# NGÔN NGỮ AUTOLISP

## I> TỔNG QUAN VỀ NGÔN NGỮ AutoLISP

### 1. Giới thiệu chung:

LISP là chữ viết tắt của cụm từ tiếng Anh: LISt Processor (Xử lý danh sách)

AutoLisp là một ứng dụng của ngôn ngữ Lisp được sử dụng trong môi trường AutoCad. LISP là ngôn ngữ lập trình thuộc nhóm trí tuệ nhân tạo do MacCarthy soạn thảo cuối những năm 50. Với AutoLisp người dùng có thể mở rộng và tuỳ biến các chức năng của AutoCad.

Hiện nay AutoLisp đã được hãng Autodesk phát triển theo các số hiệu phát hành của AutoCad. Về căn bản những phiên bản sau vẫn sử dụng được những chương trình lập bằng phiên bản trước, ngược lại thì không được do có một số biến hệ thống và lệnh của AutoCad giữa các phiên bản không giống nhau nên việc dùng chung có gặp một số trở ngại. Do vậy yêu cầu người lập trình AutoLisp phải nắm thật vững AutoCad để sử dụng AutoLisp một cách hiệu quả.

AutoLisp là một ngôn ngữ lập trình thông dịch, nghĩa là dịch đến dòng nào thực hiện dòng đó và cho kết quả, không có trình biên dịch riêng. Một tập hợp các câu lệnh của AutoLisp được gọi là hàm Lisp và tệp (file) chứa các hàm gọi là tệp (file) Lisp có phần mở rộng là *.Lsp.

Với AutoLisp, người dùng có thể dễ dàng truy cập đến dữ liệu của AutoCad, có thể thay đổi, tạo mới, xoá bỏ các đối tượng, thêm các thông tin vào bản vẽ thực hiện các công việc Tự động hoá trong thiết kế...

### 2. Các qui ước của AutoLisp:

#### *a) Cách viết chương trình của AutoLisp*

Có 2 cách viết chương trình AutoLisp:

- *Viết trực tiếp:*

Tại dòng nhắc Command: của AutoCad ta có thể gõ các câu lệnh theo cú pháp của AutoLisp. Lệnh này sẽ được thực thi ngay và cho kết quả trên màm hình tại vùng dòng lệnh, nhưng lệnh này không lưu trữ được.

- *Viết thành chương trình:*

Dùng chương trình soạn thảo (dạng mã ASCII) bất kỳ hoặc Visual LISP, viết thành chương trình như một tạp tin nguồn có phần mở rộng *.lsp

Tên tệp tuân thủ theo qui ước của hệ điều hành, thường không quá 8 ký tự, giữa các ký tự không có khoảng trống.

***b) Tải và chạy chương trình ứng dụng AutoLisp***

**Từ VLISP:** Tools\ Load Text in Editor

Từ AutoCad: Tool\ Load Application hoặc trên dòng lệnh Command: **ap**

Để AutoCad tự động tải ngay từ khi khởi động hoặc mở bản vẽ có 2 cách:

- Đặt tên tệp là ACAD.LSP và đặt trong thư mục **Support** của AutoCad
- Khi tải file lần đầu sử dụng **Startup Suite\ Contents** và chọn đường dẫn cho file

***c) Các hàm trong AutoLisp***

AutoCad nhận và xử lý các lệnh trong hàm của AutoLisp theo cú pháp sau:

- Tên hàm do người dùng định nghĩa gồm các chữ cái và con số trừ các ký tự đặc biệt: như: ? < > , . * & ^ % $ # @ ! ~ \ | { } [ ] ..., tên hàm không nên quá dài và phải dễ quản lý.
- Hàm và câu lệnh của AutoLisp phải được đặt trong cặp dấu ngoặc đơn, bắt đầu bằng "(" và kết thúc bằng ")"
- Hàm được viết từ trái qua phải theo kiểu Ba-lan, nghĩa là phần tử đầu tiên sau dấu mở ngoặc phải là tên hàm (có sẵn hay do người lập trình tự định nghĩa) hay toán tử. Các phần tử đứng sau là các tham số cần thiết để thực hiện hàm hay toán tử đó.
- Phân cách giữa tên hàm (hay toán tử) với các tham số, giữa các tham số với nhau phải có ít nhất một dấu cách ( dấu Space).
- Một câu lệnh có thể viết trên nhiều dòng. Các dòng chữ có thể viết thụt vào tuỳ ý theo cấu trúc đoạn lệnh cho dễ hiểu.
- Không phân biệt chữ hoa và chữ thường, thường thì tên hàm nên viết bằng chữ thường, tên các lệnh và các biến hệ thống của AutoCad viết bằng chữ hoa cho dễ đọc và chương trình sáng sủa hơn.
- Bất kỳ một hàm nào cũng trả về một giá trị nào đó, nếu không có giá trị trả về trị số mặc định là nil.
- Lời chú thích ghi trong chương trình AutoLisp được ghi sau dấu " ; " và không được thực thi trong chương trình.

