trị VALUE tương ứng.

2. Viết chương trình REPLACE.LSP thay thế một phần tử trong một danh sách có sẵn bằng một giá trị mới do người sử dụng nhập vào. Không được dùng hàm **Substr**. Thay vào đó, hãy sử dụng phương pháp trong chương trình REMFMLT.LSP phối hợp với hàm **Cons**.

3. Viết chương trình LISTPOS.LSP yêu cầu người sử dụng nhập vào tên một danh sách có sẵn. Sau đó yêu cầu nhập vào một vị trí trong danh sách, và chương trình sẽ trả về phần tử tại vị trí đó. Nên dùng vòng lặp cho phép người sử dụng nhập vị trí nhiều lần cho đến khi yêu cầu kết thúc bằng lựa chọn Exit.

4. Viết chương trình SET-TOL.LSP thực hiện các việc sau:

a. Các lựa chọn cho phép tạo hoặc điều chỉnh danh sách giá trị dung sai. Đây là danh sách phức hợp có 3 phần tử: *header* là các ký tự ABC, tiếp theo là các sai lệch dương và sai lệch âm của dung sai. Ví dụ: ("A" 0.005 0.002)

b. Các lựa chọn cho phép hiển thị các dung sai đã gán và có thể điều chỉnh khi cần thiết.

c. Các lựa chọn cho phép chọn một dung sai, sau đó gán giá trị cho các biến kích thước tương ứng. Ví dụ, với ("A" 0.005 0.002), ta chọn "A" và các biến DIMTP, DIMTM sẽ được gán bằng 0.005, 0.002.

10.5 Lời giải

1.

```
;Tên file: DATALIST.LSP
;
(defun C:DLIST (/ DATALIST OPT)
;
(defun RFL (I L)
  (append (reverse (cdr (member I (reverse L))))(cdr (member I L)))
)

(setq DATALIST '())
;
(While
 (or
  (initget 1 "Exit Insert Remove Display All")
  (/= "Exit"
   (setq OPT
     (getkword "\nExit/Insert/Remove/Display/All: ")
   )
  )
 )                          ;Đóng OR

;
 (cond
  ((= OPT "Insert")
   (setq H (getstring T "\nHeader: ")
         V (getstring T "\nValue: ")
       DATALIST (append DATALIST (list (cons H V)))
   )                        ;Đóng SETQ
  )                         ;Đóng Insert
     ;
  ((= OPT "Remove")
   (setq H (getstring T "\nHeader: ")
```

```
        R (assoc H DATALIST)
        DATALIST (rfl R DATALIST)
      )
      (princ R) (princ " has removed")
    )                    ;Đóng Remove

    ((= OPT "Display")
      (setq H (getstring T "\nHeader: ")
          R (assoc H DATALIST)
      )
      (princ R)
    )                    ;Đóng Display

    ((= OPT "All")
    (princ DATALIST)
    )                    ;Đóng All

)                        ;Đóng COND

)                        ;Đóng while
(princ)
;
)                        ;Đóng defun
```

2.

```
;Tên file: REPLACE.LSP

(defun Replace (I NEW L)
  (append (reverse (cdr (member I (reverse L))))
          (list NEW) (cdr (member I L)))
)
```

3.

```
;Tên file: LISTPOS.LSP

(setq LNLST '("HID" "CEN" "PHAN" "DIM" "OBJ")
      LTLST '("HIDDEN" "CENTER" "PHANTOM" "CONTINUOUS"
              "CONTINUOUS")
      COLST '("CYAN" "YELLOW" "GREEN" "RED" "WHITE")
)
```



```
(initget 1 "LAyer LInetype Color")
(setq ANS (getkword "\nLAyer/LInetype/Color: "))
(cond
  ((= ANS "LAyer")
   (setq LST LNLST)
  )
  ((= ANS "LInetype")
   (setq LST LTLST)
  )
  ((= ANS "Color")
   (setq LST COLST)
  )
)                                ;Đóng COND
;
(While
  (or
    (initget 1 "Exit")
    (/= "Exit"
      (setq POS
          (getint "\nExit/<Posion Number>: ")
      )
    )
  )                              ;Đóng OR
  ;
  (princ (nth POS LST))
)                                ;Đóng while
;Kết thúc chương trình
```

4.

```
;Tên file: SET-TOL.LSP
(defun C:SET-TOL (/ TOLLST OPT H PT MT I)
;Định nghĩa hàm thay thế phần tử
(defun Replace (I NEW L)
   (append (reverse (cdr (member I (reverse L))))
           (List NEW) (cdr (member I L)))
)
;
;(setq TOLLST '())
(While
  (or
    (initget  "Create Modify Display Set Exit")
```

```
  (/= "Exit"
    (setq OPT
        (getkword "\nCreate/Modify/Display/Set/<Exit>: ")
        OPT (if (= nil OPT) "Exit" OPT)
    )
  )
)                                ;Đóng OR
;
(cond
  ((= OPT "Create")
   (setq H (strcase (getstring "\nEnter header: "))
        PT(getreal "\nPlus tolerance: ")
        MT (abs (getreal "\Minus tolerance: "))
        TOLLST (append TOLLST (list (list H PT MT)))
   )                             ;Đóng SETQ
  )                              ;Đóng Create
  ;
  ((= OPT "Display")
    (setq H (strcase (getstring "\nEnter header: "))
    )
   (princ (assoc H TOLLST))
  )                              ;Đóng "Display"
  ;
  ((= OPT "Modify")
    (setq H (strcase (getstring "\nEnter header: "))
         PT(getreal "\nPlus tolerance: ")
         MT (abs (getreal "\Minus tolerance: "))
         I (assoc H TOLLST)
         TOLLST (replace I (list H PT MT) TOLLST)
    )
  )                              ;Đóng Modify
  ;
  ((= OPT "Set")
    (setq H (strcase (getstring "\nEnter header: ")))
    (setvar "DIMTP" (cadr (assoc H TOLLST)))
    (setvar "DIMTM" (caddr (assoc H TOLLST)))
   )                             ;Đóng Set
 )                               ;Đóng cond
)                                ;Đóng while
)                                ;Kết thúc chương trình
```

Chương 11

CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG AutoCAD

Nội dung chương

1. Quản trị cơ sở dữ liệu đối tượng **AutoCAD**.
2. Ngôn ngữ lập trình **AutoLISP** và cơ sở dữ liệu đối tượng **AutoCAD**.

11.1 Quản trị cơ sở dữ liệu

Cơ sở dữ liệu (database) là nơi chứa thông tin, được tổ chức sao cho có thể truy xuất thông tin dễ dàng và hiệu quả. Thông qua *hệ quản trị cơ sở dữ liệu*, người sử dụng có thể tạo, xóa, sửa và lấy thông tin chứa trong cơ sở dữ liệu.

Cơ sở dữ liệu được lưu thành các file. Trong mỗi file, thông tin lưu thành từng *hàng (record)*, mỗi hàng chia thành nhiều *cột (field)*.

Ví dụ:

Cơ sở dữ liệu địa chỉ thư tín sau đây có 5 *record*, mỗi *record* có 4 *field*.

NAME	COMPANY	ADDRESS	CITY
Phạm Ngọc Minh	Tân Tiến	534 Phan Kế Bính	TP HCM
Trần Khánh Vy	Vimico	912 Trương Định	TP HCM
Nguyễn Thanh Trung	Hưng Phát	101/17/19A Gò Dầu	TP HCM
Nguyễn Hữu Lộc	An Cư	12/13 Hoàng Hoa Thám	TP.HCM
Lê Văn An	Tiến Đạt	210 Lý Thường Kiệt	Nha Trang

AutoCAD cũng có thể được xem là hệ quản trị cơ sở dữ liệu. Mỗi file bản vẽ là một cơ sở dữ liệu. Mỗi đối tượng được vẽ ra (đường thẳng, đường tròn, cung tròn...) sẽ được lưu lại thành một *record*. **AutoCAD** sử dụng các *record* này để vẽ lại các đối tượng trên màn hình. Các *field* dùng để chứa thông tin mô tả đối tượng. Ví dụ một đường thẳng chứa điểm đầu, điểm cuối, lớp bản vẽ, màu sắc, dạng đường ...

Ví dụ:

Dùng lệnh **List** của **AutoCAD** để xem thông tin về một đường thẳng. Thông tin này được lấy ra từ cơ sở dữ liệu của **AutoCAD**.

```
LINE     Layer: 0
         Space: Model space
         Handle = 4C
from point, X=  5.0948  Y=  5.8785  Z=  0.0000
to point, X= 11.4118  Y=  7.5581  Z=  0.0000
Length =  6.5364,  Angle in XY Plane =    15
Delta X =  6.3169, Delta Y =  1.6796, Delta Z =  0.0000
```

Cơ sở dữ liệu bản vẽ **AutoCAD** được lưu thành file *.DWG*. Các file này ở dạng nén rất cao để lưu trữ và truy xuất hiệu quả. Do đó ta không thể đọc trực tiếp được. Khi cần thiết phải trao đổi file bản vẽ với các phần mềm khác, ta xuất bản vẽ ở dạng *.DXF (Drawing eXchange File)*. Đây là file dạng văn bản và ta có thể dùng các phần mềm như *Notepad* để đọc.

AutoLISP lấy thông tin về các đối tượng bản vẽ ở dạng danh sach phức hợp. Các danh sách con (*field*) chứa thông tin về đối tượng. *Header* của mỗi danh sách con là các *DXF group code*, dùng để xác định các *field* (gọi là *field descriptor*).

Group code	Value Type
0	Bắt đầu đối tượng. Dạng của đối tượng dựa vào giá trị text kèm theo nhóm này
6	Tên dạng đường (Linetype name)
8	Tên lớp (Layer name)
10	Điểm đầu tiên (Start point đối với đối tượng line hoặc text, điểm tâm của đường tròn...

Ví dụ:

Dưới đây là phần đầu của một *record* do **AutoLISP** lấy ra từ cơ sở dữ liệu đối tượng **AutoCAD**.

```
(
 (-1 . <Entity name: 3250500>)
 (0 . "LINE")
 (8 . "0")
 (10  2.0  2.0  0.0)
 (11  4.0  4.0  0.0)
)
```

Danh sách phức hợp trên mô tả một đối tượng, gồm có 6 *field*. Mỗi field bắt đầu bằng *field descriptor* (là một *DXF group code*), sau đó là giá trị của *field*. Với các đối tượng khác nhau, các *field* này sẽ khác nhau, phụ thuộc vào thông tin cần thiết để mô tả đối tượng.

✌ **Ví dụ:**

Field	Descriptor	Giá trị	Mô tả
(-1 . <Entity name: 3250500>)	-1	<Entity name: 3250500>	Tên đối tượng do **AutoCAD** quản lý
(0 "LINE")	0	"LINE"	Kiểu đối tượng là đường thẳng
(8 "0")	8	"0"	Đối tượng nằm ở lớp 0
(10 2.0 2.0 0.0)	10	2.0 2.0 0.0	Tọa độ điểm đầu của đường thẳng là (2.0 2.0 0.0)
(11 4.0 4.0 0.0)	11	4.0 4.0 0.0	Tọa độ điểm cuối của đường thẳng là (4.0 4.0 0.0)

11.2 AutoLISP và cơ sở dữ liệu đối tượng

Người lập trình **AutoLISP** có thể truy xuất đến từng đối tượng trong cơ sở dữ liệu đối tượng của **AutoCAD**. Tên đối tượng lấy ra có thể gán cho các biến và sử dụng khi cần thiết.

Ví dụ khi cần hiệu chỉnh đối tượng, chương trình sẽ sử dụng các biến chứa tên đối tượng, mà không cần người sử dụng chọn trên màn hình.

Thông qua tên đối tượng, chương trình sẽ lấy ra *record* tương ứng, chỉnh sửa giá trị các *field*, sau đó cập nhật vào lại cơ sở dữ liệu của **AutoCAD**. Kết quả là các đối tượng sẽ được hiệu chỉnh. Chương trình có thể tạo ra *record* mới tương ứng với việc tạo ra đối tượng mới. Việc chỉnh sửa trực tiếp các đối tượng trong cơ sở dữ liệu nhanh hơn và uyển chuyển hơn khi dùng các lệnh **AutoCAD**.

Chương 12

TẬP HỢP CÁC ĐỐI TƯỢNG CHỌN

Nội dung chương

1. Các hàm cơ bản về tập hợp các đối tượng chọn: **Ssget**, **Ssgetfirst**, **Sssgetfirst**, **Sslength**, **Ssname**, **Ssnamex**. Các tham số: **Mode** và **Filter-List**.
2. Thêm, xóa và tìm các đối tượng trong tập hợp chọn bằng các hàm: **Ssadd**, **Ssdel**, **Ssmemb**.

12.1 Cơ bản về tập hợp các đối tượng chọn (selection set)

Các lệnh hiệu chỉnh của **AutoCAD** như: **Copy**, **Move**, **Rotate**, **Scale**... thường xuất hiện dòng nhắc *"Select objects:* " yêu cầu chọn các đối tượng cần hiệu chỉnh. Khi đó một tập hợp các đối tượng chọn được tạo ra. Chúng được các lệnh **AutoCAD** thao tác cùng nhau.

Ví dụ:

Command: **Copy**↵
Select objects: (Lần lượt chọn các đối tượng)
Select objects: ↵
<Base point or displacement>/Multiple: (Thực hiện lệnh bằng cách chọn điểm chuẩn)

Khi một tập hợp chọn mới được tạo ra, tập hợp chọn cũ vẫn được **AutoCAD** ghi nhớ, gọi là tập hợp các đối tượng chọn *Previous*. Chỉ có duy nhất một tập hợp như vậy. Các tập hợp chọn cũ hơn sẽ bị mất đi.

Hàm SSGET

Hàm **Ssget** *(Selection Set GET)* tạo ra một tập hợp chọn gồm các đối tượng do người sử dụng chọn trên màn hình. Chúng chứa tên các đối tượng được chọn. Các tập hợp chọn có thể được gán cho các biến.

(**Ssget** [MODE] [PT1] [PT2]] [PT-LIST] [FILTER-LIST])

Ví dụ: Dùng hàm **Ssget** không tham số

Command: **(ssget)** ↵ *Select objects*: (chọn các đối tượng) *Select objects*: ↵ *<Selection set: 1>*	Tập hợp chọn được tạo ra khi ta chọn đối tượng bằng các phương pháp khác nhau: Pick (chọn đối tượng), Window (khung cửa sổ), Crossing (khung cửa sổ giao)... Tập hợp chọn kết thúc khi nhấn ENTER
Command: **(command ".move" "P" "" '(0 0) '(45 30)** ↵	Lúc này, tập hợp chọn ở trên trở thành tập hợp các đối tượng chọn *Previous*. Lệnh **Move** tác động lên tập hợp này bằng lựa chọn "P"

Tập hợp các đối tượng chọn có thể được gán cho một biến để quản lý và sử dụng.

Ví dụ:

Command: **(setq A (ssget))** ↵ *Select objects:* (Chọn các đối tượng) *Select objects:* ↵ *<Selection set: 2>*	Tạo ra tập hợp chọn và gán cho biến A
Command: **(command ".array" A "" "R" 3 4 15.0 20.0)** ↵	Lệnh **Array** sử dụng tập hợp chọn chứa trong biến A.
Command: **(command ".copy" A "" pause pause)** ↵	Sao chép các đối tượng chứa trong biến A. Thực hiện hai lần **Pause** để nhập hai điểm *base point* và *second point*.
Command: **(command ".erase"A "")** ↵	Xóa các đối tượng chứa trong biến A. Các đối tượng bị xóa có thể được phục hồi bằng lệnh **Oops**.

Một số quy ước và lời khuyên khi sử dụng các tập hợp chọn:

1. Nên gán các tập hợp chọn cho các biến để có thể sử dụng tại mọi thời điểm.
2. Một đối tượng có thể thuộc nhiều tập hợp chọn khác nhau.
3. Các tập hợp chọn được cập nhật thường xuyên ngay khi các đối tượng của tập hợp bị thay đổi như di chuyển, xóa ...
4. Các lệnh **Oops** và **Undo** của **AutoCAD** có thể phục hồi các tập hợp chọn về trạng thái ban đầu.
5. Các đối tượng trong không gian mô hình và không gian giấy vẽ có thể được chứa trong cùng một tập hợp chọn, nhưng được sử dụng riêng biệt trong không gian tương ứng. Khi đang ở trong không gian mô hình, các đối tượng của không gian giấy vẽ sẽ không có hiệu lực và ngược lại.
6. Có thể dùng hàm **Setq** để gán một tập hợp chọn không sử dụng nữa về nil. Khi đóng bản vẽ, tất cả các tập hợp chọn được gán bằng rỗng.
7. **AutoLISP** có khả năng quản lý tối đa 128 tập hợp chọn. Do đó các biến chứa tập hợp chọn nên được khai báo là biến cục bộ. Khi gán biến bằng nil, tập hợp chọn vẫn còn tồn tại trong bộ nhớ. Nó chỉ bị xóa hẳn

Ví dụ:

Chương trình sau đây thường được dùng khi sắp xếp các đối tượng

```
;Tên file: DXY.LSP
;Mục đích: Dời tập hợp các đối tượng chọn dọc theo trục X hoặc trục Y.
;
(defun C:DXY(/ OBJ AX BAS TAR)
  (setvar "cmdecho" 0)
  (princ "\nSelect objects for ortho move: ")
  (setq OBJ  (ssget))                          ;Tập hợp chọn OBJ
  (initget 1 "X Y")
  (setq AX   (getkword "\nMove along X/Y axis: ")
        BAS  (getpoint "\nSpecify base point: ")
        TAR  (getpoint "\nSpecify target point: ")
  )
  (command ".move" OBJ "" BAS        ;Bắt đầu lệnh Move với điểm base
                                     ;point ...
        (if (= AX "X")
          (list (car TAR) (cadr BAS))  ;nếu dời dọc theo trục X
          (list (car BAS) (cadr TAR))  ;nếu dời dọc theo trục Y
        )
  )                                  ;Đóng command
  (setvar "cmdecho" 1)
  (princ)
)                                    ;Kết thúc chương trình
```

Tham số MODE

Tham số MODE xác định phương pháp chọn đối tượng.

Mode	Phương pháp chọn	Cú pháp
None	Sử dụng được mọi phương pháp chọn	(ssget)
<point>	Chọn đối tượng phát hiện tại điểm <point>.	(ssget <point>)
"L"	Chọn đối tượng được tạo ra cuối cùng (*Last*)	(ssget "L")
"P"	Chọn tập hợp *Previous*	(ssget "P")
"W"	Chọn đối tượng bằng khung cửa sổ *Window*	(ssget "W" pt1 pt2)
"C"	Chọn đối tượng bằng cửa sổ giao *Crossing Window*	(ssget "C" pt1 pt2)
	bằng *Implied*	(ssget "I")

	selection set. Nếu trước khi thực hiện một lệnh **AutoCAD**, ta dùng chuột chọn các đối tượng bằng cửa sổ *Window* hoặc *Crossing Window*, tập hợp chọn này gọi là *Implied selection set*	
"F"	Chọn đối tượng bằng đường cắt *Fence*	(ssget "F" pt-list)
"WP"	Chọn đối tượng bằng cửa sổ đa giác *Window Polygon*	(ssget "WP" pt-list)
"CP"	Chọn đối tượng bằng đa giác cắt *Crossing Polygon*	(ssget "COPYCLIP" pt-list
"X"	Chọn tất cả các đối tượng trong cơ sở dữ liệu của bản vẽ	(ssget "X")

Ví dụ:

```
(setq PT1 '(10.0 10.0 0.0)
      PT2 '(160.0 10.0 0.0)
      PT3 '(160.0 100.0 0.0)
      PT4 '(10.0 100.0 0.0)
)
```

Gán tọa độ cho các điểm PT1, PT2, PT3, PT4

```
(ssget)
```

Người sử dụng có thể chọn các đối tượng bằng các phương pháp chọn khác nhau của **AutoCAD**.

```
(ssget '(180 120 0))
```

Ô vuông chọn (*pickbox*) ở tại điểm (180,120,0). Đối tượng nào giao với ô vuông này sẽ được chọn. Nếu có nhiều đối tượng thì chỉ có đối tượng được vẽ sau cùng mới được chọn.

```
(ssget PT1)
```

Chọn đối tượng tìm thấy tại điểm PT1.

```
(setq A (ssget "W" PT1 PT3))
```

Tập hợp chọn được tạo ra bằng cửa sổ chọn của **AutoCAD**. Cửa sổ này có hai đỉnh là PT1 và PT2. Các đối tượng chọn sẽ được gán cho biến A.

```
(ssget "C" '(0 0) '(360 240))
```

Chọn đối tượng bằng cửa sổ cắt. 2 đỉnh góc của cửa sổ là (0,0) và (360,240).

```
(while (setq G (ssget "L"))
  (command ".erase" G "")
```

Trong khi vẫn còn đối tượng trong cơ sở dữ liệu ((ssget "L") trả về khác

```
            ".delay" 50
    )
)
(setq W (ssget "P"))
(princ  "\nErase Previous set?")
  (if (= (getstring) "Y")
     (command ".erase" W "")
  )
(setq L (ssget "I")
      M (ssget PT2))
(command ".select" L M "")
(setq N (ssget "P"))


(ssget "F" (list PT3 PT4))


(ssget "WP" (list '(20 20)
                  '(40 20)
                  '(40 40)
                  '(20 40)
            );Đóng list
);Đóng ssget
(ssget "CP" (list PT1 PT2 PT3 PT4))


(ssget "X")
```

nil) thì xóa phần tử đó. Sau đó dừng lại một chút (".delay" 50) rồi tìm và xóa tiếp.

Tập hợp chọn *Previous* được gán cho biến W. Nếu muốn, tập hợp chọn này sẽ bị xóa.

Gán tập hợp *Implied selection set* cho biến L. Gán đối tượng phát hiện tại điểm PT2 cho biến M. Dùng lệnh SELECT tạo một tập hợp chọn bao gồm 2 tập hợp L và M. Tập hợp mới này được gọi lại bằng mode "P" và gán cho biến N.

Tạo tập hợp chọn gồm các đối tượng giao với đường cắt. Đường cắt này là đường thẳng qua 2 điểm PT3, PT4.

Tạo tập hợp chọn gồm các đối tượng nằm trong đa giác. Đa giác này có các đỉnh là các phần tử của danh sách. Trong ví dụ này là 4 điểm (20,20), (40,20), (40,40), (20,40).

Tạo tập hợp chọn bằng cửa sổ cắt đi qua các đỉnh PT1, PT2, PT3, PT4. Các đỉnh này là các phần tử của danh sách dùng làm tham số cho hàm **Ssget**.

Chọn tất cả các đối tượng trong cơ sở dữ liệu của bản vẽ, kể cả các đối tượng nằm ở các lớp bị đóng băng, hoặc các lớp bị tắt đi.

Ví dụ:

Chương trình sau đây tạo ra tập hợp SS1 chứa tất cả các đối tượng của bản vẽ. Sau đó làm tan băng (*thaw*) và mở lại (*on*) tất cả các lớp bản vẽ để có thể tạo ra tập hợp chọn SS2 chứa tất cả các đối tượng nằm trong giới hạn bản vẽ. Sau đó chương trình xóa tất cả các đối tượng nằm ngoài giới hạn bản vẽ.


```
; Tên file: EOL.LSP
(defun C:EOL (/ SS1 SS2)
  (setvar "cmdecho" 0)
  (prompt "\nCreating selection set of entire drawing ...")  ;Tạo SS1 toàn bộ
(setq SS1 (ssget "X"))                                       ;bản vẽ
  (command ".layer" "thaw" "*" "on" "*" "")  ;Kích hoạt tất cả các lớp
  (command ".zoom" "V")                      ;Zoom màn hình virtual
  (prompt "\nCreating selection set within drawing limits...");Tao SS2
  (setq SS2 (ssget "W" (getvar "LIMMIN") (getvar "LIMMAX")))
  (prompt "\nErasing objects outside of drawing limits...")
  (command ".erase" SS1 "remove" SS2 "");Xoá tất cả các đối tượng, nhưng
                                        ;loại bỏ SS2 trước khi ;xóa.
  (command ".zoom" "P")              ;Lựa chon Previous của lệnh Zoom
  (setvar "cmdecho" 1)
  (princ)
)                                    ;Kết thúc EOL
```

Tham số Filter-List

Tham số Filter-list cho phép bỏ bớt các đối tượng không thỏa mãn điều kiện lọc trong quá trình chọn đối tượng theo các phương pháp khac nhau đã trình bày ở trên.

(**Ssget** MODE FILTER-LIST)

Điều kiện lọc có kiểu danh sách phức hợp. Nó chứa nhiều điều kiên khác nhau, được biểu diễn bằng các danh sách con. Mỗi danh sách con là các *dotted pair*, phần tử đầu tiên là *group code* xác định loại điều kiện lọc, phần tử thứ hai là giá trị lọc tương ứng.

Bảng các Group code:

Group code	Ý nghĩa /Giá trị
-5	Persistent reactor chain.
-4	Conditional operator (for use only with **Ssget**).
-3	Extended entity data (XDATA) sentinel (fixed value).
-2	Entity name (fixed entity name reference).
0	Entity type (fixed value).
1	Primary text value for an entity.
2	A name: Block name, Attribute tag...
3-4	Miscellaneous text or name values.
6	Line type name (fixed value).
7	Text style or Attribute Definition name (fixed value).
8	Layer name (fixed value).
10	Primary entity point (Start of line, center of circle...).

39	Thickness (fixed value).
40-48	Floating-point values (Block scale factors, text height, etc.).
49	Repeated value for an entity with variable length tables (such as a LTYPE's dash length). A 7x group appears before the first 49 goup spcifying the table.
50-58	Angles.
62	Color number (fixed value; 0 = BYBLOCK, 256 = BYLAYER).
66	"Entities follow" flag (Attributes follow Block flag, etc.).
67	Drawing space (1 = Paper space, 0 = Model space).
70-78	Integers for counters, flag bits, or other modes.
90-99	32-bit integer values.
100	Subclass data marker.
102	String followed by "{arbitrary name" or "}".
105	Dimension variable symbol table entry object handle.
210	3D extrusion direcion vector (list of three real numbers).
280-289	8-bit integer values.
300-309	Arbitrary text strings.
310-319	Arbitrary binary chunks.
320-329	Arbitrary object handles.
330-339	Soft pointer handle (specifies pointer to other objects in the drawing).
340-349	Hard pointer handle (specifies pointer to other objects in the drawing).
350-359	Soft owner handle (specifies ownership to other objects in the drawing).
360-369	Hard owner handle (specifies ownership to other objects in the drawing).
999	Comments.

✌ **Ví dụ:**

(ssget "X" (list (cons 6 "HIDDEN"))) Dùng hàm **Cons** để tạo dotted pair: group code là 6 (tên dạng đường), giá trị lọc là "HIDDEN". Tập hợp chọn chứa tất cả các đối tượng trong cơ sở dữ liệu có dạng đường là HIDDEN.

(ssget "X" '((0 . "TEXT") (7 . "STANDARD") (8 . "NOTES")))

3 điều kiện lọc:
- Kiểu đối tượng là TEXT
- Kiểu chữ là STANDARD
- Thuộc lớp bản vẽ có tên là NOTES

Tất cả các đối tượng thỏa mãn đồng thời ba điều kiện trên sẽ được chọn.

(ssget "X" '((8 . "T*") (7 . "BK?"))) Có thể dùng các ký tự đại diện (* và ?) cho các group code 0, 2, 3, 6, 7 va 8. Trong ví dụ này, các đối tượng được chọn phải thuộc các lớp bản vẽ có tên bắt đầu bằng T và tên của đối tượng (block name, attribute tag...) gồm 3 ký tự, 2 ký tự đầu là BK.

✌ **Ví dụ:**

(command ".chprop" (ssget "X" '((62 . 1))) "" "C" "5" "")

Các đối tượng có màu 1 (red) được chuyển sang màu 5 (blue)

(command ".chprop" (ssget "X" '((62 . 0))) "" "C" "Bylayer" "LA" "Hatch" "")

Các đối tượng có màu là BYBLOCK được chuyển sang màu BYLAYER, và chuyển sang lớp Hatch.

Command: **Chprop**↵
Select objects: **(SSGET "X" '((0 . "CIRCLE") (8 . "0") (39 . 20)))** ↵
Select objects: **(SSGET "X" '((0 . "CIRCLE") (8 . "DRAW") (39 . 20)))** ↵
Select objects: ↵
Change what property (Color/LAyer/LType/ltScale/Thickness) ? **LA** ↵
New layer <varies>: **3D-CIRCLES** ↵ Các đối tượng là đường tròn có Thickness = 20 của lớp 0 và lớp DRAW được chuyển về lớp 3D-CIRCLES

✌ **Ví dụ:** Ngoài MODE "X" ta co thể dùng các MODE khác kết hợp với FILTER-LIST.

(ssget '(5 5) '((8 . "0"))) Chọn đối tượng tại điểm '(5 5) và của lớp "0".

(setq J (ssget "W" '(0 0 0) '(297 210 0) '((0 . "TEXT") (62 . 256))))

Chọn các đối tượng bằng khung cửa sổ có 2 đỉnh là '(0 0 0) và '(297 210 0). Chỉ các đối tượng là dòng chữ (text) có màu là "BYLAYER" (62 . 256) mới được chọn.

(setq H (ssget "I" '((0 . "CIRCLE") (10 75 115 0) (67 . 1))))

Chon các đối tượng bằng *Implied selection set.* Chỉ các đối tượng là đường tròn trong không gian giấy vẽ có tâm tại điểm '(75 115 0) mới

được chọn. Ghi chú: danh sách (10 75 115 0) chứa *group code* 10 và tọa độ điểm, trong đó tọa độ điểm không có dấu ' phía trước và cũng không chứa trong dấu ngoặc. Điều này cũng đúng với các danh sách khác chứa *group code* và tọa độ điểm.

```
(setq M (ssget "L" (list (cons 0 "LINE") (cons 6 "HIDDEN"))))
```

Nếu đối tượng được vẽ cuối cùng (*Last*) là đường thẳng và có dạng đường là "HIDDEN" thì nó được chọn. Hàm **Cons** dùng để tạo *dotted pair*.

```
(setq K (ssget "P" '((0 . "POLYLINE") (8 . "3D*") (39 . 27.5))))
```

Biến K chứa các đối tượng lấy từ tập hợp chọn Previous. Các đối tượng này phải thỏa điều kiện là các polyline, có thickness = 27.5, thuộc các lớp bản vẽ có tên bắt đầu bằng 2 ký tự "3D".

```
(ssget "WP" (list '(0.0 0.0)          Chọn các đối tượng bằng cửa sổ đa
                  '(8.0 0.0)          giác WP đi qua 4 đỉnh. Các đối
                  '(8.0 8.0)          tượng này phải thỏa điều kiện:
                  '(0.0 8.0)
            )
            '((2 . "DOOR-68")         -là block có tên "DOOR-68"
              (8 . "DOORS")           -thuộc layer "DOORS"
              (66 . 1)                -và có chứa các thuộc tính
             )                        (attributes)
);Đóng ssget
```

Ví dụ:

Chương trình sau đây cho phép người sử dụng xóa tất cả các đối tượng là *block*, kể cả các *block* thuộc các lớp bản vẽ bị đóng băng (freeze) hoặc bị tắt đi (off). Khi nhập vào tên *block* để xóa, ta có thể dùng các ký tự đại diện *, ? để xóa nhiều *block* cùng lúc.

```
;Tên file: BLOCKOUT.LSP
(defun C:BLOCKOUT (/ BLK SET)
  (setvar "cmdecho" 0)
  (command ".textscr"                 ;Chuyển sang màn hình văn bản
           ".block" "?" "*"           ;và hiển thị tên tất cả các block.
  )
  (setq BLK (strcase (getstring "\nDelete which block[s]: "))
        SET (ssget "X" (list (cons 2 BLK)))
  )
;Nếu không phát hiện block nào thì hiện thông báo, ngược lại nếu có thì
;xóa tất cả các block.
  (if (null SET)
      (princ (strcat "\nThere are no occurrences of block "  BLK))
      (command ".erase" SET "")
  )
  (setvar "cmdecho" 1)
  (princ)
);Kết thúc BLOCKOUT.LSP
```

Ví dụ:

Chương trình cho phép thực hiện các lệnh hiệu chỉnh của **AutoCAD**. Người sử dụng chọn loại đối tượng muốn hiệu chỉnh, sau đó chọn lệnh hiệu chỉnh.

```
;Tên file: EDIT-ENT.LSP
  (defun C:EDIT-ENT (/ RES LAY SS1 COM)
    (setvar "cmdecho" 0)
    (initget 1 "Arc Circle Line Polyline Text")
  (setq RES (getkword "\nEdit <Arcs/Circles/
                              Lines/Plines/Text>: ")      ;Chọn kiểu đối tượng
        LAY (getstring "\nSelect layer by names: ")      ;Chọn lớp bản vẽ
        SS1 (ssget "X" (list (cons 0 RES) (cons 8 LAY)));Chọn các đối tượng
                                                         ;thỏa điều kiện
  )                                                      ;Đóng setq
 (if SS1
   (progn
     (initget 1 "Copy Erase Move Rotate Scale")
   (setq COM (getkword "\nCommand <Copy/
                              Erase/Move/Rotate/Scale>: "))
    (command COM SS1 "")
  )                                                      ;Đóng progn
  (princ "\nNo objects found!")
  )                                                      ;Đóng if
  (setvar "cmdecho" 1)
  (princ)
)                                                        ;Đóng defun
```

Các phép so sánh số học

Khi cần thiết ta có thể dùng các phép so sánh số học để chọn các đối tượng thích hợp nhờ vào *group code* -4.

(**-4** . "Phép so sánh")

Các phép so sánh này áp dụng cho các *số nguyên, số thực, tọa độ điểm, vec tơ 3 chiều* của cơ sở dữ liệu bản vẽ.

Phép so sánh	Ý nghĩa
"*"	Chấp nhận mọi giá trị
"="	Bằng
"!="	Không bằng
"/="	
"<>"	
"<"	Nhỏ hơn
"<="	Nhỏ hơn hoặc bằng
">"	Lớn hơn
">="	Lớn hơn hoặc bằng

Các phép so sánh phải đặt trong dấu nháy chuỗi. Phép so sánh áp dụng cho danh sách con đi tiếp theo sau.

Một vài quy định khi dùng phép so sánh:

1. Nếu *group code* xác định một số nguyên hoặc số thực, thì phép so sánh mới sử dụng được.

Ví dụ:

(ssget "X" '((0 . "TEXT") (-4 . ">") (50 . 0.0)))

Tạo tập hợp chứa các dòng chữ (text) có độ nghiêng lớn hơn 0. Phép so sánh ">" áp dụng cho group code 50, xác định góc nghiêng 0.0 là số thực.

(ssget "X" '((0 . "LINE") (-4 . "<") (8 . "WINDOWS")))

Lỗi. Dự định tạo ra tập hợp chọn chứa tất cả các đường thẳng thuộc các lớp bản vẽ có tên nhỏ hơn "WINDOWS" *Group code* xác định chuỗi không thể so sánh trong trường hợp này.

2. Trong các *group code* tọa độ điểm, các tọa độ X, Y, Z có thể được so sánh riêng biệt nhau.

Ví dụ:

(ssget "X" '((0 . "CIRCLE") (-4 . ">,>,*") (10 0.0 0.0 0.0)))

Tạo tập hợp chọn chứa tất cả các đường tròn có tọa độ X và Y của tâm đường tròn lớn hơn 0, và tọa độ Z có giá trị bất kỳ. *Group code* 10 xác định tâm đường tròn.

(ssget "X" '((0 . "CIRCLE") (-4 . ">,>") (10 0.0 0.0 0.0)))

Phép so sánh tọa độ Z được ngầm hiểu là "*".

3. Chỉ có thể dùng các phép so sánh "*", "=", "!=" cho *group code* 210 (vec tơ định hướng trục Z của đối tượng).

Ví dụ:

(ssget "X" '((0 . "SOLID") (-4 . "!=") (210 0.0 0.0 1.0)))

Code 210 xác định điểm ngọn của vectơ định hướng có điểm gốc là gốc tọa độ WCS '(0.0 0.0 0.0). Ví dụ này tạo tập hợp chọn chứa tất cả các 2D-SOLID có vectơ định hướng khác với vectơ song song trục Z của WCS. Đó là các solid tạo ra trong các mặt phẳng khác với mặt phẳng XY của hệ trục tọa độ WCS.

Các ví dụ khác:

(ssget "W" '(0 0) '(297 210) '((0. "CIRCLE") (-4 "<") (40 . 50)))

Chọn tất cả các đường tròn nằm trong cửa sổ chọn có 2 đỉnh là '(0 0) và '(297 210) và có bán kính nhỏ hơn 50.

(ssget "P" '((0 . "TEXT") (-4 . "!=") (40 . 2.5)))

Tạo ra tập hợp chọn mới từ tập hợp *Previous*, chứa tất cả các dòng chữ có chiều cao khác 2.5.

(ssget "X" '((0 . "ARC") (-4 . ">=") (40 . 20.5) (-4 . "<>") (62 . 256)))

Tạo tập hợp chọn chứa tất cả các cung tròn có bán kính lớn hơn hoặc bằng 20.5 và có màu khác

```
                                    BYLAYER.
(ssget "X" '((0  "LINE")
    (-4 . ">,>,/=") (10 0.0 0.0 0.0)      ;group code 10 – điểm đầu
    (-4  "<,<,/=") (11 24.0 18.0 0.0)     ;group code 11 – điểm cuối
    )                                      ;Đóng filter-list
)                                          ;Đóng ssget
```

Các phép toán logic

Các phép toán logic **And, Or, Not** dùng để kết hợp các filter-list thành một điều kiện lọc duy nhất.

a) "<AND" ... các group code ... "AND>"

Tất cả các *group code* phải trả về True thì **And** mới được thỏa mãn, đối tượng mới được chọn.

Ví dụ:

```
(ssget "X" '((-4  "<AND")
      (0 . "TEXT")
      (8 . "NOTES")
      (40 . 2.5)
    (-4 . "AND>")  ;Đóng AND
  );Đóng filter-list
);Đóng ssget
```

Tạo ra tập hợp chọn chứa tất cả các dòng chữ thuộc lớp "NOTES" và có chiều cao là 2.5.

b) "<OR" ... các group code ... "OR>"

Chỉ cần một *group code* trả về True thì **Or** được thỏa mãn, đối tượng sẽ được chọn.

Ví dụ:

```
(ssget "X" '((-4  "<OR")
      (0 . "CIRCLE")
      (39 . 10)
      (40 . 2.5)
    (-4 . "OR>")  ;Đóng OR
  )               ;Đóng filter-list
)                 ;Đóng ssget
```

Chọn các đối tượng thỏa mãn một trong các điều kiện sau:

- Đối tượng là đường tròn.
- Đối tượng có Thickness = 10
- Mọi đối tượng có *group code* 40 chứa giá trị là 2.5

c) "<NOT" ... các group code ... "NOT>"

Chọn các đối tượng không thỏa mãn các điều kiện của các *group code*.

Ví dụ:

```
(ssget "X" '((0 . "INSERT")
   (2  "PART"-39")
   (-4 . "<NOT")  ;first NOT
     (8 . "0")
   (-4 . "NOT>")
   (-4 . "<NOT")  ;second NOT
     (41 . 1.0)
   (-4 . "NOT>")
  )               ;Đóng filter list
)                 ;Đóng ssget
```

Chọn các đối tượng là khối có tên "PART-39" thỏa mãn 2 điều kiện sau:

- Không được chèn trong lớp 0
- Không có hệ số tỉ lệ chèn theo trục X là 1.0

Ví dụ:

Thỉnh thoảng các dòng chữ nổi 3D (có chiều dày thickness lớn hơn 0) gây khó khăn cho **AutoCAD**. Chương trình sau đây gán thickness cho tất cả các dòng chữ bằng 0. Chương trình cho phép chọn lớp bản vẽ được phép giữ lại các dòng chữ nổi 3D, không bị gán Thickness bằng 0.

```
;Tên file: FLATTEXT.LSP
(defun C:FLATTEXT (/ LAY TXT)
  (setvar  "cmdecho" 0)
  (prompt "\nRemove which Layer
       from processing <ENTER for none>: ");Giữ lại các dòng chữ 3D cho
                                           ;lớp bản vẽ này
  (if (= (setq LAY (strcase (getstring))) "")
     (progn                                ;Thay đổi tất cả các bản vẽ
       (prompt "\nLocating all TEXT entities with extrusions <> 0")
       (setq TXT (ssget "X" '((0 . "TEXT") (-4 . "<>") (39  0.0))))
       (if (null TXT)
         (prompt "\nNo objects found!")
         (command ".chprop" "P" "" "Thickness" 0.0 "")
       )
     ) ;Đóng progn
     (progn                                ;Bỏ qua lớp LAY
       (prompt "\nLocating all TEXT entities with extrusions <>0")
       (princ (strcat "\nExcluding layer " LAY))
       (setq TXT (ssget "X" (list '(0  "TEXT")
                                  '(-4 . "<>")
                                  '(39 . 0.0)
                                  '(-4  "<NOT")
```

```
                                (cons 8 LAY)
                            '(-4 . "NOT>")
            )                                   ;Đóng filter-list
          )                                     ;Đóng ssget
        )                                       ;Đóng setq
        (if (null TXT)
          (prompt "\nNo objects found!")
          (command ".chprop" "P" "" "Thickness" 0.0 "")
        )                                       ;Đóng if
      )                                       ;Đóng progn–ELSE expression
  )                                           ;Đóng if
  (setvar "cmdecho" 1)
  (princ)
)                                             ;Đóng defun
```

Tóm tắt:

1. Hàm **Ssget** tạo ra tập hợp các đối tượng được chọn. Tập hợp này lưu tên của các đối tượng, tương tự như các dòng nhắc *"Select objects: "* của **AutoCAD**.
2. Tham số MODE xác định phương pháp chọn đối tượng cho hàm **Ssget**.
3. Tham số FILTER-LIST chứa các điều kiện lọc đối tượng. Mỗi điều kiện là một *dotted pair* chứa *group code* xác định tính chất của đối tượng và giá trị của tính chất này. Chỉ những đối tượng nào thỏa mãn điều kiện lọc mới được chọn.
4. Các phép so sánh (*group code* –4) dùng làm điều kiện kiểm tra các giá trị số.
5. Các phép toán logic dùng để liên kết các điều kiện thành một điều kiện duy nhất.