***d) Các biến trong AutoLisp***

- Các biến của chương trình AutoLisp hoạt động tương tự như các biến của chương trình khác.
- Tên biến gồm các chữ cái và các con số (trừ các ký tự đặc biệt: như: ? < > , . * & ^ % $ # @ ! ~ \ | { } [ ] ...), nếu chữ số đứng đầu thì tiếp sau phải là chữ cái để tránh nhầm với các hằng số. Tên biến không nên quá dài
- Tên biến không phân biệt chữ hoa và chữ thường.

- Có 2 loại biến:

  *+ Biến chung: là biến tồn tại trong suốt quá trình làm việc của AutoCad. Để kiểm tra giá trị cuả biến trong dòm Command của AutoCad gõ "!ten_biến".*

  *+ Biến riêng: Là biến chỉ tồn tại bên trong một hàm. Kết thúc hàm biến này nhận giá trị "Nil"*

Chú ý: Các biến tham gia vào các biểu thức phải được gán giá trị hoặc định nghĩa nếu không ứng dụng sẽ bị lỗi.

## II> CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG AutoLISP

### 1. Kiểu danh sách: (list)

Đây là kiểu đặc trưng của ngôn ngữ Lisp bao gồm nhóm các giá trị riêng lẻ gồm các biến, các hằng số, các hàm... cách nhau bằng khoảng trống nằm trong dấu ngoặc đơn.

Danh sách được chia làm 3 loại chính:

- Biểu thức (expression list): Chứa tên hàm và các tham số của hàm
- Toạ độ điểm (Point Coordinate List): Có hàm Quote hoặc dấu ‘ ở phía trước. Đây là trường hợp đặc biệt của danh sách kho dữ liệu, trong đó thông tin lưu trữ là toạ độ điểm.
- Kho dữ liệu (Data Storage List): Có hàm Quote hoặc dấu ‘ ở phía trước có thể chứa bất kỳ kiểu dữ liệu nào.

Ký hiệu hoàn trả của AutoLisp: “LIST”

Ví dụ: toạ độ của 1 điểm được khai báo dưới dạng danh sách: `'(1.0 1.0 1.0)` hoặc `(list 1.0 1.0 1.0)`

### 2. Kiểu số:

*2.1. Kiểu số nguyên*

- Kiểu số nguyên là số nguyên thông thường trong toán học. Trong AutoLisp giới hạn của số nguyên từ: -32768 đến +32767
- Các phép tính trên số nguyên cho kết quả là một số nguyên, nếu là phép chia cho kết quả là một số nguyên, phần dư nếu có sẽ bị cắt bỏ.

  Ví dụ: (/ 5 2) cho kết quả là 2
- Ký hiệu hoàn trả của AutoLisp: “INT”

*2.2. Kiểu số thực*

- Kiểu số thực trong AutoLisp có độ chính xác đến 14 chữ số sau dấu phẩy thập phân
- Các số thực có thể biểu diễn theo dạng chú thích khoa học, qua đó một số e hoặc E được theo sau bởi số mũ của con số đó.
- Ký hiệu hoàn trả của AutoLisp: “REAL”

### 3. Kiểu chuỗi:

- Chuỗi kí tự là tập hợp các ký tự bất kỳ đặt trong dấu ngoặc kép “ ”. Trong AutoLisp chuỗi dài không quá 132 ký tự
- Ký hiệu hoàn trả của AutoLisp: “STR”

## 4. Kiểu đối tượng và nhóm đối tượng

*4.1. Kiểu đối tượng*

- Mỗi đối tượng được vẽ trong AutoCad đều được quản lý theo tên (ENAME – Entity Name). Mỗi tên đối tượng sẽ lưu trữ toàn bộ thông tin về đối tượng đó.
- Ký hiệu hoàn trả của AutoLisp: "ENAME"

*4.2. Nhóm đối tượng*

- Nhóm đối tượng (AutoCad selection set) là kiểu đặc trưng cho tập hợp chứa các ENAME của các đối tượng được lựa chọn.
- Ta có thể lấy ENAME của một đối tượng trong tập hợp các đối tượng được chọn
- Ký hiệu hoàn trả của AutoLisp: "PICKSET"

## 5. Số Pi và Nil

- Số Pi trong toán học trong AutoLisp được ký hiệu là pi và nhận giá trị không đổi là 3.1415926. Pi tham gia vào các biểu thức toán học và là số đo góc bằng Radian
- Nil là ký hiệu để chỉ ra rằng biến hay hàm không có giá trị hoặc biểu thức lôgíc nhận giá trị không đúng.