Hàm SSGETFIRST

Khi chọn các đối tượng tại dòng nhắc *"Command:"*, có nghĩa là lúc thoát ra khỏi tất cả các lệnh của **AutoCAD**, các dấu GRIPS sẽ xuất hiện tại các điểm đặc biệt của đối tượng. Các GRIPS có ba trạng thái: **WARM**, **HOT** và **COLD**.

- Trạng thái **WARM**: Đối tượng được chọn có các GRIPS (màu mặc định là Blue) và có dạng đường HIDDEN.

- Trạng thái **COLD**: Đối tượng được chọn có các GRIPS (màu mặc định là Blue) và có dạng đường CONTINOUS.
- Trạng thái **HOT**: Đối tượng được chọn có GRIPS màu đỏ và có dạng đường HIDDEN

Khi cần xác định các đối tượng được chọn đang có GRIPS trên màn hình, ta dùng hàm **Ssgetfirst**.

(Ssgetfirst)

Hàm **Ssgetfirst** trả về một danh sách chứa hai tập hợp chọn. Tập hợp thứ nhất chứa các đối tượng có GRIPS ở trạng thái COLD. Tập hợp thứ hai chứa các đối tượng có GRIPS ở trạng thái WARM.

Ví dụ:

```
Command: (ssgetfirst) ↵
(<Selection set: 1> <Selection set: 2>) ;Selection set 1 chứa các đối tượng
                                        ;có GRIPS ở trạng thái COLD
                                        ;Selection set 2 chứa các đối tượng
                                        ;có GRIPS ở trạng thái HOT

Command: (ssgetfirst) ↵
(<Selection set: 3> nil)                ;Không có đối tượng nào có
                                        ;GRIPS ở trạng thái HOT

Command: (ssgetfirst) ↵
(nil <Selection set: 4>)                ;Không có đối tượng nào có
                                        ;GRIPS ở trạng thái WARM

Command: (ssgetfirst) ↵
(nil nil)                               ;Không có đối tượng nào
                                        ;đang có GRIPS
```

Hàm **Ssgetfirst** được dùng khi việc quản lý các đối tượng phụ thuộc vào trạng thái các GRIPS hoặc khi cần lưu giữ tập hợp các đối tượng này để sử dụng về sau.

Hàm **Ssget** chỉ chọn các đối tượng có GRIPS ở trạng thái HOT, và sau đó nó loại bỏ GRIPS, đưa các đối tượng trở về trạng thái không được chọn. Hàm **Ssgetfirst** cho phép chọn và sử dụng cả hai loại đối tượng HOT và WARM. Trạng thái GRIPS của các đối tượng này vẫn giữ nguyên sau khi kết thúc hàm **Ssgetfirst**.

Khi biến hệ thống PICKFIRST = 1, ta có thể chọn đối tượng trước, sau đó mới thực hiện lệnh hiệu chỉnh. Tuy nhiên có nhiều lệnh **AutoCAD**, như lệnh **Ddedit**, tự động xóa trạng thái GRIPS của các đối tượng trước khi

thực hiện lệnh. Do đó ta vẫn phải chọn lại các đối tượng. Ta dùng hàm **Ssgetfirst** để khắc phục vấn đề này.

Ví dụ:

Chương trình sau đây dùng hàm **Ssgetfirst** để kiểm tra có đối tượng nào đã được chọn trước chưa (đang có GRIPS). Nếu chỉ có một đối tượng TEXT được chọn thì hộp thoại **Text editor** xuất hiện cho phép sửa nội dung dòng chữ. Nếu có nhiều đối tượng đang được chọn, hoặc không có đối tượng nào đang được chọn thì lệnh **Ddedit** thực hiện bình thường, yêu cầu người sử dụng chọn đối tượng.

```
;Tên file: TE.LSP
(defun C:TE ()
  (if
    (and
      (setq SS (cadr (ssgetfirst)));Kiểm tra các đối tượng được chon grip
      (= 1 (sslength SS)) ;Thực hiên chỉ một đối tượng grip được chon
    )                     ;Đóng AND
    (command ".DDEDIT" SS);THEN- Bắt đầu Ddedit với đối tượng chọn
    (command ".DDEDIT")   ;ELSE- Bắt đầu Ddedit với không lựa chọn
  )                       ;Đóng IF
)
```

Hàm SSSETFIRST

Hàm **Sssetfirst** (*Selection Set SET FIRST*) dùng để gán trạng thái GRIPS cho các đối tượng trong bản vẽ.

(**Sssetfirst** [GRIPSET] [PICKSET])

Các đối tượng chứa trong tập hợp chọn GRIPSET sẽ được chuyển sang trạng thái COLD. Các đối tượng chứa trong tập hợp chọn PICKSET sẽ được chuyển sang trạng thái WARM. Hàm **Sssetfirst** không tham số sẽ loại bỏ tất cả các GRIPS.

Ví dụ:

(sssetfirst)	trả về nil	Xóa tất cả các GRIPS
(setq SS1 (ssget)) (sssetfirst SS1)	trả về (SS1)	Chuyển các đối tượng trong SS1 thành COLD.
(setq SS2 (ssget)) (sssetfirst SS1 SS2)	Trả về (SS1 SS2)	Chuyển các đối tượng trong SS1 thành COLD, các đối tượng trong SS2 thành WARM.

Ví dụ:

Chương trình sau đây chuyển đổi trạng thái các đối tượng từ WARM thành COLD và ngược lại.

```
;Tên file: SWHICH.LSP
(defun C:SWHICH ()
  (setq GFL (ssgetfirst))              ;GFL chứa các đối tượng đang có GRIPS
  (sssetfirst (cadr GFL) (car GFL))    ;(cadr GFL) chứa các đối tượng WARM
                                       ;(car GFL) chứa các đối tượng COLD.
  (princ)
)
```

Hàm SSLENGTH

Hàm **Sslength** (*Selection Set LENGTH*) trả về số phần tử trong tập hợp chọn. Nếu số phần tử nhỏ hơn 32767, hàm **Sslength** trả về số nguyên. Nếu lớn hơn 32767, hàm **Sslength** trả về số thực.

(**Sslength** SS)

Hàm này thường được sử dụng trong các vòng lặp hoặc dùng để so sánh các tập hợp chọn.

Ví dụ:

```
(setq C (ssget "L"))          ;C chứa đối tượng Last trong cơ sở dữ liêu
(sslength C)                  ;nên C có số phần tử bằng 1
```

Ví dụ:

Chương trình sau đây so sánh số lượng các đối tượng của lớp 0 với lớp do người sử dụng chọn.

```
;Tên file: LAY-ENT.LSP
(defun C:LAY-ENT (/ LAY SET)
  (setvar "cmdecho" 0)
  (setq LAY (strcase (getstring "\nLayer for entities calculation: "))
        SET (ssget "X" (list (cons 8 LAY)))
  )
  (if (null SET)
    (princ (strcat "The layer " LAY " does not exist or is empty!"))
```

```
    (princ (strcat "The layer " LAY "has "
           (itoa (sslength SET))          ;số sslength reports của đối tượng
           " entities."
            )                             ;Đóng strcat
    )                                     ;Đóng princ
  )                                       ;Đóng if
  (princ (strcat "\nThe layer 0 has "
          (itoa (sslength (ssget "X" (list (cons 8 "0")))))
           " entities."
         )                                ;Đóng strcat
  )                                       ;Đóng princ
  (setvar "cmdecho" 1)
  (princ)
)
```

Hàm SSNAME

Hàm **Ssname** (*Selection Set Name*) trả về tên đối tượng (tên do **AutoCAD** quản lý) tại vị trí INDEX trong tập hợp chọn SS.

(**Ssname** SS INDEX)

Vị trí đầu tiên được đánh số bằng 0. Nếu INDEX là số âm hoặc lớn hơn số lượng các đối tượng trong tập hợp chọn SS, hàm SSNAME trả về nil. Nếu số lượng đối tượng lớn hơn 32767, INDEX phải ở kiểu số thực.

Ví dụ:

```
(setq W (ssget "X" '((0 . "TEXT"))))      Chọn tất cả các dòng chữ
(command ".change" (ssname W 0) "" "" "" "" "" "" "New Text")
                                          Lấy ra phần tử đầu tiên trong W, sau
                                          đó đổi nội dung dòng chữ thành
                                          "New Text"
(setq  E (ssget "X" '((0 . "LINE"))))     Chọn tất cả các đường thẳng
(command ".erase" (ssname E 5) "")        Xóa đường thẳng thứ 6 (index = 5)
                                          trong tập hợp chọn E
```

Ví dụ:

Chương trình sau đây cho phép người sử dụng thay đổi chiều cao từng dòng chữ chứa trong tập hợp chọn. Mỗi dòng chữ sẽ được làm nổi bật lên bằng hàm (redraw ENT 3), sau đó hỏi người sử dụng có thay đổi chiều cao dòng chữ này không.

```
;Tên file: TEXTHGT.LSP
(defun C:TEXTHGT (/ RES TXTSET LAY HGT CTR ENT ANS)
```



```
(setvar "cmdecho" 0)
(setq RES (strcase
      (getstring "\nAdjust all text, or by layer <All/Layer>: ")))
(if (= RES "A")
   (setq TXTSET (ssget "X" '((0 . "TEXT"))))
   (setq LAY (strcase (getstring "\nSpecify layer: "))
         TXTSET (ssget "X" (list (cons 0 "TEXT") (cons 8 LAY)))
   )
)                                         ;Đóng if
(if (null TXTSET)
   (princ (strcat "\nThe layer " LAY
                  " does not exist or there is no text!"))
   (progn
   (initget 1)
      (setq HGT (getreal "\nNew text height: ")
            CTR 0
      )                                   ;Đóng setq
      (while (setq ENT (ssname TXTSET CTR))
        (redraw ENT 3)
        (setq ANS (strcase
              (getstring "\nChange this text entry <Y>: "))
        )                                 ;Đóng setq
        (if (or (= ANS "Y") (= ANS ""))
              (command ".change" ENT "" "" "" "" HGT "" "")
        )                                 ;Đóng if
        (redraw ENT 4)
        (setq CTR (1+ CTR))
      )                                   ;Đóng while
     )                                    ;Đóng prog
   )                                      ;Đóng if
  (setvar "cmdecho" 1)
  (princ)
)
```

Hàm SSNAMEX

Hàm **Ssnamex** (*Selection Set NAME indeX*) dùng để truy xuất đối tượng tên đối tượng và các thông tin phụ về đối tượng tại vị trí INDEX trong tập hợp chọn SS. Hàm trả về một danh sách chứa thông tin mô tả đối tượng và phương pháp chọn đối tượng này.

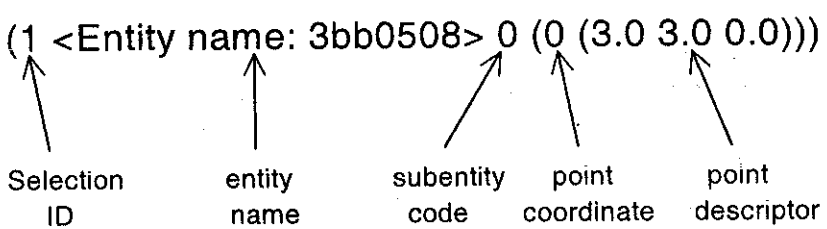
(**Ssnamex** SS [INDEX])

Nếu không có tham số INDEX, hàm **Ssnamex** trả về thông tin tất cả các đối tượng. Mỗi đối tượng chứa trong một danh sách con.

Ví dụ:

Command: **(setq SS (ssget))** ↵
Select objects: **3,3** ↵
1 found
Select objects: ↵
<Selection set: 25>
Command: **(ssnamex SS 0)** ↵
((1 <Entity name: 3bb0508> 0 (0 (3.0 3.0 0.0))))

Giải thích:



Selection ID: Phương pháp chọn đối tượng

Selection ID	Mô tả
0	Không xác định
1	Dùng chuột chọn (pick) hoặc nhập tọa độ một điểm
2	Dùng cửa sổ chọn (W) hoặc đa giác chọn (WP)
3	Dùng cửa sổ cắt (C) hoặc đa giác cắt (CP)
4	Dùng đường cắt (F)
8	Group name

Entity name: Tên đối tượng do **AutoCAD** quản lý

Subentity code: Cho biết đối tượng là đối tượng phức (ví dụ như polyline). Khi chọn đối tượng bằng cách pick, nếu chọn tại phân đoạn thứ nhất thì code này bằng 1. Nếu chọn tại phân đoạn thứ 2 thì mã này bằng 2...

Point descriptor: Bằng 0. Theo sau là danh sách tọa độ điểm.

Point coordinate: Tọa độ điểm. Nếu chọn bằng cửa sổ chọn hoặc đường cắt thì giá trị này bằng –1.

Ví dụ:

Sau đây là mô tả một đa tuyến polyline được chọn bằng cửa sổ cắt.

Command: **(setq SS (ssget))** ↵
Select objects: **C** ↵
First corner: (Select first corner)
Other corner: (Select other corner)
1 found
Select objects: ↵
<Selection set: 26>
Command: **(ssnamex SS 0)** ↵
((3 <Entity name: 34d0538> 1 -1) (-1 (0 (10.3528 5.32394 0.0)) (0 (15.1899 5.32394 0.0)) (0 (15.1899 2.91549 0.0)) (0 (10.3528 2.91549 0.0))))

Giải thích:

```
(
  (3 <Entity name: 34d0538> 1 -1)      3: chọn đối tượng bằng phương pháp
                                          Crossing hoặc Crossing Polygon
                                       1: Đối tượng phức
                                       -1: Đối tượng được chọn bằng cửa
                                          sổ chọn có mã là -1
   (-1                                 Mã cửa sổ chọn (-1).
      (0 (10.3528 5.32394 0.0))        Tọa độ các điểm tạo nên cửa sổ
      (0 (15.1899 5.32394 0.0))        chọn.
      (0 (15.1899 2.91549 0.0))
      (0 (10.3528 2.91549 0.0))
   )
)
```

Ví dụ: Đối tượng được chọn bằng đường cắt "F".

```
(
  (4 <Entity name: 34d0538> 0          ; 4: Chọn đối tượng bằng đường cắt
    (0 (12.5474 5.09797 0.0))          ;Đường cắt "F" cắt đối tượng tại 2
    (0 (11.2594 5.00562  0.0))         ; điểm
  )
)
```

Ví dụ: Chọn các đối tượng bằng nhiều phương pháp:

Command: **(setq SS (ssget))** ↵
Select objects: (chọn một đường thẳng bằng phương pháp Pick)

1 found
Select objects: **C** ↵
First corner: (Vẽ cửa sổ cắt chọn một Polyline)
Other corner: 1 found
Select objects: ↵
<Selection set: 275>

Command: **(ssnamex SS)** ↵
((1 <Entity name: 34d0528> 0 (0 (5.3 5.7 0.0))) (3 <Entity name: 34d0538> 1 -1) (-1 (0 (10.9892 5.41901 0.0)) (0 (15.4127 5.41901 0.0)) (0 (15.4127 2.75704 0.0)) (0 (10.9892 2.75704 0.0))))

Giải thích:

```
(
  (1 <Entity name: 34d0528> 0     ;đối tượng thứ nhất là đối tượng đơn
    (0 (5.3 5.7 0.0)))            ;điểm chọn trên đối tượng là '(5.3 5.7 0.0)
  (3 <Entity name: 34d0538> 1 -1) ;đối tượng thứ hai là đối tượng phức
    (-1                           ;chọn bằng phương pháp cửa sổ cắt.
      (0 (10.9892 5.41901 0.0))   ;Toạ độ các điểm tạo nên cửa sổ cắt...
      (0 (15.4127 5.41901 0.0))
      (0 (15.4127 2.75704 0.0))
      (0 (10.9892 2.75704 0.0))
    )
)
```

Ví dụ:

Dùng nhiều cửa sổ chọn. Mỗi cửa sổ chọn được xác định bằng một mã số.

Command: **(setq SS (ssget))**↵
Select objects: Other corner: 1 found
Select objects: Other corner: 1 found
Select objects:
<Selection set: 7>

Command: **(ssnamex SS)**↵
((2 <Entity name: 34d0528> 0 -1) (2 <Entity name: 34d0520> 0 -2) (-2 (0 (1.79227 4.75352 0.0)) (0 (4.40179 4.75352 0.0)) (0 (4.40179 3.96127 0.0)) (0 (1.79227 3.96127 0.0))) (-1 (0 (6.27937 6.36972 0.0)) (0 (7.74324 6.36972 0.0)) (0 (7.74324 5.26056 0.0)) (0 (6.27937 5.26056 0.0))))

Giải thích:

```
(
  (2 <Entity name: 34d0528> 0 -1) ;Đối tượng được chọn bởi cửa sổ chọn -1
  (2 <Entity name: 34d0520> 0 -2) ;Đối tượng được chọn bởi cửa sổ chọn -2
  (-2                             ;Cửa sổ chọn -2
    (0  (1.79227 4.75352 0.0))
    (0 (4.40179 4.75352 0.0))
    (0 (4.40179 3.96127 0.0))
    (0 (1.79227 3.96127 0.0))
  )
  (-1                             ;Cửa sổ chọn -1
    (0 (6.27937 6.36972 0.0))
    (0 (7.74324 6.36972 0.0))
    (0 (7.74324 5.26056 0.0))
    (0 (6.27937 5.26056 0.0))
  )
)
```

Tóm tắt:

1. Hàm **Ssgetfirst** dùng để xác định trạng thái GRIPS của các đối tượng được chọn.
2. Hàm **Sssetfirst** cho phép người sử dụng gán trạng thái GRIPS cho các đối tượng trong bản vẽ.
3. Hàm **Sslength** cho biết chiều dài của tập hợp chọn, tức là số phần tử trong tập hợp chọn.
4. Hàm **Ssname** cho biết tên đối tượng tại một vị trí trong tập hợp chọn. Vị trí được đánh số thứ tự từ 0.
5. Hàm **Ssnamex** cho biết tên đối tượng trong tập hợp chọn và chi tiết về việc chọn đối tượng đó.

12.2 Thêm, xóa và tìm các đối tượng trong tập hợp chọn

Sau khi đã tạo ra tập hợp chọn, ta có thể thêm, xóa các phần tử trong tập hợp này, hoặc kiểm tra một đối tượng có mặt trong tập hợp chọn hay không.

Hàm SSADD

Hàm **Ssadd** cho phép người sử dụng bổ sung thêm các đối tượng [ENAME] vào tập hợp chọn [SS]. Nhờ đó ta khỏi phải tạo thêm tập hợp chọn mới.

(**Ssadd** [ENAME [SS]])

Ví dụ:

(setq P (ssadd))	Hàm **Ssadd** không tham số tạo ra một tập hợp chọn mới không chứa phần tử nào.
(setq Q (ssget))	Tạo tập hợp chọn Q.
(setq R (ssadd (ssname Q 0)))	(ssname Q 0) là tên đối tượng đầu tiên trong tập hợp chọn Q. Hàm **Ssname** chỉ chứa tham số ENAME, không chứa tham số SS sẽ tạo ra tập hợp chọn mới chứa đối tượng ENAME.
(ssadd (ssname Q 0) P)	Thêm đối tượng đầu tiên trong tập hợp chọn Q bổ sung vào tập hợp P.
(ssadd (ssname P 5) L)	Vì tập hợp P ở trên chỉ có 1 phần tử, nên hàm **Ssname** gây ra lỗi.
(ssadd (ssname (ssget "P") 0) Q)	Thêm đối tượng đầu tiên trong tập hợp *Previous* bổ sung vào tập hợp Q.

Ví dụ:

Chương trình sau đây hiển thị tên tất cả các tập hợp chọn đang có trong bản vẽ. Sau đó yêu cầu người sử dụng chọn tập hợp muốn thêm phần tử, và chọn các đối tượng muốn thêm vào.

```
;Tên file:  ADDTOSET.LSP
;
(defun C:ATS (/ CTR ITEM LST SS CTR ENT DUMMY)
  (setvar "cmdecho" 0)
  (princ "\n\nWorking...")
```



```
(setq LST '())                                          ;[1]
  (foreach ITEM (atoms-family 0)                        ;[2]
    (if (= (type (eval ITEM)) 'PICKSET)                 ;[3]
      (setq LST (cons ITEM LST))                        ;[4]
    )                                                   ;Đóng if
  )                                                     ;Đóng foreach
(princ "Available selection-sets: \n\n")
(if LST (princ LST))
(cond
  (LST                                                  ;[5]
    (setq SS (strcase
               (getstring "\n\nAdd entities to which selection- set: "))
    )
    (if (member (read SS) LST)                          ;[6]
      (progn
        (setq SS (eval (read SS))                       ;[7]
          CTR 0                                         ;[8]
        )                                               ;Đóng setq
        (while (setq ENT (ssname SS CTR))               ;[9]
          (redraw ENT 3)                                ;[10]
          (setq CTR (1+ CTR))                           ;[11]
        )                                               ;Đóng while
;
;Thêm đối tượng vào tập hợp chọn SS.
;
        (princ "\nSet is highlighted ... Select entities to add:\n")
        (setq DUMMY   (ssget)                           ;[12]
              CTR 0
        )
        (while (setq ENT (ssname DUMMY CTR))            ;[13]
              (ssadd ENT SS)                            ;[14]
              (setq CTR (1+ CTR))                       ;[15]
        )
        (princ (strcat "\nThis selection set now has "
                       (itoa (sslength SS))
                       " data elements.")
                                                        ;Đóng princ
        )
        (setq DUMMY  nil
              CTR 0
        )
        (while (setq ENT (ssname SS CTR))  ;Khi vẫn còn đối tượng
```

```
                                        ;trong SS...
            (redraw ENT 4)              ;thì unhighlight các đối tượng
                                        ;này
            (setq CTR (1+ CTR))
          )                             ;Đóng while
        )                               ;Đóng progn
        (princ "\nNot a selection set!")
      )                                 ;Đóng if
    );Đóng 1st cond list
    (T                                  ;[16]
      (princ "\nThere are no selection set currently available.")
    )                                   ;Đóng 2nd cond list
  )                                     ;Đóng cond

  (redraw)
  (setvar "cmdecho" 1)
  (princ)
)
```

Giải thích:

[1] **(setq LST '())** Khởi tạo biến LST là danh sách rỗng.

[2] **(foreach ITEM (atoms-family 0)** Hàm **Foreach** dùng biến ITEM để duyệt danh sách **Atoms-family**. Biểu thức **(atoms-family 0)** trả về danh sách chứa tất cả các hàm và các nguyên tố (*atom*) của AutoLISP.

[3] **(if (= (type (eval ITEM)) 'PICKSET)** Biến ITEM được đinh giá trị. Tập hợp chọn có kiểu dữ liệu là PICKSET. Nếu ITEM là tập hợp chọn, thì ...

[4] **(setq LST (cons ITEM LST))** Tạo danh sách LST chứa các tập hợp chọn ITEM.

[5] **(cond (LST** Nếu LST có chứa giá trị (trả về True) thì thực hiện hàm **If** theo sau.

[6] **(if (member (read SS) LST)** READ tham số SS (chứa tên tập hợp chọn do người sử dụng nhập vào). Hàm **Read** trả về kiểu dữ liệu là nguyên tố, phù hợp với hàm **Member**. Nếu có trong danh sách LST, thì

[7] **(setq SS (eval (read SS))** Gán lại nội dung chuỗi SS cho chính xác để dùng cho hàm **Ssname**.

[8] **(setq CTR 0)** Khởi tạo biến đếm CTR bằng 0 để dùng cho vòng lặp **While**.

[9] **(while (setq ENT (ssname SS CTR))** Các phần tử trong tập hợp chọn SS lần lượt được lấy ra gán cho biến ENT. Vòng lặp kết thúc khi không còn phần tử trong SS.

[10] **(redraw ENT 3)** Hàm **Redraw** làm đối tượng lưu trong biến ENT nổi bật trên màn hình.

[11] **(setq CTR (1+ CTR))** Tăng biến đếm CTR lên 1 dùng cho lần lặp tiếp theo của vòng **While**. Bằng cách này tất cả các đối tượng trong SS đều được làm nổi bật trên màn hình so với các đối tượng khác.

[12] **(setq DUMMY (ssget))** Dùng biến DUMMY để lưu các đối tượng do người sử dụng chọn thêm vào tập hợp chọn SS.

[13] **(while (setq ENT (ssname DUMMY CTR))** Lấy ra từng đối tượng trong biến DUMMY gán cho biến ENT.

[14] **(ssadd ENT SS)** Thêm các đối tượng trong ENT vào SS

[15] **(setq CTR (1+ CTR))** Tăng biến đếm CTR thêm 1 dùng cho hàm **While**. Nhờ đó tất cả các đối tượng trong DUMMY được thêm vào SS.

[16] **(T (princ "\nThere are no selection set currently available."))** Nếu hàm **Cond** đầu tiên trả về nil, thì thực hiện biểu thức này. Hàm **Princ** thông báo cho người sử dụng không có tập hợp chọn nào được tạo ra ban đầu.

Hàm SSDEL

Hàm **Ssdel** (*Selection Set DELete*) dùng để xóa đối tượng ENAME trong tập hợp chọn SS.

(**Ssdel** ENAME SS)

Ý nghĩa các tham số ENAME, SS giống như hàm **Ssadd**, nhưng tất cả bắt buộc phải có.

✌ **Ví dụ:**

(setq R (ssget))	Tạo tập hợp chọn R
(sslength R)	Có 15 phần tử trong R
15	
(ssdel (ssname R 0) R)	Xóa phần tử đầu tiên trong R

(ssdel (ssname R 22) R) — Vì R có số phần tử nhỏ hơn 22, nên hàm **Ssname** trả về nil. Toàn bộ biểu thức **Ssdel** trả về lỗi.

Hàm SSMEMB

Hàm **Ssmemb** (*Selection Set MEMBer*) kiểm tra phần tử ENAME có trong tập hợp chọn SS hay không. Nếu có, hàm trả về tên ENAME này; nếu không, trả về nil.

(**Ssmemb** ENAME SS)

Ví dụ:

```
(setq J (ssget "C" '(0 0) '(297 210)))      Tạo tập hợp chọn J

(setq ENT1 (ssname J 0))                    Tên của phần tử đầu tiên
<Entity name: 85 c0500>

(ssmemb ENT1 J)                             Trả về tên chứa trong ENT1
<Entity name: 85 c0500>

(setq ENT2 (ssname R 0))
<Entity name: 85 c0508>
(ssmemb ENT2 J)                             Trả về nil vì ENT2 không có trong
nil                                         tập hợp chọn J
```

Tóm tắt:

1. Các hàm **Ssadd** và **Ssdel** dùng để thêm hoặc xóa các phần tử trong các tập hợp chọn. Đây là phương pháp hiệu chỉnh các tập hợp chọn.
2. Hàm **Ssmemb** dùng để kiểm tra đối tượng co trong tập hợp chọn hay không. Nếu có, sẽ trả về tên đối tượng này.

12.3 Các ví dụ mẫu

Ví dụ 1:

Chương trình đếm số lượng các khối được chen vào bản vẽ.

```
;Tên file: BLKCOUNT.LSP
(defun C:BLKCOUNT (/BLKNAME SS NUM)
  (prompt "\nBLKCOUNT - Count the number of times a block is
      inserted")
```



```
  (setq BLKNAME (getstring "\nEnter the name
      of the block to count: "))              ;Tên block
  (setq SS (ssget "X" (list (cons 0 "INSERT")  ;SS chứa các khối chèn
                            (cons 2 BLKNAME)  ;và có tên là BLKNAME
                      )                       ;Đóng list
           )                                  ;Đóng ssget
  )                                           ;Đóng setq
  (if (null SS)
      (setq NUM 0)                            ;NUM chứa số lượng khối
      (setq NUM (sslength SS))                ;đã chèn
  )                                           ;Đóng if
  (prompt                                     ;Hiện kết quả
    (strcat "\nThere are " (itoa NUM)
        " insertions of " BLKNAME ".")
  )
  (prompt "\nProgram complete. ")
  (princ)
)
```

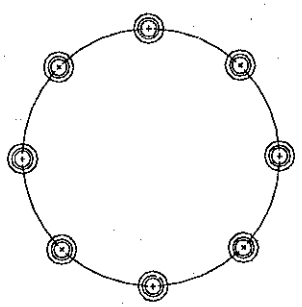
Ví dụ 2: Lập chương trình vẽ các lỗ tròn với dấu tâm hoặc đường tâm nằm cách đều chung quanh một điểm. Các lỗ này có thể là lỗ suốt, lỗ khoét trụ hoặc lỗ khoét côn.

```
;Tên file: BOLTCIR.LSP
(defun C:BOLTCIR (/ NH DH TH CB CP RP RA CX CY CML CM ANG HP
OH EX)
  (setvar "Cmdecho" 0)
  (setvar "Highlight" 0)
  (setq NH (getint  "\nNumber of holes: ")
  DH (getreal "\nDIA of holes: ") )
  (initget "Yes No")
  (setq TH (getkword   "\nTapped holes? yes or <No>: ")
    CB (getreal    "\nDIA of CBORE or CSK <none>: ")
    CP (getpoint   "\nCenter point of bolt circle: ")
    RP (getdist CP "\nRadius of bolt circle: ")
    RA (getreal    "\nStarting angle <0 Degrees>: ")
  )
(command "layer" "new" "lines,center,hidden" "ltype" "center" "center" "ltype"
"hidden" "hidden" "")
  (setq K   (getvar "Dimscale")
    CM  (getvar "Dimcen")
    CML (abs CM)
```

```
  CX  (car CP)
  CY  (cadr CP)
  HP  (list (+ CX RP) CY)
  ANG (angle CP (list (+ CX RP) (+ CY (* K CML))))
 )
 (if (/= CB nil)
  (setq OH CB)
  (setq OH DH)
 )
 (command "Layer" "Set" "Center" ""
  "Line" (list (+ CX (- RP (* K CML))) CY)
 (list (+ CX (+ RP (* K CML))) CY) "")
 (setq SS (ssadd (entlast)))
 (if (< (* K (* 4.0 CML)) OH)
  (setq EX (/ (- OH (* K (* 4.0 CML))) 2.0))
  (setq EX 0.0)
 )
 (if (and (< (* K (* 4.0 CML)) OH) (< CM 0.0))
  (progn
   (command "Line" (list (+ CX (- (- RP (* K (* 3.0 CML))) EX)) CY)
   (list (+ CX (- RP (* K (* 2.0 CML)))) CY) "")
   (ssadd (entlast) SS)
   (command "Line" (list (+ CX (+ RP (* K (* 2.0 CML)))) CY)
   (list (+ CX (+ (+ RP (* K (* 3.0 CML))) EX)) CY) "")
   (ssadd (entlast) SS)
  )
  ()
 )
 (command "Arc" (polar CP (- 0.0 ANG) RP)
 "E" (polar CP ANG RP) "R" RP)
 (ssadd (entlast) SS)
 (command "Arc" (polar CP (* ANG 2.0) RP)
 "E" (polar CP (- (/ (* 2.0 pi) NH) (* 2.0 ANG)) RP) "R" RP)
 (ssadd (entlast) SS)
 (if (or (= TH "No") (= TH nil))
  (progn
   (command "Layer" "Set" "Lines" "" "Circle" HP (/ DH 2.0))
   (ssadd (entlast) SS)
  )
  (progn
   (command "Layer" "Set" "Hidden" "" "Circle" HP (/ DH 2.0))
```



```
   (ssadd (entlast) SS)
   (command "Layer" "Set" "Lines" "" "Circle" HP(/ (/ DH 2.0) 1.333333))
   (ssadd (entlast) SS)
  )
 )
 (if (/= CB nil)
  (progn (command "Circle" HP (/ CB 2.0)) (ssadd (entlast) SS))
  ()
 )
 (if (/= RA nil)
  (command "Rotate" SS "" CP RA)
  ()
 )
 (command "Array" SS "" "Polar" CP NH "360" "Yes")
 (setvar "Highlight" 1)
 (princ)
) ; Kết thúc chương trình BOLTCIR.LSP
```



12.4 Bài tập

1. Viết các biểu thức chứa hàm **Ssget** thực hiện các công việc sau:

a. Gán tập hợp chọn chứa tất cả các đường tròn có bán kính bằng 50 cho biến VAR1. (Gợi ý: Dùng *code* 40 cho bán kính).

b. Gán tập hợp chọn chứa tất cả các block có tên "PART" thuộc lớp 0 cho biến BLK.

c. Gán tập hợp chọn chứa tất cả các đường thẳng có màu đỏ (màu số 1) cho biến RL. Chỉ chọn các đối tượng trong giới hạn bản vẽ bằng cach dùng cửa sổ cắt.

d. Gán tập hợp chọn Previous chứa tất cả các dòng chữ (text) có kiểu chữ "Simplex" cho biến TX. (Gợi ý: Dùng *code* 7 cho kiểu chữ).

2. Viết các biểu thức chứa hàm **Ssget** thực hiện các công việc sau. Cố gắng sử dụng các phép toán so sánh và logic.

a. Gán cho biến PNT tập hợp chọn chứa tất cả các điểm (POINT) có tọa độ X, Y lớn hơn giá trị biến LIMMAX. (Gợi ý: Sử dụng *code* 10 cho tọa độ điểm).

b. Gán cho biến VP tập hợp chọn chứa tất cả các khung nhìn đông trong không gian giấy vẽ có các tọa độ X nhỏ hơn 50.0 và các tọa độ Y nhỏ hơn 40.0 (Gợi ý: dùng *code* 40 cho X và *code* 41 cho Y).

c. Gán cho biến CIR tập hợp chọn chứa tất cả các đường tròn có bán kính nhỏ hơn hoặc bằng 50 và không thuộc lớp 0. (Gợi ý: dùng *code* 40 cho bán kính).

3. Viết chương trình BLOCKHGT.LSP tạo tập hợp chọn chứa tất cả các block của bản vẽ. Sau đó dùng hàm **Redraw** để trên màn hình lần lượt phân biệt từng block. Nếu block có chứa dòng chữ (text) hoặc thuộc tính (attribute), hãy hỏi người sử dụng có xóa block này ra khỏi tập hợp chọn hay không. Sau khi duyệt hết tập hợp chọn và xóa các block do người sử dụng chọn, hãy xóa tập hợp chọn tạo ra. Hãy làm cho tất cả các lớp bản vẽ đều thấy được (ON) trước khi thực hiện chương trình. Dùng chương trình TEXTHGT.LSP để làm mẫu cho bài tập này.

4. Thông thường các kích thước bị nằm rải rác ở các lớp bản vẽ. Hãy viết chương trình tạo ra tập hợp các đối tượng kích thước và chuyển chúng sang một lớp bản vẽ mới. Chương trình sẽ hỏi người sử dụng tên lớp bản vẽ mới. Kiểu dữ liệu của kích thước là "DIMENSION".

5. Tạo một hình chữ nhật có kích thước 10x30 bằng lệnh **Line** của **AutoCAD**. Sau đó ghi dòng chữ "DESK" tại tâm hình chữ nhật, có chiều cao là 5, độ nghiêng là 0. Nhóm các đối tượng này thành một block có tên "DESK", có điểm chèn tại một đỉnh của hình chữ nhật. Chèn block này 15 lần tại các vị trí bất kỳ của lớp 0, có các hệ số chèn (X = Y = Z) từ 1 đến 5. Viết chương trình **AutoLISP** thực hiện các công việc sau:

a. Tạo tập hợp chọn SS1 chứa tất cả các đối tượng bản vẽ.

b. Tạo tập hợp chọn SS2 chứa tất cả các block có tên "DESK", với hệ số chèn từ 1 đến 3.

c. Xóa tất cả các đối tượng, chỉ để lại các đối tượng của tập hợp SS2.

d. Tạo lớp bản vẽ "FURNITURE" và chuyển tất cả các block trong SS2 sang lớp này.

6. Vẽ 7 đường tròn có bán kính 10 và 3 đường tròn có bán kính. Sau đó viết chương trình thực hiện các công việc sau:

a. Tạo tập hợp chọn SS1 chứa tất cả các đường tròn có bán kính khác 10.

b. Duyệt từng phần tử trong SS1, và hỏi người sử dụng có thay đổi bán kính hay không.

c. Dùng lệnh **Change** của **AutoCAD** để thay đổi bán kính.

12.5 Lời giải

1.

a. (setq VAR1 (ssget "X" '((0 . "CIRCLE") (40 . 50))))

b. (setq BLK (ssget "X" '((2 . "PART") (8 . "0"))))

c. (setq RL (ssget "C" (getvar "LIMMIN") (getvar "LIMMAX")
'((0 . "LINE") (62 . 1))))

d. (setq TX (ssget "X" '((0 . "TEXT") (7 . "Simplex"))))

2.

a.

```
(setq PNT (ssget "X" (list '(0 . "point")
                           '(-4 . ">,>")
                           (list 10
                                 (car (getvar "LIMMAX"))
                                 (cadr (getvar "LIMMAX"))
                           )
                     )
          )                          ;Đóng ssget
)                                    ;Đóng setq
```

b.

```
(setq VP (ssget "X" '((-4 . "<AND")
                      (0 . "VIEWPORT")
                      (67 . 1)
                      (-4 . "<")(40 . 50.0)
                      (-4 . "<")(41 . 40.0)
                      (-4 . "AND>")
                     )                ;Đóng filter-list
          )                           ;Đóng ssget
)                                     ;Đóng setq
```

c.

```
(setq CIR (ssget "X" '( (-4 . "<AND")
                        (0 . "CIRCLE")
```

```
                    (-4 . "<NOT")(8 . "0")(-4 . "NOT>")
                    (-4 . "<=")(40 . 50)
                    (-4 . "AND>")
                  )                    ;Đóng filter-list
            )                          ;Đóng ssget
    )                                  ;Đóng setq
```

3.

```
;Tên file: BLOCKHGT.LSP
(defun C:BLOCKHGT (/ BLKSET ENT CTR ANS)
  (setvar "cmdecho" 0)
                              ;Mở tất các các lớp
  (command "-layer" "ON" "*" "")
                              ;Vòng lặp cho tất cả blocks
  (setq BLKSET (ssget "X" '((0 . "INSERT"))))
  (if (null BLKSET)
     (princ (strcat "\nNo blocks in drawing!"))
     (progn
          (setq CTR 0)
          (while (setq ENT (ssname BLKSET CTR))    ;While
          (redraw ENT 3)
          (setq ANS (strcase
               (getstring "\nDelete this block <Y>: "))
          )                                        ;Đóng setq
          (if (or (= ANS "Y") (= ANS ""))
               (ssdel ENT BLKSET)
          )                                        ;Đóng if
          (redraw ENT 4)
          (setq CTR (1+ CTR))
        )                                  ;Đóng while
     )                                     ;Đóng prog
  )                                        ;Đóng if
  (setq BLKSET null)
  (GC)
  (setvar "cmdecho" 0)
  (princ)
)                                          ;Kết thúc chương trình
```

4.

```
;Tên file: LAY-DIM.LSP
(defun C:LAY-DIM (/ BLKSET NEWLAY)
```



```
  (setvar "cmdecho" 0)
;Layer mới
  (setq NEWLAY (getstring "\nEnter new layer for dimensions <DIM>: ")
       NEWLAY (if (= NEWLAY "") "DIM" NEWLAY)
  )
  (command "-layer" "N" NEWLAY "")
;Thay đổi layer
  (setq DIMSET (ssget "X" '((0   "DIMENSION"))))
  (command ".chprop" DIMSET "" "LA" NEWLAY "")
   (setvar "cmdecho" 0)
   (princ)
)
```

5.

```
;Tên file: DESK.LSP
(defun C:DESK (/ SS1 SS2)
   (setvar "cmdecho" 0)
   (setq SS1 (ssget "X" ))
   (setq SS2 (ssget "X" '((-4 . "<AND")
                          (0 . "INSERT") (2 . "DESK")
                          (-4 . ">=")(41 . 1)
                          (-4 . "<=")(41 . 3)
                          (-4 . "AND>")
                              )                    ;Đóng filter-list
              )                                    ;Đóng ssget
     )                                             ;Đóng setq
     (command ".erase" SS1 "remove" SS2 "")
     (command "-layer" "N" "FURNITURE" "")
     (command ".chprop" "all" "" "LA" "FURNITURE" "")
     (setvar "cmdecho" 0)
     (princ)
)
```

6.

```
;Tên file: CIR.LSP
(defun C:CIR (/ SS1 CTR ENT ANS)
  ; (setvar "cmdecho" 0)
     (setq SS1 (ssget "X" '( (0 . "CIRCLE")
                             (-4 . "<>")(40 . 10)
                           )
              )
```

```
    )
    (setq CTR 0)
    (while (setq ENT (ssname SS1 CTR))          ;while there are items
      (redraw ENT 3)                            ;highlight block
      (setq ANS (strcase
                  (getstring "\nChange radius <Y>: "))
      );Đóng setq
      (if (or (= ANS "Y") (= ANS ""))
            (progn
            (princ "\nSpecify a point: ")
            (command ".change" ENT "" pause)
          );Đóng progn
      )                                         ;Đóng if

      (redraw ENT 4)                            ;dehighligh text
      (setq CTR (1+ CTR))
    )                                           ;Đóng while
;
  (setvar "cmdecho" 0)
  (princ)
)
```