## III> CÁC HÀM CHUẨN CỦA AutoLISP

### 1. Phép gán

**a. Hàm** `(setq...)`:

CHỨC NĂNG: Gán giá trị cho 1 biến

CÚ PHÁP:

```
(setq b1 gt1 [b2 gt2] ...)
```

GIẢI THÍCH:

Gán gt1 cho biến b1, gt2 cho biến b2 ... Mỗi biến nhận một giá trị viết sau nó

Giá trị có thể là dữ liệu, một biến khác hoặc một biểu thức đã xác định trước đó

Để xoá một biến ra khỏi bộ nhớ: `(setq biến Nil)`

Để AutoCad thực hiện lệnh của AutoLisp mà không hiển thị các dòng lệnh hiện trên màn hình ta gán giá trị 0 cho biến hệ thống CMDECHO bằng lệnh: `(setq cmdecho 0)`

**b. Hàm** `(setvar...)`:

CHỨC NĂNG: Gán giá trị cho 1 biến hệ thống.

CÚ PHÁP:

```
(setvar varname value)
```

GIẢI THÍCH:

`varname`: Tên biến hệ thống

`value`: Giá trị cần gán

VD: `(setvar "FILLETRAD" 10.00) ---> 10.00`

**c. Hàm** `(set...)`:

CHỨC NĂNG: Gán tên biến cho 1 biến

CÚ PHÁP: `(set 'biến1 'biến2)`

GIẢI THÍCH:

Gán tên `biến2` cho `biến1`

Mỗi khi lấy giá trị của `biến1` thì thực chất là lấy giá trị của `biến2`

VD:
```
(setq a 10.00)
(set 'b 'a)
---> b = 10.00
```

## 2. Các hàm chuyển lệnh từ AutoLisp sang AutoCad

**a. Hàm** `(Load ...)`

CHỨC NĂNG: Gọi một chương trình ứng dụng vào AutoCad

CÚ PHÁP:

```
(Load "Tên_tệp")
```

GIẢI THÍCH:

`Tên_tệp`: là tên tập tin có phần mở rộng *.LSP, .ARX, .ADS hoặc .EXE

Nếu chương trình đặt ngoài thư mục làm việc của AutoCad thì phải chỉ rõ đường dẫn của tệp tin đó

**b. Hàm** `(Command ...)`

CHỨC NĂNG: Thực hiện lệnh của AutoCad

CÚ PHÁP:

```
(Command "Tên_lệnh" [các đáp ứng lời nhắc] [các tuỳ chọn]...)
```

GIẢI THÍCH:

`Tên_lệnh`: là tên các lệnh của AutoCad

`[các đáp ứng lời nhắc]` và `[các tuỳ chọn]`: Tuân theo các lệnh của AutoCad

Nếu `Tên_lệnh = _Tên_lệnh hoặc _.Tên_lệnh` Autocad sẽ truy cập các gía trị trong bảng số liệu của lệnh thực thi trong AutoCad

VD: `(Command "CIRCLE" '(100.00 100.00) 10.00)`

`---> Vẽ đường tròn tâm có toạ độ (100.00, 100.00) có bán kính 10.00`

## 3. Các hàm nhập liệu từ người dùng

***Các hàm sau sẽ tạm dừng chương trình để yêu cầu người dùng nhập dữ liệu vào từ bàn phím hoặc chuột***

**a. Hàm** `(getpoint ...)`

CHỨC NĂNG: Chờ người dùng nhập toạ độ một diểm

CÚ PHÁP:

```
(getpoint [point] [prompt])
```

GIẢI THÍCH:

`point`: Nếu có, cho bằng 1 danh sách điểm , là điểm thứ nhất, còn điểm người dùng nhập vào sẽ là điểm thứ 2. Điểm thứ 2 có thể cho bằng toạ độ tương đối.