Phụ lục 1

BẢNG TRA CỨU HÀM AutoLISP

Tên hàm	Ý nghĩa	Trang
(- [NUMBER NUMBER] . . .)	Trả về hiệu tham số thứ nhất với tổng các tham số còn lại.	16
(* [NUMBER NUMBER] . . .)	Trả về tích các tham số.	16
(/ [NUMBER NUMBER] . . .)	Chia tham số thứ nhất cho tích các tham số còn lại và trả về kết quả phép chia.	17
(/= ATOM ATOM ...)	trả về T nếu từng phần tử khác với phần tử đứng bên phải nó, ngược lại, hàm trả về nil.	134
(+ [NUMBER NUMBER] . . .)	Trả về tổng các tham số này.	15
(< ATOM ATOM ...)	Trả về T nếu mỗi phần tử nhỏ hơn phần tử đứng bên phải nó.	134
(<= ATOM ATOM ...)	Trả về T nếu mỗi phần tử nhỏ hơn hoặc bằng phần tử đứng bên phải nó.	134
(= ATOM ATOM ...)	Trả về giá trị T nếu tất cả các phần tử bằng nhau.	132
(> ATOM ATOM ...)	Trả về T nếu mỗi phần tử lớn hơn phần tử đứng bên phải nó.	134
(**1-** NUMBER)	Trừ tham số NUMBER đi 1 và trả về kết quả này.	162
(**1+** NUMBER)	Cộng tham số NUMBER thêm 1 và trả về kết quả này.	162
(**Abs** NUMBER)	Trả về giá trị tuyệt đối của một số.	101
(**Acad_Colordlg** COLORNUM [FLAG])	Làm xuất hiện hộp thoại *Select Color* của **AutoCAD** để người sử dụng chọn màu.	71
(**Acad_Strlsort** LIST)	Nhận tham số là danh sách gồm các phần tử là chuỗi và trả về danh	125

	sách đã được sắp xếp theo thứ tự ABC.	
(**And** EXPRESSION ...)	Nếu tất cả các tham số có giá trị khác *nil*, hàm **And** trả về *T*. Ngược lại, nếu có một tham số bằng *nil*, hàm trả về *nil*.	142
(**Angle** PT1 PT2)	Trả về góc đo (tính bằng radian) trong mặt phẳng x-y, tạo bởi đường thẳng qua hai điểm với trục x.	81
(**Angtof** STRING [MODE])	Chuyển đổi một chuỗi chứa số đo góc thành một số thực.	116
(**Angtos** ANGLE [MODE [PRECISION]])	Chuyển đổi số đo một góc thành một chuỗi.	116
(**Append** EXPR ...)	Gộp nhiều danh sách thành một danh sách duy nhất.	161
(**Apply** FUNCTION LIST)	Thực hiện hàm FUNCTION với tham số là các phần tử trong danh sách LIST.	198
(**Ascii** STRING)	Chuyển đổi ký tự đầu tiên trong một chuỗi thành mã ký tự ASCII tương ứng và trả về mã này.	117
(**Assoc** ITEM ALIST)	Trả về một danh sách con trong danh sách phức hợp.	188
(**Atan** NUM1 [NUM2])	Trả về giá trị arctang của một góc.	104
(**Atof** STRING)	Chuyển đổi một chuỗi thành một số thực.	112
(**Atoi** STRING)	Chuyển đổi một chuỗi thành một số nguyên.	113
(**Atom** ITEM)	Kiểm tra dữ liệu có phải là *nguyên tố (atom)* hay không.	135
(**Boundp** ATOM)	Trả về T nếu tham số là nguyên tố và nó được gắn với một giá trị. Ngược lại, sẽ trả về nil.	137
(**Caddr** LIST)	Trả về phần tử thứ ba trong danh sách.	55
(**Cadr** LIST)	Trả về phần tử thứ hai trong danh sách.	55
(**Car** LIST)	Trả về phần tử đầu tiên của danh sách.	52
(**Cdr** LIST)	Trả về danh sách không chứa phần tử đầu tiên.	52
(**Chr** INTEGER)	Chuyển đổi mã ASCII thành ký tự tương ứng trong bảng mã ASCII.	117
(**Command** [ARGUMENT] . . .)	Thực hiện một hoặc nhiều lệnh của	37

	AutoCAD.	
(**Cond** (TEST1 RESULT1 ...) (TEST2 RESULT2 ...) ...)	Lần lượt định giá trị các biểu thức TEST, cho đến khi phát hiện biểu thức TEST đầu tiên khác nil thì dừng lại và thực hiện các biểu thức RESULT tương ứng.	146
(**Cons** NEW-FIRST-ELEMENT LIST)	Thêm tham số NEW-FIRST-ELEMENT vào đầu danh sách LIST.	189
(**Cos** ANGLE)	Trả về giá trị cosin của một góc.	104
(**Cvunit** VALUE FROM TO)	Chuyển đổi một giá trị hoặc tọa độ một điểm từ đơn vị đo này sang đơn vị đo khác.	80
(**Defun** FUNCTION_NAME ARGUMENT_LIST EXPRESSION ...)	Định nghĩa một hàm tự tạo có tên chứa trong tham số FUNCTION_NAME.	40
(**Distance** PT1 PT2)	Trả về khoảng cách giữa hai điểm (2D hoặc 3D).	81
(**Distof** STRING [MODE])	Chuyển đổi một chuỗi thành một số thực.	113
(**Eq** EXPR1 EXPR2)	So sánh sự trùng nhau giữa hai danh sách.	133
(**Equal** EXPR1 EXPR2 [FUZZ])	So sánh giá trị các biểu thức với sai lệch cho phép.	132
(**Eval** EXPR)	Định giá trị biểu thức EXPR cho đến mức cuối cùng, loại bỏ các kết quả ở các mức trung gian.	165
(**Exp** NUMBER)	Trả về giá trị lũy thừa e^n	105
(**Expt** BASE POWER)	Trả về lũy thừa của một số.	105
(**Fix** NUMBER)	Trả về phần nguyên một số.	100
(**Float** NUMBER)	Chuyển một số có kiểu số nguyên hoặc số thực thành kiểu số thực.	100
(**Foreach** NAME LIST EXPR ...)	Duyệt từng phần tử trong danh sách LIST. Tại mỗi thời điểm, giá trị của từng phần tử trong danh sách được gán tạm thời cho biến NAME. Sau đó các biểu thức EXPR được định giá trị.	164
(**Gcd** NUMBER1 NUMBER2)	Trả về ước số chung lớn nhất của hai số nguyên.	102
(**Getangle** [PT] [PROMPT])	Yêu cầu nhập vào giá trị góc đo, hoặc chọn hai điểm trên màn hình bằng cách nhấn chuột. Hàm **Getangle** trả về góc đo tính bằng	83

	radian.	
(**Getcorner** PT [PROMPT])	Yêu cầu nhập một điểm. Nếu sử dụng tham số PT, sẽ xuất hiện cửa sổ động trên màn hình.	56
(**Getdist** [PT] [PROMPT])	Yêu cầu nhập vào giá trị khoảng cách, hoặc chọn hai điểm trên màn hình bằng cách nhấn chuột để tính khoảng cách.	87
(**Getint** [PROMPT])	Yêu cầu nhập vào một số nguyên.	64
(**Getkword** [PROMPT]	Yêu cầu nhập dữ liệu ở dạng từ khóa.	69
(**Getorient** [PT] [PROMPT])	Trả về số đo một góc dựa theo các khai báo về đường chuẩn ANGBASE = 0 và hướng đo góc ANGDIR = 0.	85
(**Getpoint** [PT] [PROMPT])	Yêu cầu người sử dụng nhập vào tọa độ một điểm. Khi dùng tham số PT sẽ xuất hiện *sợi dây thun* trên màn hình.	34
(**Getreal** [PROMPT])	Yêu cầu nhập vào một số thực.	65
(**Getstring** [PROMPT])	Yêu cầu nhập vào một chuỗi.	66
(**Getvar** VARNAME)	Trả về giá trị hiện hành của biến hệ thống **AutoCAD**.	70
(**If** TESTexpr THENexpr [ELSEexpr])	Nếu biểu thức điều kiện TESTexpr đúng (trả về T) thì biểu thức THENexpr sẽ được thực hiện. Ngược lại, nếu có biểu thức ELSEexpr thì biểu thức này sẽ được thực hiện.	139
(**Initget** [BITS] [STRING])	Cung cấp danh sách các giá trị nhập hợp lệ bằng cách gán các *bit kiểm tra (bit code)* và danh sách các từ khóa.	67
(**Inters** PT1 PT2 PT3 PT4 [ONSEG])	Trả về giao điểm giữa hai đường thẳng không song song.	94
(**Itoa** INTEGER)	Chuyển đổi một số nguyên thành một chuỗi.	115
(**Lambda** ARGUMENTS EXPR ...)	Định nghĩa hàm không tên.	200
(**Last** LIST)	Trả về phần tử cuối cùng của danh sách.	56
(**Length** LIST)	Trả về số lượng các phần tử có trong danh sách.	56
(List EXPRESSION...)	Tạo danh sách	50

(**Listp** ITEM)	Kiểm tra dữ liệu có phải là kiểu danh sách hay không.	136
(**load** FILENAME ONFAILURE])	Tải file chương trình AutoLISP hoặc trả về ONFAILURE nếu bị lỗi.	30
(**Log** NUMBER)	Trả về giá trị logarit của một số.	105
(**Mapcar** FUNCTION LIST1 ... LISTN)	Thực hiện hàm FUNCTION nhiều lần, mỗi lần sử dụng một danh sách các tham số khác nhau.	199
(**Max** NUMBER NUMBER ...)	Trả về giá trị lớn nhất trong các số.	102
(**Member** EXPR LIST)	Trả về một danh sách bắt đầu bằng phần tử EXPR nếu nó được tìm thấy trong danh sách LIST.	190
(**Min** NUMBER NUMBER ...)	Trả về giá trị nhỏ nhất trong các số.	102
(**Minusp** ITEM)	Kiểm tra dữ liệu có phải là số âm hay không.	136
(**Not** ITEM)	Trả về *T* nếu tham số của nó có giá trị *nil*. Ngược lại, trả về *nil*.	143
(**Nth** N LIST)	Lấy ra phần tử ở vị trí thứ N trong danh sách LIST.	193
(**Null** ITEM)	Kiểm tra một biến hoặc danh sách có rỗng hay không.	137
(**Numberp** ITEM)	Kiểm tra dữ liệu có phải là kiểu số hay không.	136
(**Or** EXPRESSION ...)	Nếu tất cả các tham số đều bằng *nil*, hàm **Or** trả về giá trị *nil*. Ngược lại, nếu có một tham số khác *nil*, hàm sẽ trả về T.	143
(**Osnap** PT MODE-STRING)	Cung cấp chức năng truy bắt đối tượng khi chọn điểm và trả về điểm 3D.	89
(**Polar** PT ANGLE DISTANCE)	Dùng tọa độ cực để tạo ra điểm (2D) mới từ điểm ban đầu.	82
(**Prin1** [EXPR [FILE-DESC]])	In dữ liệu chứa trong tham số EXPR lên màn hình hoặc in thành file và trả về kết quả là tham số này.	119
(**Princ** [EXPR [FILE-DESC]])	In dữ liệu chứa trong tham số EXPR lên màn hình hoặc in thành file và trả về kết quả là tham số này.	120
(**Print** [EXPR [FILE-DESC]])	Tự động xuống dòng, sau đó in dữ liệu chứa trong tham số EXPR lên màn hình hoặc ra một file và trả về kết quả là tham số này.	120

(**Progn** EXPRESSION ...)	Nhóm nhiều biểu thức thành một biểu thức duy nhất.	141
(**Prompt** MESSAGE)	Hiện dữ liệu kiểu *chuỗi* chứa trong tham số MESSAGE.	36
(**Read** STRING)	Trả về phần tử đầu tiên chứa trong một chuỗi và chuyển đổi phần tử này về kiểu dữ liệu tương ứng.	118
(**Rem** NUMBER NUMBER ...)	Trả về số dư của phép chia.	101
(**Repeat** NUMBER EXPR ...)	Định giá trị các biểu thức EXPR với số lần là NUMBER.	158
(**Reverse** LIST)	Trả về danh sách theo thứ tự ngược lại.	192
(**Rtos** NUMBER [MODE [PRECISION]])	Chuyển đổi một số thành một chuỗi.	114
(**Set** SYM EXPR)	Hàm **Set** gán giá trị của biểu thức EXPR cho biến chứa trong tham số SYM.	164
(**Setq** SYMBOL1 VALUE1 [SYMBOL2 VALUE2] ...)	Gán giá trị cho biến và trả về kết quả là giá trị này.	20
(**Setvar** VARNAME VALUE)	Gán giá trị cho biến hệ thống **AutoCAD**.	71
(**Sin** ANGLE)	Trả về giá trị sin của một góc.	103
(**Sqrt** NUMBER)	Trả về căn bậc hai của một số.	105
(**Ssadd** [ENAME [SS]])	Thêm đối tượng [ENAME] vào tập hợp chọn [SS].	239
(**Ssdel** ENAME SS)	Xóa đối tượng ENAME trong tập hợp chọn ψ.	243
(**Ssget** [MODE] [PT1] [PT2]] [PT-LIST] [FILTER-LIST])	Tạo ra một tập hợp chọn gồm các đối tượng do người sử dụng chọn trên màn hình.	214
(**Ssgetfirst**)	Trả về một danh sách chứa hai tập hợp chọn. Tập hợp thứ nhất chứa các đối tượng có GRIPS ở trạng thái COLD. Tập hợp thứ hai chứa các đối tượng có GRIPS ở trạng thái WARM.	229
(**Sslength** SS)	Trả về số phần tử trong tập hợp chọn.	232
(**Ssmemb** ENAME SS)	Kiểm tra phần tử ENAME có trong tập hợp chọn SS hay không.	243
(**Ssname** SS INDEX)	Trả về tên đối tượng tại vị trí INDEX trong tập hợp chọn SS.	233
(**Ssnamex** SS [INDEX])	Trả về một danh sách chứa thông tin mô tả đối tượng và phương pháp	235

	chọn đối tượng này.	
(**Sssetfirst** [GRIPSET] [PICKSET])	Gán trạng thái GRIPS cho các đối tượng trong bản vẽ.	231
(**Strcase** STRING [WHICH])	Biến đổi các ký tự trong chuỗi STRING thành các chữ hoa hoặc chữ thường.	122
(**Strcat** STRING1 [STRING2] ...)	Kết nối nhiều chuỗi thành một chuỗi duy nhất.	122
(**Strlen** [STRING] ...)	Trả về chiều dài của một chuỗi (số ký tự của chuỗi).	124
(**Substr** STRING START [LENGTH])	Trả về một chuỗi con trong tham số STRING.	124
(**Type** ITEM)	Trả về kiểu dữ liệu kết nối với tham số ITEM.	138
(**While** TESTEXPR EXPR ...)	Khi TESTEXPR vẫn còn trả về giá trị khác nil, các biểu thức EXPR sẽ tiếp tục được đinh giá trị. Quá trình này được thực hiện cho đến khi TESTEXPR trả về giá trị nil.	159
(**Zerop** ITEM)	Kiểm tra dữ liệu có phải là số 0 hay không.	137

CÁC BIẾN HỆ THỐNG AutoCAD

Các biến hệ thống (*system variables*) của **AutoCAD** lưu trữ các giá trị xác định môi trường làm việc của **AutoCAD** và các file bản vẽ.

Một số biến hệ thống cho phép ta đọc và gán giá trị trực tiếp cho nó. Một số biến *chỉ đọc* được mà không thể gán giá trị trực tiếp cho nó được.

Kiểu dữ liệu: Chuỗi, số nguyên, số thực, logic, tọa độ điểm 2D, tọa độ điểm 3D...

Các biến thường được gán *giá trị mặc định ban đầu* khi cài đặt **AutoCAD** hoặc khi tạo bản vẽ mới.

Một số biến lưu lại giá trị của mình khi bản vẽ hoặc **AutoCAD** được đóng lại. Một số biến không lưu lại giá trị của mình.

Nơi lưu trữ: file bản vẽ hoặc file Registry của Windows.

Các hàm **Getvar** và **Setvar** của **AutoLISP** dùng để đọc và gán giá trị cho các biến hệ thống.

Tên biến	Ý nghĩa và giá trị

A

ACADPREFIX

Kiểu: Chuỗi.
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Chứa đường dẫn thư viện của **AutoCAD**.

ACADVER

Kiểu: Chuỗi.
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Chứa giá trị 14 hoặc 14a biểu diễn phiên bản **14** của **AutoCAD**. Đối với **AutoCAD** 2000 sẽ hiển thị 15.0.

ACISOUTVER

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 16
Quy định phiên bản ACIS của các file SAT tạo ra khi dùng lệnh **Acisout**.

AFLAGS

Kiểu: Số nguyên.
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Số nguyên biểu diễn đặc điểm thuộc tính của khối (ATTDEF). Là tổng của các giá trị sau:

0	No attribute mode selected
1	Invisible
2	Constant
4	Verify
8	Preset

ANGBASE

Kiểu: Số thực.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Hướng đường chuẩn xác định góc. Các giá trị thường sử dụng:

0	=	East	3 o'clock
90	=	North	12 o'clock
180	=	West	9 o'clock
270	=	South	6 o'clock

ANGDIR

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Xác định chiều đo góc. Các giá trị như sau:

0	Ngược chiều kim đồng hồ
1	Cùng chiều kim đồng hồ

APBOX

Kiểu: Số nguyên.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Hiển thị hoặc không hiển thị ô vuông truy bắt (*aperture box*) tại giao điểm 2 sợi tóc (*crosshairs*).

0	Không hiển thị ô vuông truy bắt
1	Hiển thị ô vuông truy bắt

APERTURE

Kiểu: số nguyên.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 10
Chiều cao của ô vuông truy bắt. Đơn vị tính là *pixel*.

AREA

Kiểu: Số thực.
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Lưu trữ giá trị diện tích được tính bởi lệnh **Area**, **List**, hoặc

Dblist.

ATTDIA

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Làm xuất hiện hoặc không xuất hiện hộp thoại *Enter Attributes* để nhập giá trị cho thuộc tính khi dùng lệnh **Insert** để chèn khối có thuộc tính.
0 Nhập giá trị cho thuộc tính tại dòng nhắc lệnh.
1 Xuất hiện hộp thoại để nhập giá trị cho thuộc tính.

ATTMODE

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc hiển thị các thuộc tính của khối.
0 Không hiển thị tất cả các thuộc tính.
1 Chế độ bình thường: Các thuộc tính có tính chất *thấy được* (*visible*) sẽ được hiển thị; các thuộc tính có tính chất *không thấy được* (*invisible*) sẽ không được hiển thị.
2 Hiển thị tất cả các thuộc tính.

ATTREQ

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Cho phép lệnh **Insert** sử dụng hoặc không sử dụng các giá trị mặc định của các thuộc tính khi chèn khối.
0 Gán giá trị mặc định cho các thuộc tính. Không cho phép nhập giá trị mới.
1 Cho phép nhập giá trị cho các thuộc tính

AUDITCTL

Kiểu: Số nguyên.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Kiểm soát việc tạo file ADT khi thực hiện lệnh **Audit**.
0 Không ghi kết quả ra file ADT.
1 Ghi kết quả ra file ADT.

AUNITS

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Gán đơn vị đo góc.
0 Decimal degrees
1 Degrees/minutes/seconds
2 Gradians
3 Radians
4 Surveyor's units

AUPREC

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.

Giá trị ban đầu: 0
Quy định số chữ số thập phân của đơn vị đo góc.

AUTOSNAP

Kiểu: Số nguyên.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 7
Mở hoặc tắt các chế độ *Marker*, *SnapTip* và *magnet* khi thực hiện chức năng truy bắt đối tượng.
Marker Khung hình ký hiệu điểm truy bắt.
SnapTip Khung hình ký hiệu mô tả tên của vị trí truy bắt.
Magnet Chế độ kéo và giữ ô vuông truy bắt với điểm cần truy bắt.
0 Tắt *Marker*, *SnapTip*, và *Magnet*
1 Mở *Marker*
2 Mở *SnapTip*
4 Mở *Magnet*
Cộng các mã trên để phối hợp các chế độ tắt mở tương ứng (Ví dụ: 7 = 1+2+4 = Mở *Marker* + Mở *SnapTip* + Mở *Magnet*).

B

BACKZ

Kiểu: Số thực.
Chỉ đọc.
Lưu trữ: Trong file bản vẽ.
Trong lệnh **Dview** có lựa chọn CLip dùng để đặt hai mặt cắt vuông góc với tia nhìn, gọi là mặt cắt trước và mặt cắt sau để quy định phạm vi nhìn thấy. Biến BACKZ lưu trữ khoảng cách từ mục tiêu đến mặt cắt sau (*back clipping plane*).
Chỉ có tác dụng khi giá trị của biến VIEWMODE cho phép sử dụng mặt cắt sau.

BLIPMODE

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Xác định chế độ hiện hoặc không hiện các dấu + (*blipmode*) khi nhấn chuột trên màn hình.
0 Tắt chế độ hiện *blipmode*.
1 Bật chế độ hiện *blipmode*.

C

CDATE

Kiểu: Số thực
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Biểu diễn ngày giờ hiện hành.

Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "BYLAYER"
Gán màu cho các đối tượng mới sẽ được tạo ra.

CELTSCALE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Gán hệ số tỉ lệ dạng đường cho đối tượng sắp vẽ.
Dạng đường của đối tượng phụ thuộc vào hai biến CELTSCALE và LTSCALE. Đường thẳng có CELTSCALE = 2 trong bản vẽ có LTSCALE = 0.5 sẽ xuất hiện giống đường thẳng có CELTSCALE = 1 trong bản vẽ có LTSCALE = 1.

CELTYPE
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "BYLAYER"
Gán dạng đường cho đối tượng mới sắp được vẽ.

CHAMFERA
Kiểu: Số thực.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5000
Khoảng cách vát mép trên đường thứ nhất.

CHAMFERB
Kiểu: Số thực.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5000
Khoảng cách vát mép trên đường thứ hai.

CHAMFERC
Kiểu: Số thực.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Khoảng cách vát mép (khi dùng lựa chọn góc của đường vát mép).

CHAMFERD
Kiểu: Số thực.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Góc của đường vát mép.

CHAMMODE
Kiểu: Số nguyên.
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Phương pháp vát mép.
0 Sử dụng khoảng cách vát mép trên đường thứ nhất và đường thứ hai.
1 Sử dụng khoảng cách vát mép và góc của đường vát mép.

CIRCLERAD
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0.0000

CLAYER
Kiểu: Chuỗi.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "0"
Xác định lớp bản vẽ hiện hành.

CMDACTIVE
Kiểu: Số nguyên.
Chỉ đọc.
Không lưu lại khi đóng bản vẽ.
Cho biết các lệnh đang được thực hiện là lệnh thông thường, lệnh trung gian, script hoặc hộp thoại. Giá trị của biến này là tổng các mã tương ứng sau đây:
1 Lệnh thông thường đang được thực hiện.
2 Lệnh thông thường và lệnh trung gian đang được thực hiện.
4 Script đang được thực hiện.
8 Hộp thoại đang hiển thị.
16 Lệnh **AutoLISP** hoặc DDE đang được thực hiện.

CMDDIA
Kiểu: Số nguyên.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định sử dụng hoặc không sử dụng hộp thoại cho lệnh **Plot** và các lệnh ngoài.
0 Không sử dụng hộp thoại.
1 Sử dụng hộp thoại.

CMDECHO
Kiểu: Số nguyên.
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1
Quy định hiển thị hoặc không hiển thị các kết quả trung gian khi thực hiện các hàm **(Command)** của **AutoLISP**.
0 Không hiển thị.
1 Hiển thị.

CMDNAMES
Kiểu: Số nguyên
Chỉ đọc.
Cho biết tên của lệnh đang thực hiện và tên lệnh trung gian (nếu có). Ví dụ: Giá trị LINE'ZOOM chỉ ra lệnh **Zoom** là lệnh trung gian khi đang thực hiện lệnh **Line**.
Biến này được sử dụng cho các phần kết nối chương trình như **AutoLISP**, **DIESEL** và **ActiveX Automation**.

CMLJUST
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định việc định vị trí đường *Mline* bằng đường tâm (*Middle*), đường trên (*Top*) hoặc đường dưới (*Bottom*).
0 Đường trên (*Top*).
1 Đường tâm (*Middle*).

2 Đường dưới (*Bottom*).

CMLSCALE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ chiều rộng đường *Mline*. Hệ số này dương, ví dụ bằng 2, sẽ tăng chiều rộng lên 2 lần. Hệ số này bằng 0 sẽ biến đường *Mline* thành đường thẳng một nét bình thường. Hệ số âm sẽ đổi vị trí các nét của đường *Mline*. Các đường ở phía *Top* sẽ chuyển về phía *Bottom*, và ngược lại.

CMLSTYLE
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "STANDARD"
Kiểu đường Mline sắp vẽ.

COORDS
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc hiển thị tọa độ điểm trên dòng trạng thái.
0 Tọa độ điểm chỉ thay đổi khi dùng chuột chọn một điểm trên màn hình.
1 Hiển thị tọa độ tuyệt đối. Tọa độ thay đổi liên tục khi di chuyển con chuột trên màn hình.
2 Hiển thị tọa độ tuyệt đối, được thay đổi liên tục khi di chuyển con chuột trên màn hình. Trong lúc đang thực hiện lệnh, sẽ hiển thị tọa độ cực tương đối.

CURSORSIZE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 5
Quy định kích thước *sợi tóc* (*crosshairs*) theo tỉ lệ phần trăm so với kích thước màn hình. Giá trị hợp lệ từ 1 đến 100. Khi CURSORSIZE = 100, sợi tóc sẽ to đầy màn hình.

CVPORT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Chứa số định danh của khung nhìn hiện hành. Thay đổi giá trị này tương ứng với việc thay đổi khung nhìn.

D

DATE
Kiểu: Số thực
Chỉ đọc.
Không lưu lại khi đóng bản vẽ.
Chứa ngày giờ hiện hành.

DBMOD
Kiểu: Số nguyên
Chỉ đọc

Không lưu lại khi đóng bản vẽ.
Cho biết trạng thái hiệu chỉnh bản vẽ. Giá trị này bằng tổng các mã tương ứng sau đây:
1 Cơ sở dữ liệu đối tượng đã có sửa đổi.
4 Biến cơ sở dữ liệu đã có sửa đổi.
8 Window đã có sửa đổi.
16 View đã có sửa đổi.

DCTCUST
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: ""
Chứa tên file và đường dẫn của từ điển chính tả tự tạo hiện hành (sử dụng cho lệnh **Spell**).

DCTMAIN
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: Thay đổi tùy theo ngôn ngữ sử dụng lúc cài đặt.
Chứa tên file tự điển chính tả chính. Không cần đường dẫn thư mục vì file này phải chứa trong thư mục *support* của **AutoCAD**. Có thể dùng hàm **Setvar** để thay đổi file tự điển. Các giá trị hợp lệ cho biến DCTMAIN như sau:

Giá trị	Tên ngôn ngữ
enu	American English
ena	Australian English
ens	British English (ise)
enz	British English (ize)
ca	Catalan
cs	Czech
da	Danish
nl	Dutch (primary)
nls	Dutch (secondary)
fi	Finnish
fr	French (unaccented capitals)
fra	French (accented capitals)
de	German (Scharfes s)
ded	German (Dopple s)
it	Italian
no	Norwegian (Bokmal)
non	Norwegian (Nynorsk)
pt	Portuguese (Iberian)
ptb	Portuguese (Brazilian)
ru	Russian (infrequent io)
rui	Russian (frequent io)
es	Spanish (unaccented capitals)
esa	Spanish (accented capitals)
sv	Swedish

DELOBJ

Kiểu: Số nguyên.
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định xóa hay không xóa các đối tượng dùng để tạo các đối tượng khác. Ví dụ dùng lệnh extrude cho một đường tròn để tạo ra khối trụ. Khi đó, tùy theo biến DELOBJ mà đường tròn được giữ lại hoặc xóa đi.

0 Đối tượng được giữ lại
1 Đối tượng bị xóa đi

DEMANDLOAD

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 3
Nếu trong bản vẽ có chứa các đối tượng tự tạo, **AutoCAD** yêu cầu mở các ứng dụng tương ứng với các đối tượng này hay không là do biến DEMANDLOAD quy định

0 Không yêu cầu mở ứng dụng.
1 Yêu cầu mở ứng dụng khi bản vẽ có chứa đối tượng tự tạo được mở.
2 Yêu cầu mở ứng dụng khi ta gọi thi hành các lệnh của ứng dụng này.
3 Yêu cầu mở ứng dụng khi bản vẽ có chứa đối tượng tự tạo được mở hoặc khi ta gọi thi hành các lệnh của ứng dụng này.

DIASTAT

Kiểu: Số nguyên
Chỉ đọc
Không lưu lại khi đóng bản vẽ.
Chứa giá trị trả về tùy theo cách đóng hộp thoại (gần nhất).

0 Cancel
1 OK

DIMALT

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Quy định sử dụng hệ thống đơn vị thay thế, cho phép khi ghi các chữ số kích thước trong dấu [ ]. Xem thêm các biến DIMALTD, DIMALTF, DIMALTZ, DIMALTTZ, DIMALTTD, và DIMAPOST.

OFF Không sử dụng đơn vị thay thế
ON Sử dụng đơn vị thay thế

DIMADEC

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: -1
Quy định số lượng các chữ số thập phân khi ghi kích thước góc.

-1 Sử dụng giá trị gán trong biến DIMDEC (số chữ số thập phân của đơn vị ghi kích thước dài)

0 – 8 Số chữ số thập phân 0 – 8

DIMALTD

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Quy định số chữ số thập phân trong hệ thống đơn vị thay thế.

DIMALTF

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 25.4000
Hệ số tỉ lệ của hệ thống đơn vị thay thế. Khi DIMALT = ON, giá trị kích thước ghi trong dấu [ ] sẽ bằng giá trị kích thước thật nhân với giá trị trong biến DIMALTF.

DIMALTTD

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Số lượng các chữ số thập phân của giá trị dung sai trong hệ thông đơn vị thay thế.

DIMALTTZ

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Cách hiển thị các số 0 trong chữ số dung sai khi ghi kích thước theo đơn vị Anh.

DIMALTU

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Kiểu đơn vị ghi kích thước cho hệ thống đơn vị thay thế (ngoại trừ kích thước góc).

1 Scientific
2 Decimal
3 Engineering
4 Architectural (stacked)
5 Fractional (stacked)
6 Architectural
7 Fractional
8 Windows Desktop (Sử dụng các quy định trong Control Panel)

DIMALTZ

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Cách hiển thị các số 0 trong hệ thống đơn vị thay thế

DIMAPOST

Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ""
Xác định tiền tố và hậu tố cho chữ số kích thước của hệ

thống đơn vị thay thế (ngoại trừ kích thước góc).

DIMASO
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: On
Kích thước liên kết.
OFF Các thành phần kích thước (đường thẳng, cung tròn, mũi tên, chữ số) là các đối tượng riêng biệt.
ON Các thành phần kích thước được nhóm thành một khối.

DIMASZ
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.1800
Kích thước mũi tên kích thước (theo chiều dài). Mũi tên chỉ xuất hiện khi DIMTSZ = 0.

DIMAUNIT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Đơn vị đinh góc cho chữ số kích thước.
0 Decimal degrees
1 Degrees/minutes/seconds
2 Gradians
3 Radians
4 Surveyor's units

DIMBLK
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ""
Tên của một khối tự tạo dùng thay cho mũi tên kích thước.

DIMBLK1
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ""
Chỉ có tác dụng khi DIMSAH = ON. Chứa tên một khối tự tạo dùng thay cho mũi tên kích thước tại đường gióng thứ nhất.

DIMBLK2
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ""
Chỉ có tác dụng khi DIMSAH = ON. Chứa tên một khối tự tạo dùng thay cho mũi tên kích thước tại đường gióng thứ hai.

DIMCEN
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0900
Đinh kích thước dấu tâm và đường tâm khi ghi kích thước bằng các lệnh Dimcenter, Dimdiameter, và Dimradius.
0 Không vẽ dấu tâm
<0 Vẽ dấu tâm và đường tâm
>0 Vẽ dấu tâm

DIMCLRD
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Đinh màu cho đường kích thước, gồm các giá trị từ 0 đến 256, hoặc BYBLOCK, BYLAYER.

DIMCLRE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Đinh màu cho đường gióng kích thước, gồm các giá trị từ 0 đến 256, hoặc BYBLOCK, BYLAYER.

DIMCLRT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Đinh màu cho chữ số kích thước, gồm các giá trị từ 0 đến 256, hoặc BYBLOCK, BYLAYER.

DIMDEC
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 4
Quy đinh số lượng các chữ số thập phân khi ghi kích thước.

DIMDLE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Khoảng vượt của đường kích thước ra khỏi đường gióng. Chỉ có tác dụng khi thay mũi tên bằng dấu gạch chéo bằng biến DIMTSZ.

DIMDLI
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.3800
Khoảng cách giữa các đường kích thước khi có chung đường gióng (khi sử dụng lệnh **Dimbaseline**).

DIMEXE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.1800
Khoảng cách vượt của đường gióng khỏi đường kích thước.

DIMEXO
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0625
Khoảng cách từ đường gióng đến đường kích thước.

DIMFIT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 3
Đinh vị trí của mũi tên và chữ số kích thước so với hai đường

gióng khi khoảng cách giữa các đường gióng tương đối nhỏ.

0 Text and Arrows
1 Text Only
2 Arrows Only
3 Best Fit
4 Leader
5 No Leader

DIMGAP
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0900
Khoảng cách từ chữ số kích thước đến đường kích thước.

DIMJUST
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Định vị trí chữ số kích thước theo phương ngang so với đường kích thước.

0 Ở giữa đường kích thước.
1 Ở gần đường gióng thứ nhất.
2 Ở gần đường gióng thứ hai.
3 Ở trên và nằm song song theo đường gióng thứ nhất.
4 Ở trên và nằm song song theo đường gióng thứ hai.

DIMLFAC
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ khi ghi kích thước dài. Chữ số kích thước hiện lên sẽ bằng kích thước thật trên hình vẽ nhân với hệ số tỉ lệ này.

DIMLIM
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu bằng ON, chữ số kích thước sẽ hiện lên giá trị giới hạn, khi đó biến DIMTOL trở thành OFF.

DIMPOST
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ""
Định tiền tố và hậu tố cho chữ số kích thước

DIMRND
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000

DIMSAH
Kiểu: Logic (ON/ OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Quy định việc dùng các khối làm mũi tên kích thước.

OFF Mũi tên ở hai đầu giống nhau, do biến DIMBLK quy đinh.
On Mũi tên ở hai đầu khác nhau, do hai biến DIMBLK1 và DIMBLK2 quy đinh.

DIMSCALE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ chung cho tất cả giá trị các biến kích thước

DIMSD1
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu DIMSD1 = ON thì không xuất hiện nửa đường kích thước giữa đường gióng thứ nhất và chữ số kích thước.

DIMSD2
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu DIMSD2 = ON thì không xuất hiện nửa đường kích thước giữa đường gióng thứ hai và chữ số kích thước.

DIMSE1
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu DIMSE1 = ON thì không xuất hiện đường gióng thứ nhất.

DIMSE2
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu DIMSE1 = ON thì không xuất hiện đường gióng thứ hai.

DIMSHO
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ON
Nếu DIMSHO = ON thì kích thước khi tạo sẽ hiển thị động trên màn hình.

DIMSOXD
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Nếu DIMOXD = ON thì không cho vẽ đường kích thước bên ngoài đường gióng.

DIMSTYLE
Kiểu: Chuỗi
Chỉ đọc
Lưu trữ trong file bản vẽ.
Tên của kiểu kích thước hiện hành. Dùng lệnh DDIM hoặc DIMSTYLE để thay đổi giá trị này.

DIMTAD
Kiểu: Số nguyên

Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định vị trí chữ số kích thước so với đường kích thước theo phương thẳng đứng.

0 Chữ số kích thước nằm giữa đường kích thước.
1 Chữ số kích thước nằm phía trên đường kích thước. Biến DIMGAP chứa khoảng cách này. Chỉ có tác dụng khi đường kích thước nằm ngang và DIMTIH = OFF.
2 Chữ số kích thước nằm ở phía đường kích thước xa với các điểm định nghĩa.
3 Chữ số kích thước sắp xếp theo tiêu chuẩn Nhật (JIS).

DIMTDEC

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 4
Quy định số lượng các chữ số thập phân của giá trị dung sai.

DIMTFAC

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ giữa chiều cao của chữ số dung sai và chữ số kích thước.

DIMTIH

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: On
Quy định vị trí chữ số kích thước khi ở phía trong hai đường gióng.

OFF Chữ số kích thước song song với đường kích thước.
ON Chữ số kích thước nằm ngang.

DIMTIX

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF

OFF Đối với kích thước thẳng và kích thước góc: nếu khoảng cách giữa hai đường gióng đủ chỗ thì chữ số kích thước vẫn nằm trong hai đường gióng, nếu không sẽ nằm ngoài. Đối với kích thước đường kính và bán kính thì chữ số kích thước nằm ngoài đường tròn hoặc cung tròn.
On Bắt buộc chữ số kích thước phải nằm trong hai đường gióng.

DIMTM

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Sai lệch âm của dung sai.

DIMTOFL

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Quy định vẽ hoặc không vẽ đường kích thước khi chữ số kích thước nằm ngoài.

OFF Không vẽ đường kích thước khi mũi tên nằm ngoài.
ON Vẽ đường kích thước ngay cả khi các mũi tên nằm ngoài.

DIMTOH

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ON
Quy định vị trí chữ số kích thước khi nằm ngoài các đường gióng.

OFF Chữ số song song với đường kích thước.
ON Chữ số nằm ngang.

DIMTOL

Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Khi DIMTOL = ON sẽ ghi kích thước kèm theo dung sai.

DIMTOLJ

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Điểm canh lề của chữ số dung sai

0 Bottom
1 Middle
2 Top

DIMTP

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Sai lệch dương của dung sai.

DIMTSZ

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Kích thước dấu gạch nghiêng thay cho mũi tên tại đầu đường kích thước.

0 Vẽ mũi tên.
>0 Vẽ dấu gạch nghiêng với kích thước bằng DIMTSZ nhân với DIMSCALE.

DIMTVP

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Dịch chuyển chữ số kích thước theo phương vuông góc với đường kích thước. Chỉ có tác dụng khi DIMTAD = OFF.

DIMTXSTY
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "STANDARD"
Kiểu chữ của chữ số kích thước.

DIMTXT
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2.5
Chiều cao của chữ số kích thước.

DIMTZIN
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định cách hiện các số 0 cho chữ số dung sai.

DIMUNIT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Định dạng đơn vị dài cho kích thước.

1	Scientific
2	Decimal
3	Engineering
4	Architectural (stacked)
5	Fractional (stacked)
6	Architectural
7	Fractional
8	Windows Desktop (theo các quy định trong Control Panel)

DIMUPT
Kiểu: Logic (ON/OFF)
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: OFF
Khi DIMUPT = ON, vị trí của chữ số kích thước nằm ở vị trí khi định vị trí của đường kích thước.

DIMZIN
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định cách xuất hiện số 0 khi ghi kích thước theo kiểu Anh.

DISPSILH
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Khi biến này bằng 1 (ON) thì mô hình chỉ hiện lên các đường viền khi đang ở dạng khung dây.

DISTANCE
Kiểu: Số thực
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Lưu trữ khoảng cách được tính bằng lệnh **Dist**.

DONUTID
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0.5000
Giá trị mặc định của đường kính trong khi vẽ donut.

DONUTOD
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1.0000
Giá trị mặc định của đường kính ngoài khi vẽ donut.

DRAGMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Quy định việc hiển thị đối tượng khi đang được kéo rê (drag).

DRAGP1
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 10
Gán regen-drag input sampling rate.

DRAGP2
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 25
Gán fast-drag input sampling rate.

DWGCODEPAGE
Kiểu: Chuỗi
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Có cùng giá trị với biến SYSCODEPAGE (để tương thích với các phiên bản cũ).

DWGNAME
Kiểu: Chuỗi
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Tên bản vẽ do người sử dụng nhập vào. Nếu bản vẽ chưa được đặt tên, DWGNAME sẽ chứa giá trị mặc định là "Drawing.dwg." Nếu người sử dụng nhập cả tên ổ đĩa, tên thư mục, nó sẽ được lưu trong biến DWGPREFIX.

DWGPREFIX
Kiểu: Chuỗi
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Chứa tên ổ đĩa, tên thư mục chứa file bản vẽ.

DWGTITLED
Kiểu: Số nguyên
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Cho biết bản vẽ hiện hành đã được đặt tên chưa.

0	Bản vẽ chưa được đặt tên.
1	Bản vẽ đã được đặt tên.

E

EDGEMODE
Kiểu: Số nguyên

Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định đường biên và đường cắt trong các lệnh **Trim** và **Extend**.
0 Không kéo dài các đường biên và đường cắt.
1 Kéo dài tưởng tượng các đường biên và đường cắt.

ELEVATION
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Cao độ hiện hành trong không gian và trong hệ trục tọa độ UCS hiện hành.

EXPERT
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Quy định việc hiển thị hoặc không hiển thị một số dòng nhắc trong một số trường hợp.
0 Hiển thị tất cả các dòng nhắc bình thường.
1 Không hiển thị các dòng nhắc "*About to regen, proceed?*" và "*Really want to turn the current layer off?*"
2 Không hiển thị các dòng nhắc như trên và các dòng "*Block already defined. Redefine it?*" (Lệnh **Block**) và "*A drawing with this name already exists. Overwrite it?*" (Lệnh **Save** hoặc **Wblock**).
3 Không hiển thị các dòng nhắc như trên và các dòng do lệnh LINETYPE tạo ra khi tải hoặc tạo mới một dạng đường đã có rồi.
4 Không hiển thị các dòng nhắc như trên và các dòng tạo ra khi đặt tên cho một UCS hoặc một khung nhìn bị trùng với tên đã có rồi.
5 Không hiển thị các dòng nhắc như trên và các dòng tạo ra khi đặt tên một kiểu kích thước bị trùng với tên đã có rồi.

EXPLMODE
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1
Quy định việc phá vỡ các khối NUS (*nonuniformly scaled*)
0 Không phá vỡ.
1 Phá vỡ.

EXTMAX
Kiểu: Điểm 3D
Chỉ đọc.
Lưu trữ: Trong file bản vẽ.
Lưu trữ tọa độ góc phải trên của vùng bản vẽ mở rộng (là vùng chứa các đối tượng vẽ ra ngoài giới hạn bản vẽ).

EXTMIN
Kiểu: Điểm 3D
Chỉ đọc.
Lưu trữ: Trong file bản vẽ.
Lưu trữ tọa độ góc trái dưới của vùng bản vẽ mở rộng (là vùng chứa các đối tượng vẽ ra ngoài giới hạn bản vẽ).

F

FACETRES
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5
Mật độ lưới các mặt của khối rắn khi thực hiện các lệnh **Hide**, **Shade** và **Render**. Các giá trị hợp lệ từ 0.01 đến 10.0.

FILEDIA
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc hiển thị hộp thoại file.

FILLETRAD
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5000
Bán kính bo góc hiện hành.

FILLMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc tô các đường Mline, Trace, mặt rắn (Solid), mặt cắt có mẫu là solid và các đa tuyến có chiều rộng.
0 Đối tượng không được tô
1 Đối tượng được tô.