`[prompt]`: Nếu có, là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập. Dòng nhắc phải được đặt trong ngoặc kép " "

VD:

```
(setq pt1 (getpoint "Cho tam duong tron:"))
```

Kết quả cho trên dòng nhắc:

```
Cho tam duong tron:
```

**b. Hàm** `(getdist ...)`

CHỨC NĂNG: Chờ người dùng nhập vào:

- Một số thực là một khoảng cách
- Toạ độ của 1 hoặc 2 điểm

Nếu nhập toạ độ điểm, AutoLisp hoàn trả khoảng cách giữa hai điểm

CÚ PHÁP:

```
(getdist [point] [prompt])
```

GIẢI THÍCH:

`point`: giống như `getpoint`

`[prompt]`: là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập

VD:

```
(setq r1 (getdist "Cho ban kinh duong tron:"))
```

Kết quả cho trên dòng nhắc:

`Cho tam duong tron:` - Nhập vào một số thực dương hoặc

- Nhập toạ độ một điểm, dòng nhắc xuất hiện:

`Second point`: Tiếp tục nhập vào điểm thứ 2 để lấy khoảng cách giữ 2 điểm

**c. Hàm** `(getangle ...)`

CHỨC NĂNG: Chờ người dùng nhập vào:

- Một số thực là số đo bằng độ của góc hoặc cung tròn
- Toạ độ của 1 hoặc 2 điểm

Nếu nhập toạ độ điểm, AutoLisp hoàn trả góc nghiêng giữa đoạn thẳng nối hai điểm so với phương nằm ngang

Kết quả trả về: REAL ( số đo là Radian)

CÚ PHÁP:

(**getangle** [point] [prompt])

GIẢI THÍCH:

point: giống như **getpoint**

[prompt]: là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập

VD:

(**setq** a1 (**getangle** "Cho goc nghieng cua duong thang:"))

Kết quả cho trên dòng nhắc chờ người dùng nhập số liệu:

Cho goc nghieng cua duong thang:

**d. Hàm** (**getint ...**)

CHỨC NĂNG: Chờ người dùng nhập vào một số nguyên

Kết quả: INT

CÚ PHÁP:

(**getint** [prompt])

GIẢI THÍCH:

[prompt]: là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập

**e. Hàm** (**getreal ...**)

CHỨC NĂNG: Chờ người dùng nhập vào một số thực

Kết quả: REAL

CÚ PHÁP:

(**getreal** [prompt])

GIẢI THÍCH:

[prompt]: là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập

**f. Hàm** (**iniget...**)

CHỨC NĂNG: Kiểm soát các kiểu giá trị nhập vào từ người dùng cho các hàm nhập số liệu họ getxxx. Hàm này, nếu dùng phải đặt trước các hàm nhập số liệu họ getxxx cần khống chế

CÚ PHÁP:

(**iniget** mã_số ["Chuỗi_định_dạng"])

GIẢI THÍCH:

`mã_số`: Giá trị mã số kiểm soát cách nhập số liệu vào. Giá trị `mã_số` và các hàm chịu tác động cho trong bảng sau:

| Mã số | Chức năng kiểm soát | Hàm chịu tác động |
|---|---|---|
| 0 | Bình thường trả về Nil nếu gõ ENTER khi chưa nhập số liệu | Toàn bộ các hàm getxxx trừ hàm getstr |
| 1 | Không cho phép người dùng gõ ENTER khi chưa nhập số liệu | getint, getreal, getdist, getangle, getpoint, getkword... |
| 2 | Không cho phép nhập số 0 | getint, getreal, getdist, getangle, getpoint, |
| 4 | Không cho phép nhập số âm | getint, getreal, getdist, |
| 8 | Cho phép nhập toạ độ điểm nằm ngoài LIMITS của bản vẽ | getpoint, getcorner |

Nếu mã số có giá trị tổng của mốtố giá trị cho trong bảng thì chức năng kiểm soát sẽ là tổng các chức năng.

`Chuỗi_định_dạng`: Là một chuỗi ký tự dùng làm từ khoá hỗ trợ cho các hàm getxxx có các tuỳ chọn khi nhập số liệu

+ Mỗi từ khoá cách nhau một dấu trống

+ Các chữ cái viết hoa viết liền nhau trong từ khoá sẽ là dấu hiệu của từ khoá. Khi nhập số liệu, người dùng gõ theo các chữ cái viết hoa để lựa chọn

+ Nếu tất cả các chữ cái trong từ khoá đều viết hoa thì các từ khoá cách nhau bằng dấu phẩy (,)

VD1:

```
(iniget 1)
(setq a1 (getdist "Cho ban kinh:"))
```

Kết quả cho trên dòng nhắc chờ người dùng nhập số liệu:

```
Cho ban kinh:
```

Nếu người dùng gõ ENTER mà không nhập gì cả sẽ có dòng thông báo:

```
Requires numeric distance or two poionts
Cho ban kinh:
```