FONTALT
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "simplex.shx"
Tên font dùng để thay thế cho các font khi **AutoCAD** không tìm thấy các font này. Nếu không có tên font thay thế, **AutoCAD** sẽ hiện hộp thoại *Alternate Font dialog box*.

FONTMAP
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "acad.fmp"
Chứa tên *file ánh xạ font* được sử dụng. Mỗi dòng của file này chứa tên hai font: font được thay thế và font dùng để thay thế, ngăn cách nhau bằng dấu chấm phẩy (;). Ví dụ: font *romanc.shx* được thay thế bằng font *times.ttf* như sau:

Romanc.shx;times.ttf

FRONTZ
Kiểu: Số thực

Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Trong lệnh **Dview** có lựa chọn CLip dùng để đặt hai mặt cắt vuông góc với tia nhìn, gọi là *mặt che trước* và *mặt che sau* để quy định phạm vi nhìn thấy. Biến FRONTZ lưu trữ khoảng cách từ mục tiêu đến mặt cắt trước (*front clipping plane*).
Chỉ có tác dụng khi giá trị của biến VIEWMODE cho phép sử dụng mặt cắt trước.

G

GRIDMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định việc hiện lưới trên vùng giới hạn bản vẽ.
0 Không hiện lưới
1 Hiện lưới

GRIDUNIT
Kiểu: Tọa độ điểm 2D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5000,0.5000
Xác định khoảng cách các mắt lưới theo hai phương X và Y.

GRIPBLOCK
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định việc xuất hiện các *Grips* trên các đối tượng tạo thành khối.
0 Chỉ xuất hiện *Grips* tại điểm chèn của khối.
1 Xuất hiện *Grips* trên tất cả các đối tượng tạo thành khối.

GRIPCOLOR
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 5
Quy định màu cho các *Grips* không được chọn (đang ở trạng thái WARM hoặc COLD). Giá trị hợp lệ từ 1-255.

GRIPHOT
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định màu cho các *Grips* được chọn (trạng thái HOT). Các giá trị hợp lệ từ 1 – 255.

GRIPS
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc xuất hiện *Grips* trên đối tượng khi thực hiện các lệnh Stretch, Move, Rotate, Scale, và Mirror.
0 Không xuất hiện *Grips*
1 Xuất hiện *Grips*

GRIPSIZE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 3
Kích thước của *Grips*, tính bằng *pixels*. Giá trị hợp lệ từ 1-255.

H

HANDLES
Kiểu: Số nguyên
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: ON
Cho biết các ứng dụng có thể truy xuất giá trị định danh của đối tượng (*handle*) hay không.
OFF Không thể
On Có thể

HIGHLIGHT
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1
Khi HIGHLIGHT = 1, các đối tượng được chọn trên màn hình sẽ chuyển sang dạng đường nét đứt để phân biệt với các đối tượng khác.

HPANG
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Góc quay của mẫu mặt cắt.

HPBOUND
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định kiểu đối tượng tạo ra bằng lệnh **Bhatch** và **Boundary**.
0 Miền (*Region*)
1 Đa tuyến (*Polyline*)

HPDOUBLE
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Quy định việc tạo nét đôi cho các mẫu mặt cắt tự tạo.
0 Không tạo nét đôi.
1 Tạo nét đôi.

HPNAME
Kiểu: Chuỗi
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: "ANSI31"

Tên mẫu mặt cắt mặc định. Tối đa 34 ký tự, không chấp nhận khoảng trắng.
Nếu không có giá trị mặc định, khi đọc biến này, giá trị trả về là " ".
Khi không muốn có giá trị mặc định, ta gán dấu chấm (.) cho biến này.

HPSCALE
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ của mẫu mặt cắt. Giá trị này phải khác 0.

HPSPACE
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1.0000
Khoảng cách giữa các nét trong mẫu mặt cắt tự tạo. Giá trị này phải khác 0.

I

INDEXCTL
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định các lớp bản vẽ và không gian vẽ có được sắp xếp hay không.
0 Không sắp xếp.
1 Sắp xếp các lớp bản vẽ.
2 Sắp xếp các không gian vẽ.
3 Sắp xếp các lớp bản vẽ và không gian vẽ.

INETLOCATION
Kiểu: Số thực
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "www.autodesk.com/acaduser"
Địa chỉ trên mạng Internet khi sử dụng BROWSER.

INSBASE
Kiểu: Tọa độ điểm 3D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000,0.0000,0.0000
Lưu trữ điểm chèn của bản vẽ khi dùng lệnh **Base**.

INSNAME
Kiểu: Chuỗi
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: " "
Tên khối mặc định khi dùng các lệnh **Ddinsert** hoặc **Insert**.
Nếu không có giá trị mặc định, khi đọc biến này, giá trị trả về là " ".
Khi không muốn có giá trị mặc định, ta gán dấu chấm (.) cho biến này.

ISAVEBAK
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Tăng cường tốc độ lưu bổ sung file, đặc biệt đối với các file bản vẽ lớn. Biến ISAVEBAK quy định việc tạo file lưu trữ (*.bak*).
0 Không tạo file .bak
1 Tạo file .bak

ISAVEPERCENT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 50
Quy định độ lớn của phần không gian không sử dụng trong file bản vẽ. Giá trị hợp lệ từ 0 đến 100. Giá trị mặc định 50 nghĩa là phần không gian không sử dụng trong file bản vẽ không được vượt quá 50% kích thước file. Phần không gian này sẽ mất đi khi dùng lệnh **Save** (*Full Save*). Khi vượt quá 50%, bất kỳ loại lệnh **Save** nào sau đó cũng là *Full Save*.

ISOLINES
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 4
Số các đường biểu diễn mặt cong của các solid khi mô hình đang ở dạng khung dây.

L

LASTANGLE
Kiểu: Số thực
Chỉ đọc. Không lưu lại khi đóng bản vẽ.
Lưu trữ góc tiếp tuyến tại điểm cuối của cung tròn được vẽ cuối cùng trong mặt phẳng XY.

LASTPOINT
Kiểu: Tọa độ điểm 3D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000, 0.0000, 0.0000
Tọa độ điểm chọn cuối cùng. Biểu diễn bằng ký tự @

LASTPROMPT
Kiểu: Chuỗi
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: " "
Lưu trữ dòng nhắc cuối cùng xuất hiện tại dòng nhắc lệnh, kể cả số liệu nhập của người sử dụng.

LENSLENGTH
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 50.0000
Giá trị tiêu cự (tính bằng mm) dùng trong hình chiếu phối cảnh.

LIMCHECK
Kiểu: Số nguyên

Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Cho phép vẽ ngoài giới hạn bản vẽ hay không.
0 Cho phép
1 Không cho phép

LIMMAX
Kiểu: Tọa độ điểm 2D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 12.0000,9.0000
Tọa độ góc phải trên của giới hạn bản vẽ. Biểu diễn trong hệ trục tọa độ WCS.

LIMMIN
Kiểu: Tọa độ điểm 2D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000,0.0000
Tọa độ góc trái dưới của giới hạn bản vẽ. Biểu diễn trong hệ trục tọa độ WCS.

LISPINIT
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Cho phép các hàm **AutoLISP** tự tạo và giá trị các biến được giữ lại khi tạo bản vẽ mới hoặc chúng chỉ có tác dụng trong phạm vi bản vẽ hiện hành.
0 Các hàm **AutoLISP** tự tạo và giá trị các biến được giữ nguyên khi chuyển từ bản vẽ này sang bản vẽ khác.
1 Các hàm **AutoLISP** tự tạo và giá trị các biến chỉ có tác dụng trong bản vẽ hiện hành.

LOCALE
Kiểu: Chuỗi
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: Thay đổi tùy theo lúc cài đặt.
Ngôn ngữ sử dụng của phiên bản **AutoCAD** đang sử dụng.

LOGFILEMODE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định nội dung trong cửa sổ văn bản (*text window*) có được ghi ra file hay không.
0 Không ghi ra file.
1 Ghi ra file.

LOGFILENAME
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "C:\ACADR14\acad.log"
Đường dẫn thư mục của file chứa nội dung cửa sổ văn bản.

LOGINNAME
Kiểu: Chuỗi
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Tên người sử dụng khi cài đặt **AutoCAD**. Tối đa 30 ký tự.

LTSCALE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1.0000
Hệ số tỉ lệ dạng đường (phải khác 0).

LUNITS
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 2
Đơn vị đo chiều dài
1 Scientific
2 Decimal
3 Engineering
4 Architectural
5 Fractional

LUPREC
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 4
Số lượng các chữ số thập phân của đơn vị đo dài.

M

MAXACTVP
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 48
Số lượng tối đa các viewport hoạt động cùng lúc.

MAXOBJMEM
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Quy định trang bộ nhớ của đối tượng.

MAXSORT
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 200
Số lượng tối đa các tên biến hoặc tên khối có thể được sắp xếp. Nếu vượt quá giá trị này sẽ không có phần tử nào được sắp xếp.

MEASUREMENT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định hệ thống đơn vị là hệ Anh hoặc hệ Mét. Ngoài ra, biến này còn quy định file dạng đường và file mẫu mặt cắt

nào được sử dụng.
0 Hệ Anh. Sử dụng các file dạng đường và file mẫu mặt cắt được gán trong các mục ANSIHatch và ANSILinetype của Registry.
1 Hệ Mét. Sử dụng các file dạng đường và file mẫu mặt cắt được gán trong các mục ISOHatch và ISOLinetype của Registry.

MENUCTL
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Kiểm soát việc chuyển đổi trang của menu màn hình.

MENUECHO
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Kiểm soát các dòng nhắc của menu.

MENUNAME
Kiểu: Chuỗi
Chỉ đọc.
Lưu trữ: Trong *Application header*
Tên của Menugroup.

MIRRTEXT
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc thực hiện lệnh đối xứng dòng chữ.
0 Giữ nguyên chiều dòng chữ.
1 Đối xứng cả chiều dòng chữ

MODEMACRO
Kiểu: Chuỗi
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: " "
Xuất hiện chuỗi trên dòng trạng thái như tên bản vẽ hiện hành, ngày giờ và các thông tin khác.

MTEXTED
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "Internal"
Tên của phần mềm soạn thảo dùng để hiệu chỉnh đối tượng Mtext. Mặc định là *multiline text editor*. Nếu ta gán dấu chấm (.) cho biến này, thì phần mềm soạn thảo mặc định sẽ được sử dụng.

O

OFFSETDIST
Kiểu: Số thực
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 1.0000
Giá trị khoảng cách khi tạo đối tượng song song bằng lệnh Offset.
<0 Tạo đối tượng song song đi qua một điểm.
>0 Khoảng cách offset mặc định.

OLEHIDE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định việc hiển thị các đối tượng OLE trong **AutoCAD.**
0 Hiển thị tất cả các đối tượng OLE
1 Chỉ hiển thị các đối tượng OLE trong không gian giấy vẽ.
2 Chỉ hiển thị các đối tượng OLE trong không gian mô hình.
3 Không hiển thị các đối tượng OLE

Biến này có tác dụng khi hiện đối tượng trên màn hình lẫn khi in ra giấy.

ORTHOMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Khi ORTHOMODE = 1, con trỏ chỉ có thể di chuyển theo 2 phương song song với 2 trục XY của UCS hiện hành.
0 Tắt chế độ Ortho.
1 Mở chế độ Ortho.

OSMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định các chế độ truy bắt đối tượng thường trú. Giá trị của biến này bằng tổng các mã sau đây:

0	NONe
1	ENDpoint
2	MIDpoint
4	CENter
8	NODe
16	QUAdrant
32	INTersection
64	INSertion
128	PERpendicular
256	TANgent
512	NEArest
1024	QUIck
2048	APParent Intersection

OSNAPCOORD
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 2

Quy định việc nhập tọa độ tại dòng nhắc lệnh sẽ bỏ qua chế độ truy bắt đối tượng thường trú.

0 Chế độ truy bắt thường trú bỏ qua tọa độ nhập từ bàn phím.

1 Tọa độ nhập từ bàn phím bỏ qua chế độ truy bắt thường trú.

2 Tọa độ nhập từ bàn phím bỏ qua chế độ truy bắt thường trú, ngoại trừ tọa độ điểm nhập từ script.

P

PDMODE

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Cách hiển thị đối tượng điểm.
Các mã 0, 2, 3, và 4 quy định hình dáng của đối tượng điểm.
Mã 1 không hiển thị điểm.
Các mã 32, 64, và 96 được cộng thêm vào các mã trên để vẽ khung bao quanh điểm.

PDSIZE

Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Kích thước vẽ đối tượng điểm.

0 Kích thước điểm bằng 5% chiều cao màn hình đồ họa.

>0 Kích thước tuyệt đối.

<0 Tính theo phần trăm kích thước viewport.

PELLIPSE

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Kiểu elip tạo ra bằng lệnh **Ellipse**.

0 Đối tượng là elip thật sự.

1 Dùng đa tuyến để biểu diễn elip.

PERIMETER

Kiểu: Số thực
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Giá trị chu vi được tính bởi lệnh **Area**, **List**, hoặc **Dblist**.

PFACEVMAX

Kiểu: Số nguyên
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Số đỉnh tối đa của một mặt

PICKADD

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc chọn thêm đối tượng

0 Tắt PICKADD. Để bổ sung thêm đối tượng vào tập hợp đã chọn, ta phải nhấn phím SHIFT trong khi dùng chuột chọn đối tượng.

1 Mở PICKADD. Khi dùng chuột chọn liên tiếp các đối tượng, các đối tượng này được thêm vào tập hợp chọn. Để loại bỏ đối tượng ra khỏi tập hợp chọn, ta nhấn phím SHIFT trong khi chọn đối tượng.

PICKAUTO

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định chế độ chọn đối tượng bằng *automatic window* tại dòng nhắc *Select Objects*.

0 Tắt PICKAUTO

1 Vẽ cửa sổ chọn tự động tại dòng nhắc *Select Objects*.

PICKBOX

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 3
Chiều cao ô vuông chọn đối tượng, đơn vị tính bằng *pixels*.

PICKDRAG

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định phương pháp vẽ cửa sổ chọn.

0 Vẽ cửa sổ chọn bằng 2 điểm. Dùng chuột chọn một đỉnh, sau đó chọn đỉnh thứ hai.

1 Vẽ cửa sổ bằng cách kéo rê. Dùng chuột chọn một đỉnh, sau đó kéo rê (*drag*) đến đỉnh thứ hai thì thả ra.

PICKFIRST

Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc chọn đối tượng trước hoặc sau khi gọi thực hiện lệnh.

0 Tắt PICKFIRST

1 Mở PICKFIRST

PICKSTYLE

Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc chọn theo nhóm và chọn mặt cắt liên kết.

0 Không chọn theo nhóm và mặt cắt liên kết.

1 Chọn theo nhóm.

2 Chọn mặt cắt liên kết.

3 Chọn theo nhóm và mặt cắt liên kết.

PLATFORM

Kiểu: Chuỗi
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Tên hệ điều hành.

Có thể chứa các chuỗi sau:
"Microsoft Windows NT Version 3.51 (x86)"
"Microsoft Windows NT Version 4.00 (x86)"
"Microsoft Windows Version 4.00 (x86)"

PLINEGEN
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định cách vẽ dạng đường (nét đứt) ở gần các đỉnh của đa tuyến.
0 Dạng đường được chỉnh sửa sao cho có nét liền ở cả hai phía của đỉnh.
1 Dạng đường được vẽ tiếp tục khi đi qua đỉnh, không quan tâm đến đỉnh.

PLINETYPE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 2
Quy định việc sử dụng dạng thức mới của đa tuyến.
0 Các đa tuyến ở dạng thức cũ không được chuyển đổi sang dạng thức mới. Các đa tuyến mới tạo ra vẫn ở dạng thức cũ.
1 Các đa tuyến ở dạng thức cũ không được chuyển đổi sang dạng thức mới. Các đa tuyến mới được tạo ra theo dạng thức mới.
2 Các đa tuyến ở dạng thức cũ được chuyển đổi sang dạng thức mới. Các đa tuyến mới được tạo ra theo dạng thức mới.

PLINEWID
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Chiều rộng mặc định của đa tuyến.

PLOTID
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: " "
Chứa tên máy in mặc định.

PLOTROTMODE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định hướng giấy in.

PLOTTER
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Chứa số định danh của máy in mặc định. Máy in đầu tiên có số là 0. Có thể đinh cấu hình cho tối đa 29 máy in.

POLYSIDES
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 4
Số cạnh mặc định khi tạo đa giác bằng lệnh **Polygon**. Giá trị hợp lệ từ 3 đến 1024.

POPUPS
Kiểu: Số nguyên
Chỉ đọc, không lưu lại khi đóng bản vẽ.
Trạng thái của màn hình.
0 Không hỗ trợ *dialog boxes, menu bar, pull-down menus*, và *icon menus*.
1 Hỗ trợ các đặc điểm trên.

PROJECTNAME
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: " "
Tên *project* của bản vẽ hiện hành.

PROJMODE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Lựa chọn *Projection* (chiếu điểm 3D xuống mặt phẳng) khi thực hiện lệnh **Trim** và **Extend**.
0 Sử dụng điểm 3D (không chiếu xuống mặt phẳng).
1 Chiếu xuống mặt phẳng XY của UCS hiện hành.
2 Chiếu xuống mặt phẳng nhìn (*view*) hiện hành.

PROXYGRAPHICS
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định hình ảnh các *proxy* có lưu cùng với bản vẽ hay không.
0 Không lưu cùng với bản vẽ. Khung hình chữ nhật bao sẽ được hiển thị thay thế.
1 Lưu cùng bản vẽ.

PROXYNOTICE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Hiển thị dòng ghi chú cho *proxy*. *Proxy* được tạo ra khi ta mở bản vẽ có chứa các đối tượng do ứng dụng khác tạo ra, mà ứng dụng này không hiện diện.
0 Không hiển thị.
1 Hiển thị.

PROXYSHOW
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc hiển thị *proxy* trong bản vẽ.

0 Không hiển thị các *proxy*.
1 Hiển thị hình ảnh cho tất cả các *proxy*.
2 Chỉ hiện khung chữ nhật bao cho các *proxy*.

PSLTSCALE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Hệ số tỉ lệ dạng đường của không gian giấy vẽ.
0 Sử dụng hệ số tỉ lệ dạng đường LTSCALE.
1 Sử dụng hệ số tỉ lệ dạng đường PSLTSCALE. Chiều dài nét liền của dạng đường dựa theo đơn vị vẽ của không gian giấy vẽ, ngay cả khi đối tượng được tạo ra trong không gian mô hình.

PSPROLOG
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: " "
Tên của *prolog section* trong file acad.psf được lệnh **Psout** đọc.

PSQUALITY
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 75
Quy định chất lượng tô bóng các hình PostScript và quy định các hình này được tô hay chỉ xuất hiện đường viền.
0 Không tạo các hình PostScript.
<0 Độ phân giải các hình PostScript (số lượng các *pixels* trên một đơn vị vẽ của **AutoCAD**).
>0 Số lượng các *pixels* trên một đơn vị vẽ của **AutoCAD,** nhưng sử dụng giá trị tuyệt đối. Chỉ hiện đường viền, không tô các hình PostScript.

Q

QTEXTMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định việc hiển thị các dòng chữ.
0 Hiển thị các ký tự như bình thường.
1 Mở chế độ Quick Text, xuất hiện khung hình chữ nhật thay cho dòng chữ.

R

RASTERPREVIEW
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định các hình ảnh xem trước có lưu cùng với bản vẽ hay không.
0 Không lưu hình ảnh xem trước.
1 Lưu hình ảnh xem trước.

REGENMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Tự động tái tạo bản vẽ.
0 Tắt chức năng REGENAUTO.
1 Mở chức năng REGENAUTO.

RE-INIT
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Khởi tạo lại các thiết bị chuột, màn hình, máy in, các cổng nhập xuất, và file acad.pgp. Giá trị của biến này bằng tổng các mã tương ứng sau:
1 Khởi tạo lại cổng nhập xuất.
4 Khởi tạo các thiết bị chuột,màn hình, máy in.
16 Tải lại file .pgp.

RTDISPLAY
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc hiển thị các hình ảnh *raster* trong quá trình *realtime zoom* hoặc *pan.*
0 Hiển thị nội dung hình ảnh *raster.*
1 Chỉ hiển thị đường viền của hình ảnh *raster.*

S

SAVEFILE
Kiểu: Chuỗi
Chỉ đọc.
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: "auto.sv$"
Tên file *auto-save* của **AutoCAD.**

SAVENAME
Kiểu: Chuỗi
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: " "
Tên file và đường dẫn của bản vẽ hiện hành.

SAVETIME
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 120
Khoảng thời gian tự động lưu bản vẽ, tính bằng phút.
0 Tắt chức năng tự động lưu file.

>0 Tự động lưu file sau một khoảng thời gian.

SCREENBOXES
Kiểu: Số nguyên
Chỉ đọc
Lưu trữ: Trong file Registry của Windows.
Số lượng các mục trong menu màn hình.

SCREENMODE
Kiểu: Số nguyên
Chỉ đọc
Lưu trữ: Trong file Registry của Windows.
0 Hiển thị màn hình văn bản.
1 Hiển thị màn hình đồ họa
2 Cả 2 loại.