Đợi người dùng nhập liệu chương trình mới tiếp tục

VD2:

```
(iniget 1 "DAi ROng")
(setq d1 (getdist "DAi/ROng/<Cho duong cheo>:"))
```

Kết quả cho trên dòng nhắc chờ người dùng nhập số liệu:

```
DAi/ROng/<Cho duong cheo>:
```

Người dùng có thể nhập số thực hoặc các chữ cái DA hoặc RO

**g. Hàm** `(getkword ...)`

CHỨC NĂNG: Chờ người dùng nhập vào một ký tự hoặc một chuỗi ký tự liền nhau không có dấu trống

Kết quả trả về: STR

CÚ PHÁP:

```
(getkword [prompt])
```

GIẢI THÍCH:

Hàm này luôn đặt sau hàm `initget`, nếu chuỗi nhập vào không trùng với từ khoá trong hàm `initget` trước đó, `getkword` yêu cầu nhập lại

Hàm này thường áp dụng khi người dùng nhập vào các lựa chọn

`[prompt]:` là dòng nhắc nên chỉ rõ các từ khoá để người dùng dễ nhập dữ liệu.

VD:

```
(initget "VUong DAgiac:")
(getkword "Nhap cac lua chon: VUong/DAgiac:")
```

Kết quả cho trên dòng nhắc:

```
Nhap cac lua chon: VUong/DAgiac:
```

Ta chỉ cần nhập các chữ cái:

VU kết quả trả về: "`vuong`"

DA kết quả trả về: "`dagiac`"

**h. Hàm** `(getstring ...)`

CHỨC NĂNG: Chờ người dùng nhập vào một chuỗi ký tự . Nếu chuỗi ký tự dài hơn 132 ký tự, hàm chỉ lấy 132 ký tự

Kết quả: STR

CÚ PHÁP:

```
(getstring [string] [prompt])
```

GIẢI THÍCH:

`[string]`: Nếu không có hoặc nhận giá trị Nil, không được nhập dấu trống, nếu gõ dấu trống tương đương gõ ENTER kết thúc nhập.

Nếu có giá trị khác Nil cho phép nhập cả dấu trống. Kết thúc nhập phải gõ ENTER

`[prompt]`: là dòng nhắc hoặc giải thích về dữ liệu sẽ nhập

**i. Hàm** `(getvar ...)`

CHỨC NĂNG: Lấy giá trị hiện hành của biến hệ thống trong AutoCad

CÚ PHÁP:

`(getvar varname)`

GIẢI THÍCH:

`varname`: Tên biến hệ thống

VD:

`(getvar "DIMLFAC" 10)`

---> cho giá trị biến DIMLFAC = 10

**4. Các hàm toán học**

***4.1. Các phép tính***

CHỨC NĂNG: Thực hiện các phép tính số học thông thường

CÚ PHÁP:

`(phép_toán tham_số1 [tham_số2]...)`

GIẢI THÍCH:

`phép_toán`: là các phép tính toán số học thông thường

| | |
|---|---|
| Phép cộng: | + |
| Phép trừ: | - |
| Phép nhân: | * |
| Phép chia: | / |
| Phép tăng thêm 1: | 1+ |
| Phép giảm đi 1: | 1- |

`tham_số1`: Trong phép trừ và chia thì `tham_số1` là số bị trừ hoặc bị chia

`tham_số2...`: Trong phép trừ và chia thì `tham_số2...` là số trừ hoặc số chia

Trong phép tăng thêm 1 hoặc giảm đi 1 thì chỉ có 1 tham số

CHÚ Ý:

Tham số có thể là hằng hoặc biểu thức

Kết quả tính phụ thuộc chủ yếu vào kiểu của tham số tham gia phép tính.

### *4.2. Các phép so sánh*

CHỨC NĂNG: Thực hiện các phép so sánh trong các biểu thức logíc

CÚ PHÁP:

```
(phép_so_sánh tham_số1 [tham_số2]...)
```

GIẢI THÍCH:

`phép_so_sánh`: Bao gồm:

| | |
|---|---|
| Bằng: | = |
| Không bằng (khác): | /= |
| Lớn hơn: | > |
| Lớn hơn hoặc bằng: | >= |
| Nhỏ hơn: | < |
| Nhỏ hơn hoặc bằng: | <= |
| Và: | `and` |
| Hoặc: | `or` |
| Phủ định: | `not` |

`tham_số1, tham_số2`: là các đối tượng để so sánh

CHÚ Ý:

Tham số có thể là hằng số, biến số hoặc biểu thức

Kết quả tính phụ thuộc chủ yếu vào kiểu của tham số tham gia phép tính.