SCREENSIZE
Kiểu: Tọa độ điểm 2D
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Lưu trữ kích thước viewport (X và Y), tính bằng *pixels*

SHADEDGE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 3
Quy định cách tô bóng các cạnh của đối tượng.
0 Tô bóng bề mặt, không có các cạnh
1 Tô bóng bề mặt, các cạnh được vẽ bằng màu nền.
2 Các mặt không được tô, các cạnh được vẽ bằng màu của đối tượng.
3 Các mặt tô bằng màu của đối tượng, các cạnh được vẽ bằng màu nền.

SHADEDIF
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 70
Tỉ lệ giữa ánh sáng phản chiếu và ánh sáng môi trường (tính theo phần trăm của ánh sáng phản chiếu).

SHPNAME
Kiểu: Chuỗi
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: " "
Tên *shape* mặc định.
Nếu không có giá trị mặc định, biến này trả về " "
Để biến này không chứa giá trị mặc định, ta gán dấu chấm (.).

SKETCHINC
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.1000
Độ min của hình được vẽ bằng lệnh **Sketch**.

SKPOLY
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định lệnh Sketch tạo ra các đa tuyến hoặc các đường thẳng.
0 Tạo ra các đường thẳng.
1 Tạo ra các đa tuyến.

SNAPANG
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Góc nghiêng của *snap* và *grid* trong viewport hiện hành. Góc nghiêng được tính theo UCS hiện hành.
AutoCAD không tự động cập nhật màn hình khi các biến này bị thay đổi.

SNAPBASE
Kiểu: Tọa độ điểm 2D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000,0.0000
Tọa độ điểm gốc của *snap* và *grid* trong viewport hiện hành, trong hệ trục UCS hiện hành.
AutoCAD không tự động cập nhật màn hình khi các biến nay bị thay đổi.

SNAPISOPAIR
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định các mặt của hình chiếu trục đo vuông góc đều (*isometric*) trong viewport hiện hành.
0 Left
1 Top
2 Right

SNAPMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Mở tắt chế độ *Snap*.
0 Tắt *Snap*
1 Mở *Snap*

SNAPSTYL
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Kiểu *Snap*.
0 Standard
1 Isometric

SNAPUNIT
Kiểu: Tọa độ điểm 2D
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.5000,0.5000
Giá trị bước nhảy *Snap*. Nếu SNAPSTYL = 1, **AutoCAD** tự động gán giá trị X của biến SNAPUNIT để tạo ra *Snap* hình chiếu trục đo đều.

AutoCAD không tự động cập nhật màn hình khi các biến này bị thay đổi.

SORTENTS
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 96
Quy định các chức năng sắp xếp đối tượng khi thực hiện lệnh **Ddselect**. Giá trị biến này bằng tổng các mã tương ứng sau:
0 Không sắp xếp.
1 Sắp xếp dùng cho việc chọn đối tượng.
2 Sắp xếp dùng cho việc truy bắt đối tượng.
4 Sắp xếp dùng cho lệnh **Redraw.**
8 Sắp xếp dùng cho việc tạo slide bằng lệnh **Mslide.**
16 Sắp xếp dùng cho lệnh tái tạo bản vẽ **Regen.**
32 Sắp xếp dùng cho việc in ấn.
64 Sắp xếp dùng cho việc xuất hình PostScript.

SPLFRAME
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định cách hiển thị các đường *spline* và đa tuyến dạng *spline-fit.*
0 Không hiện các điểm điều khiển.
1 Hiện các điểm điều khiển.

SPLINESEGS
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 8
Số lượng các phân đoạn của đa tuyến dạng *spline-fit* tạo bởi lệnh **Pedit**.

SPLINETYPE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Dạng đường cong spline tạo bởi lệnh **Pedit**.
5 B-spline bậc hai.
6 B-spline bậc ba.

SURFTAB1
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Mật độ lưới theo hướng thứ nhất (s) khi tạo mặt lưới đa giác.

SURFTAB2
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Mật độ lưới theo hướng thứ hai (t) khi tạo mặt lưới đa giác.

SURFTYPE
Kiểu: Số nguyên

Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Kiểu mặt cong tạo bởi lựa chọn *Smooth* trong lệnh **Pedit**.
5 Mặt B-spline bậc hai.
6 Mặt B-spline bậc ba.
8 Mặt Bezier.

SURFU
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Mật độ của lưới theo hướng M khi làm min bằng lựa chọn *Smooth* trong lệnh **Pedit**.

SURFV
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 6
Mật độ của lưới theo hướng N khi làm min bằng lựa chọn *Smooth* trong lệnh **Pedit**.

SYSCODEPAGE
Kiểu: Chuỗi
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Xác định các trang mã hệ thống trong file acad.xmx.

T

TABMODE
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Quy định việc sử dụng Tablet.
0 Tắt chế độ sử dụng Tablet.
1 Mở chế độ sử dụng Tablet.

TARGET
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tọa độ điểm Target trong hình chiếu phối cảnh.

TDCREATE
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Ngày giờ bản vẽ được tạo ra.

TDINDWG
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tổng số thời gian hiệu chỉnh bản vẽ.

TDUPDATE
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Ngày giờ cập nhật bản vẽ lần cuối cùng.

TDUSRTIMER
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Lưu trữ *user-elapsed timer.*

TEMPPREFIX
Kiểu: Chuỗi
Chỉ đọc, Không lưu lại khi đóng bản vẽ.
Tên thư mục chứa các file tạm.

TEXTEVAL
Kiểu: Số nguyên
Không lưu lại khi đóng bản vẽ.
Giá trị ban đầu: 0
Quy định việc định giá trị các dòng chữ.
0 Không định giá trị các chuỗi nhập cho dòng nhắc nhập chuỗi hoặc giá trị thuộc tính.
1 Các chuỗi bắt đầu bằng dấu ngoặc [(] hoặc dấu chấm than (!) được xem là biểu thức **AutoLISP** và do đó được định giá trị.

TEXTFILL
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc tô các *TrueType font* khi in, khi xuất ra file bằng lệnh Psout và khi tô bóng.
0 Các chữ chỉ xuất hiện đường viền.
1 Các chữ được tô.

TEXTQLTY
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 50
Độ phân giải của các *TrueType font* khi in, khi xuất ra file bằng lệnh Psout và khi tô bóng. Đơn vị tính là *dots-per-inch.*

TEXTSIZE
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.2000
Chiều cao mặc định khi tạo đối tượng dòng chữ mới.

TEXTSTYLE
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: "STANDARD"
Tên của kiểu chữ hiện hành.

THICKNESS
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Giá trị độ dày (*thickness*) hiện hành.

TILEMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Cho phép truy xuất đến không gian giấy vẽ.
0 Mở không gian giấy vẽ.
1 Không cho truy xuất đến không gian giấy vẽ.

TOOLTIPS
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc hiện *Tooltips.*
0 Không hiện *Tooltips.*
1 Hiện *Tooltips.*

TRACEWID
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0500
Chiều rộng mặc định của đối tượng Trace.

TREEDEPTH
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 3020
Độ sâu tối đa phân nhánh sắp xếp.

TREEMAX
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 10000000
Giới hạn bộ nhớ khi tái tạo bản vẽ.

TRIMMODE
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định **AutoCAD** có cắt xén các cạnh trong lệnh Chamfer và Fillet hay không.
0 Để nguyên các cạnh.
1 Cắt xén các cạnh.

U

UCSFOLLOW
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Tự động tạo hình chiếu bằng (*plan view*) bất kỳ lúc nào chuyển từ hệ trục UCS này sang UCS khác. Mỗi viewport có một giá trị của biến UCSFOLLOW riêng.
0 Tắt chức năng này.
1 Mở chức năng này.
Chức năng này không có tác dụng trong không gian giấy vẽ.

UCSICON
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1

Hiển thị biểu tượng hệ trục tọa độ trong *viewport* hiện hành. Giá trị của biến này bằng tổng các mã sau:

0 Không hiển thị biểu tượng.
1 Hiển thị biểu tượng.
2 Biểu tượng luôn luôn trùng với gốc tọa độ.

UCSNAME
Kiểu: Chuỗi
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tên hệ trục tọa độ hiện hành trong không gian vẽ hiện hành. Nếu UCS hiện hành không được đặt tên, biến này trả về giá trị rỗng.

UCSORG
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tọa độ điểm gốc của hệ trục tọa độ hiện hành trong không gian vẽ hiện hành. Giá trị này luôn luôn là tọa độ tuyệt đối trong hệ trục WCS.

UCSXDIR
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Hướng trục X của UCS hiện hành trong không gian vẽ hiện hành.

UCSYDIR
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Hướng trục X của UCS hiện hành trong không gian vẽ hiện hành.

UNDOCTL
Kiểu: Số nguyên
Chỉ đọc. Không lưu lại khi đóng bản vẽ.
Xác định trạng thái của chức năng UNDO. Giá trị của biến này bằng tổng các mã sau:

0 Tắt chức năng UNDO.
1 Tắt chức năng UNDO.
2 Chỉ UNDO được một lệnh.
4 Mở lựa chọn Auto.
8 Có một nhóm lệnh đang hiện hành.

UNDOMARKS
Kiểu: Số nguyên
Chỉ đọc. Không lưu lại khi đóng bản vẽ.
Số lượng các vị trí đánh dấu đã được tạo ra trong lệnh **Undo** bằng lựa chọn *Mark*. Các lựa chọn *Mark* và *Back* không có tác dụng khi có một nhóm lệnh đang hiện hành.

UNITMODE
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định cách hiển thị của đơn vị đo.

0 Hiển thị số thập phân, feet, inch, surveyor's angles như đã quy định.
1 Hiển thị số thập phân, feet, inch, surveyor's angles giống như giá trị nhập vào.

USERI1–5
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Các biến USERI1, USERI2, USERI3, USERI4, và USERI5 dùng để lưu trữ và cung cấp các dữ liệu kiểu số nguyên.

USERR1–5
Kiểu: Số thực
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0.0000
Các biến USERR1, USERR2, USERR3, USERR4, và USERR5 dùng để lưu trữ và cung cấp các dữ liệu kiểu số thực.

USERS1–5
Kiểu: Chuỗi
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: " "
Các biến USERS1, USERS2, USERS3, USERS4, và USERS5 dùng để lưu trữ và cung cấp các dữ liệu kiểu chuỗi.

V

VIEWCTR
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tọa độ tâm của khung nhìn *View* trong *viewport* hiện hành, biểu diễn trong hệ trục UCS.

VIEWDIR
Kiểu: Vectơ 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Lưu trữ vectơ vuông góc với mặt phẳng khung nhìn *View* trong *viewport* hiện hành, biểu diễn trong hệ trục UCS.

VIEWMODE
Kiểu: Số nguyên
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Quy định các chế độ View trong *viewport* hiện hành. Giá trị của biến này bằng tổng các mã sau:

0 Tắt.
1 Hình chiếu phối cảnh.
2 Mở mặt che trước (*Front clipping*).
4 Mở mặt che sau (*Back clipping*).
8 Đang mở chức năng UCSFollow.
16 Sử dụng cùng với mã 2. Cho phép sử dụng biến FRONTZ để xác định vị trí mặt che trước.

VIEWSIZE
Kiểu: Số thực

Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Chiều cao của khung nhìn *View* trong *viewport* hiện hành, biểu diễn bằng đơn vị bản vẽ.

VIEWTWIST
Kiểu: Số thực
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Lưu trữ giá trị *twist angle* của *viewport* hiện hành.

VISRETAIN
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định việc hiển thị các lớp bản vẽ trong các file tham khảo ngoài *xref*.
0 Ưu tiên sử dụng các giá trị On/Off, Freeze/Thaw, màu sắc, dạng đường của các lớp bản vẽ trong file hiện hành.
1 Ưu tiên sử dụng các giá trị On/Off, Freeze/Thaw, màu sắc, dạng đường của các lớp bản vẽ trong file tham khảo ngoài.

VSMAX
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tọa độ góc phải trên của màn hình ảo của *viewport* hiện hành, biểu diễn trong hệ trục UCS.

VSMIN
Kiểu: Tọa độ điểm 3D
Chỉ đọc
Lưu trữ: Trong file bản vẽ.
Tọa độ góc trái dưới của màn hình ảo của *viewport* hiện hành, biểu diễn trong hệ trục UCS.

W

WORLDUCS
Kiểu: Số nguyên
Chỉ đọc
Không lưu lại khi đóng bản vẽ.
Cho biết hệ trục UCS có trùng với WCS hay không.
0 UCS hiện hành không trùng với WCS.
1 UCS hiện hành trùng với WCS.

WORLDVIEW
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 1
Quy định UCS có bị thay bằng WCS trong các lệnh **Dview** hoặc **Vpoint** hay không.
0 Giữ nguyên UCS hiện hành.
1 UCS hiện hành được thay bằng WCS trong

các lệnh **Dview** hoặc **Vpoint**.

X

XCLIPFRAME
Kiểu: Số nguyên
Lưu trữ: Trong file bản vẽ.
Giá trị ban đầu: 0
Quy định việc hiển thị các đường biên cắt của Xref.
0 Không hiển thị.
1 Hiển thị.

XLOADCTL
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 1
Quy định việc tải các Xref và kiểm soát việc mở bản vẽ gốc hay tạo và mở bản sao chép.
0 Không tải các Xref.
1 Tải các Xref và mở bản vẽ gốc.
2 Tải các Xref và mở bản vẽ sao chép.
Khi XLOADCTL được gán bằng 2, bản sao chép của file tham khảo ngoài được lưu trong thư mục các file tạm hoặc trong thư mục do người sử dụng chỉ ra.

XLOADPATH
Kiểu: Chuỗi
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: " "
Đường dẫn chứa đường dẫn thư mục chứa các file sao chép của các file tham khảo ngoài.

XREFCTL
Kiểu: Số nguyên
Lưu trữ: Trong file Registry của Windows.
Giá trị ban đầu: 0
Quy định **AutoCAD** có chép các tham khảo ngoài ra thành một file .XLG hay không.
0 Không chép ra file.
1 Chép ra file.

Phụ lục III

BẢNG MÃ ASCII

Phụ lục này trình bày bộ mã ASCII tiêu chuẩn. Giá trị bát phân được dùng cho ký tự (character) hoặc chuỗi ký tự hằng số (string constants) sử dụng dạng \nnn. Hệ thống của bạn có thể định nghĩa các mã phụ trên bộ 256 ký tự mở rộng (mã phụ lớn hơn 127). Một vài hệ thống còn được xác định lại bởi các mã ASCII nhỏ hơn như 1-6 và 14-26.

Biểu đồ chuyển đổi mã ASCII

Dec.	Oct.	Hex.	Character	Dec.	Oct.	Hex.	Character
0	000	00	NUL	64	100	40	@
1	001	01	SOH	65	101	41	A
2	002	02	STX	66	102	42	B
3	003	03	ETX	67	103	43	C
4	004	04	EOT	68	104	44	D
5	005	05	ENQ	69	105	45	E
6	006	06	ACK	70	106	46	F
7	007	07	BEL (bell)	71	107	47	G
8	010	08	BS (backspace)	72	110	48	H
9	011	09	HT	73	111	49	I
10	012	0A	LF (line-feed)	74	112	4A	J
11	013	0B	VT	75	113	4B	K
12	014	0C	FF	76	114	4C	L
13	015	0D	CR (return)	77	115	4D	M
14	016	0E	SO	78	116	4E	N
15	017	0F	SI	79	117	4F	O
16	020	10	DLE	80	120	50	P
17	021	11	DC1	81	121	51	Q
18	022	12	DC2	82	122	52	R
19	023	13	DC3	83	123	53	S
20	024	14	DC4	84	124	54	T
21	025	15	NAK	85	125	55	U
22	026	16	SYN	86	126	56	V
23	027	17	ETB	87	127	57	W
24	030	18	CAN	88	130	58	X
25	031	19	EM	89	131	59	Y
26	032	1A	SUB	90	132	5A	Z
27	033	1B	ESC (escape)	91	133	5B	[
28	034	1C	FS	92	134	5C	\
29	035	1D	GS	93	135	5D	]
30	036	1E	RS	94	136	5E	^
31	037	1F	US	95	137	5F	_
32	040	20	(space)	96	140	60	`
33	041	21	!	97	141	61	a
34	042	22	"	98	142	62	b
35	043	23	#	99	143	63	c
36	044	24	$	100	144	64	d
37	045	25	%	101	145	65	e
38	046	26	&	102	146	66	f
39	047	27	'	103	147	67	g
40	050	28	(	104	150	68	h
41	051	29	)	105	151	69	i
42	052	2A	*	106	152	6A	j
43	053	2B	+	107	153	6B	k
44	054	2C	,	108	154	6C	l
45	055	2D	-	109	155	6D	m
46	056	2E	.	110	156	6E	n
47	057	2F	/	111	157	6F	o
48	060	30	0	112	160	70	p
49	061	31	1	113	161	71	q
50	062	32	2	114	162	72	r
51	063	33	3	115	163	73	s
52	064	34	4	116	164	74	t
53	065	35	5	117	165	75	u
54	066	36	6	118	166	76	v
55	067	37	7	119	167	77	w
56	070	38	8	120	170	78	x
57	071	39	9	121	171	79	y
58	072	3A	:	122	172	7A	z
59	073	3B	;	123	173	7B	{
60	074	3C	<	124	174	7C	|
61	075	3D	=	125	175	7D	}
62	076	3E	>	126	176	7E	~
63	077	3F	?	127	177	7F	DEL (delete)

Lập trình thiết kế với AutoLISP và Visual LISP

Tập 1

NGUYỄN HỮU LỘC – NGUYỄN THANH TRUNG

Chịu trách nhiệm xuất bản : Trần Đình Việt
Biên tập : Đức Nhân – Trung Hiếu
Sửa bản in : Dương Ly – Thiên Hương
Trình bày : Hữu Lộc, Thanh Trung
Bìa : Mai Quế Vũ

NHÀ XUẤT BẢN THÀNH PHỐ HỒ CHÍ MINH

62 Nguyễn Thị Minh Khai, Quận 1.
Điện thoại: 8225340 – 8296764 – 8220405 – 8222762 – 8296731- 8223637- 8250616. **FAX**: 84.8.8222726
Email: nxbtphcm@bdvn.vnd.net

In lần thứ hai, Số lượng: 1000 cuốn, khổ 16x24 cm.
Tại: Xí nghiệp In số 5.
Số XB: 399-196/XB-QLXB cấp ngày: 11-04-2003
In xong và nộp lưu chiểu: 08-2003.

Lập trình thiết kế với

AutoLISP và Visual LISP

Sách cùng tác giả đã xuất bản:

1. **Sử dụng AutoCAD thiết kế các mô hình 3 chiều**. *NXB TP.Hồ Chí Minh, 1998*
   In lần thứ 1 (Release 12, 13). In lần thứ 2 (Bổ sung Release 14).
2. **Cơ sở thiết kế máy**. Phần 1 (Chủ biên) *Trường ĐH Bách khoa TP.HCM, 1997*
3. **Sử dụng AutoCAD** (Release 12, phần 2D) *NXB TP.Hồ Chí Minh, 1997*
4. **Sử dụng AutoCAD 13** (Phần 2D) *NXB TP.Hồ Chí Minh, 1997*
5. **Sử dụng AutoCAD 14** (Phần 2D) Tập 1 và 2 *NXB TP.Hồ Chí Minh, 1998*
6. **Sử dụng AutoCAD 2000** *NXB TP.Hồ Chí Minh, 1999*
   Tập 1 - Cơ sở về thiết kế hai chiều (2D)
7. **Sử dụng AutoCAD 2000** *NXB TP.Hồ Chí Minh, 1999*
   Tập 2 - Hoàn thiện bản vẽ thiết kế hai chiều.
8. **Đồ họa máy tính và mô hình hóa hình học** *NXB TP.Hồ Chí Minh, 2000*
   Tác giả: Vera B. Anand. Người dịch: Nguyễn Hữu Lộc
9. **Bài tập Thiết kế mô hình ba chiều với AutoCAD 2000** *NXB TP.Hồ Chí Minh, 2000*
10. **Sử dụng AutoCAD 2002** *NXB TP.Hồ Chí Minh, 2001*
    Tập 1 - Cơ sở về thiết kế hai chiều.
11. **Thiết kế mô hình ba chiều với AutoCAD** *NXB TP.Hồ Chí Minh, 2002*
12. **Độ tin cậy trong thiết kế kỹ thuật**. *NXB Đại học Quốc gia TP.Hồ Chí Minh, 2002*
13. **Cơ sở thiết kế máy**. Phần 2 *NXB Đại học Quốc gia TP.Hồ Chí Minh, 2002*
14. **Bài tập chi tiết máy** *NXB Đại học Quốc gia TP.Hồ Chí Minh, 2002*
15. **Tạo các tiện ích thiết kế trên AutoCAD** *NXB TP.Hồ Chí Minh, 2003*
16. **Thiết kế cơ khí với AutoCAD Mechanical** *NXB TP.Hồ Chí Minh, 2003*
17. **Sử dụng AutoCAD 2002** *NXB TP.Hồ Chí Minh, 2003*
    Tập 2 - Hoàn thiện bản vẽ thiết kế hai chiều.
18. **Lập trình thiết kế với AutoLISP và Visual LISP** *NXB TP.Hồ Chí Minh, 2003*
    Tập 1 và 2 (Biên soạn với Nguyễn Thanh Trung)

Giá : 33.000 đ