Riêng đối với thực, máy tính coi số 0 chỉ là xấp xỉ 0 nên khi dùng dấu = so sánh với số 0 có thể trả giá trị Nil. Trong trường hợp này nên dùng `(equal...)` để so sánh

### *4.3. Các hàm so sánh*

**a. Hàm** (eq ...)

CHỨC NĂNG: Thực hiện so sánh xem hai tham số có thực sự bằng nhau hay không

Kết quả: cho T (đúng) hoặc Nil (sai)

CÚ PHÁP:

```
(eq tham_số1 tham_số2)
```

GIẢI THÍCH:

Kết quả đúng khi tham_số1 giống hệt tham_số2 (cùng là kiểu tham số, cùng lá số...)

**b. Hàm** (equal ...)

CHỨC NĂNG: Thực hiện so sánh xem hai tham số có thực sự bằng nhau hay không theo tiêu chuẩn là độ chính xác kèm theo

Kết quả: cho T (đúng) hoặc Nil (sai)

CÚ PHÁP:

```
(equal tham_số1 tham_số2 [độ_chính_xác])
```

GIẢI THÍCH:

Kết quả đúng khi tham_số1 bằng tham_số2 với sai số là độ_chính_xác (nếu có)

**c. Hàm** (max ...)

CHỨC NĂNG: Tìm giá trị lớn nhất trong một dãy các tham số

Kết quả: trả về giá trị lớn nhất

CÚ PHÁP:

```
(max tham_số1 tham_số2 [tham_số3]...)
```

**d. Hàm** (min ...)

CHỨC NĂNG: Tìm giá trị nhỏ nhất trong một dãy các tham số

Kết quả: trả về giá trị nhỏ nhất

CÚ PHÁP:

```
(min tham_số1 tham_số2 [tham_số3]...)
```

**e. Hàm** (gcd ...)

CHỨC NĂNG: Tìm ước số chung lớn nhất trong một dãy các tham số kiểu số nguyên

CÚ PHÁP:

(**gcd** tham_số1 tham_số2 [tham_số3]...)

CHÚ Ý:

tham_số1 tham_số2 [tham_số3]...: Phải là kiểu số nguyên

**f. Hàm** (rem ...)

CHỨC NĂNG: Tìm phần dư trong phép chia 2 tham số.

Kết quả: trả về số dư

CÚ PHÁP: (**rem** tham_số1 tham_số2)

GIẢI THÍCH:

tham_số1: Số bị chia

tham_số2: Số chia

### *4.4. Các hàm đại số*

CÚ PHÁP:

(**sqrt** number): Lấy căn bậc 2 của số thực dương number

(**expt** cơ_số số_mũ): Lấy luỹ thừa bậc số_mũ của số thực cơ_số

(**exp** số_mũ): Lấy luỹ thừa bậc số_mũ của cơ số e

(**abs** number): Lấy giá trị tuyệt đối của một số number

(**log** number): Lấy logarit cơ số e của một số number

### *4.5. Các hàm lượng giác*

CÚ PHÁP:

(**sin** angle)

(**cos** angle)

(**atan** mumber1 [number2])

CHÚ Ý:

angle: Là số đo góc lấy theo radian

mumber1 [number2]: Là số thực

Kết quả của hàm atan là góc có số đo là radian

### *4.6. Các hàm kiểm soát dạng số*

**e. Hàm** `(fix ...)`

CHỨC NĂNG: Trả về phần nguyên của một số. Phần số nguyên này không được làm tròn

CÚ PHÁP:

```
(fix number)
```

CHÚ Ý:

Hàm này được sử dụng như một bộ lọc dữ liệu. Dữ liệu là số nguyên hay số thực sẽ được lọc thành số nguyên trước khi được gán cho tham số

Ngoài ra ta có thể sử dụng hàm này để lấy phần thập phân (sử dụng kết hợp với phép trừ)

**b. Hàm** `(fload ...)`

CHỨC NĂNG: Chuyển một số có kiểu nguyên hoặc thực sang kiểu số thực

CÚ PHÁP:

```
(fload number)
```

### 5. Các hàm tạo cấu trúc điều khiển

**a. Hàm** `(if ...)`

CHỨC NĂNG: Ra điều liện thực hiện một số lệnh

CÚ PHÁP:

```
(if testexpr thenexpr [elseexpr])
```

GIẢI THÍCH:

`testexpr`: Là biểu thức kiểm tra điều kiện

`thenexpr`: Biểu thức lệnh được thực hiện nếu biểu thức `testexpr` nhận giá trị T

`elseexpr`: Biểu thức lệnh được thực hiện nếu biểu thức `testexpr` nhận giá Nil. Nếu không có biểu thức này, hàm không thực hiện và trả về giá trị nil

**b. Hàm** `(while ...)`

CHỨC NĂNG: Thực hiện các biểu thức, lệnh trong hàm một số lần lặp có điều kiện

CÚ PHÁP:

```
(while testexpr expr)
```

GIẢI THÍCH:

`testexpr`: Là biểu thức kiểm tra, khi nào `testexpr` nhận giá trị nil, vòng lặp chương trình ngừng thực hiện

`expr`: Biểu thức, lệnh cần thực hiện

VÍ DỤ:

```
(defun C:vdt()
(setq n 1)
(setq pt1 (list 0 0))
     (while  (<= n 10)
          (setq pt2 (list n (+ n 0.5)))
          (setq n (1+ n))
          (command "LINE" pt1 pt2 "")
     )
     (command "ZOOM" "e")
  )
```

**c. Hàm** `(repeat ...)`

CHỨC NĂNG: Thực hiện các biểu thức, lệnh trong hàm một số lần lặp nhất định

CÚ PHÁP:

```
(repeat int [expr]...)
```

GIẢI THÍCH:

`int`: Là số nguyên chỉ lần lặp

`expr`: Biểu thức, lệnh cần thực hiện

VÍ DỤ:

```
(defun C:vdt()
(setq n 1)
(setq pt1 (list 0 0))
     (repeat 10
          (setq pt2 (list n (+ n 0.5)))
          (setq n (1+ n))
          (command "LINE" pt1 pt2 "")
     )
     (command "ZOOM" "e")
```

```
)
```

**d. Hàm** `(progn ...)`

CHỨC NĂNG: Tập hợp các biểu thức thành một biểu thức và bắt các biểu thức này thực hiện theo một điều khiển chung

CÚ PHÁP:

```
(progn [expr]...)
```

GIẢI THÍCH:

`expr`: Biểu thức, lệnh cần thực hiện

**6. Các hàm xử lý danh sách**

**a. Hàm** `(quote ...)` hoặc `'(...)`

CHỨC NĂNG: Trả ra 1 biểu thức, kiểu là kiểu của biểu thức

CÚ PHÁP:

```
(quote expr) hoặc '(expr)
```

GIẢI THÍCH:

`expr`: Biểu thức. Nếu biểu thức là số có thể tạo danh sách điểm, nếu là biến mặc dù đã gán giá trị bằng số cũng không tạo được danh sách điểm

**b. Hàm** `(list ...)`

CHỨC NĂNG: Tạo ra một danh sách

CÚ PHÁP:

```
(list expr)
```

GIẢI THÍCH:

`expr`: Biểu thức, ít nhất có một tham số. Biểu thức là biến hay là số cũng tạo được danh sách điểm

**c. Hàm** `(car ...)`

CHỨC NĂNG: Hoàn trả phần đầu tiên của danh sách. Kết quả là một giá trị, kiểu là kiểu của thành phần đó.

CÚ PHÁP:

```
(car list)
```

GIẢI THÍCH:

`list`: Là một danh sách

**d. Hàm** `(cadr ...)`

CHỨC NĂNG: Hoàn trả phần phần tử thứ 2 của danh sách. Kết quả là một giá trị, kiểu là kiểu của thành phần đó.

CÚ PHÁP:

```
(cadr list)
```

**e. Hàm** `(caddr ...)`

CHỨC NĂNG: Hoàn trả phần phần tử thứ 3 của danh sách. Kết quả là một giá trị, kiểu là kiểu của thành phần đó.

CÚ PHÁP:

```
(caddr list)
```

**f. Hàm** `(last ...)`

CHỨC NĂNG: Hoàn trả phần phần tử cuối cùng của danh sách. Kết quả là một giá trị, kiểu là kiểu của thành phần đó.

CÚ PHÁP:

```
(last list)
```

**g. Hàm** `(nth ...)`

CHỨC NĂNG: Hoàn trả phần phần tử thứ `int` của danh sách. Kết quả là một giá trị, kiểu là kiểu của thành phần đó. Thứ tự bắt đầu 0, 1, 2, ...

CÚ PHÁP:

```
(nth int list)
```

GIẢI THÍCH:

`int`: Là thứ tự của phần tử cần lấy giá trị

`list`: Là một danh sách

**h. Hàm** `(cdr ...)`

CHỨC NĂNG: Hoàn trả danh sách không có phần tử đầu

CÚ PHÁP:

```
(cdr list)
```

**i. Hàm** `(reverse ...)`

CHỨC NĂNG: Hoàn trả danh sách ngược với danh sách ban đầu

CÚ PHÁP:

```
(reverse list)
```

**k. Hàm** `(length ...)`

CHỨC NĂNG: Hoàn trả số thành phần trong danh sách

CÚ PHÁP:

```
(length list)
```

**m. Hàm** `(append ...)`

CHỨC NĂNG: Gộp các danh sách thành phần thành một danh sách đơn

CÚ PHÁP:

```
(append list1 list2 ...)
```

**n. Hàm** `(member ...)`

CHỨC NĂNG: Tạo một danh sách mới từ một danh sách đã có theo một thành viên cho trước, danh sách mới được tạo ra có các thành viên là cách thành viên của danh sách gốc bắt đầu từ thành viên `expr` cho đến thành viên cuối

CÚ PHÁP: `(member expr list)`

GIẢI THÍCH:

`expr`: Là một trong các phần tử của một danh sách

`list`: Là một danh sách gốc

**o. Hàm** `(assoc ...)`

CHỨC NĂNG: Trả về một danh sách con trong danh sách phức hợp

CÚ PHÁP:

```
(assoc item alist)
```

GIẢI THÍCH:

`item`: Là phần tử đầu tiên của danh sách con trả về, nếu không tìm thấy danh sách con nào chứa phần tử đầu tiên là `item`, hàm sẽ trả giá trị `nil`

`alist`: Phải là một danh sách phức hợp

VÍ DỤ:

```
(setq alist '((1 "ONE") (2 "TWO") (3 "THREE")))

(assoc 1 alist) --->(1 "ONE")
(assoc 2 alist) --->(2 "TWO")
```

**p. Hàm** `(subst ...)`

CHỨC NĂNG: Thay thế các thành viên của một danh sách

CÚ PHÁP:

```
(subst newitem olditem list)
```

GIẢI THÍCH:

`newitem`: Là thành viên mới

`oldwitem`: Là thành viên cũ

`list`: Danh sách gốc

VÍ DỤ:

```
(setq alist '((1 "ONE") (2 "TWO") (3 "THREE")))
```

**7. Các hàm nhập – xuất dữ liệu**

**a. Hàm** `(promt ...)`

CHỨC NĂNG: Xuất một ký tự ra màn hình (dòng command)

CÚ PHÁP:

```
(promt msg)
```

GIẢI THÍCH:

`msg`: Chuỗi ký tự cần xuất ra như một dòng thông báo

**b. Hàm** `(open ...)`

CHỨC NĂNG: Mở một tập tin dữ liệu trên đĩa

CÚ PHÁP:

```
(open filename mode)
```

GIẢI THÍCH:

`filename`: Tên tập tin (file) chứa dữ liệu.

Phần mở rộng của tập tin là *.txt hoặc bất kỳ do người dùng đặt

Nếu tập tin này không nằm trong thư mục làm việc của Autocad phải cho đường dẫn

`mode`: Mã của tập tin

"`r`" : Đọc tập tin đang có trên đĩa

"`w`" : Ghi vào tập tin, mỗi lần ghi tạo một tập tin mới

"`a`" : Ghi tiếp vào tập tin đang tồn tại hoặc tạo tập tin mới để ghi nếu chưa có tập tin trên đĩa

**c. Hàm** `(close ...)`

CHỨC NĂNG: Đóng tập tin dữ liệu đang mở bằng hàm `open` để giải phóng bộ nhớ cho các chương trình khác làm việc

CÚ PHÁP:

```
(close filename)
```

GIẢI THÍCH:

`filename`: Tên tập tin chứa dữ liệu được mở bằng hàm `open`

**d. Hàm** `(findfile ...)`

CHỨC NĂNG: Tìm tập tin dữ liệu trên đĩa, nếu thấy AutoLisp sẽ trả về tên tập tin kèm theo đường dẫn

CÚ PHÁP:

```
(findfile filename)
```

**e. Hàm** `(read-line ...)`

CHỨC NĂNG: Đọc một dòng ký tự từ bàn phím hoặc tập tin

CÚ PHÁP:

```
(read-line [fp])
```

GIẢI THÍCH:

`fp` : Tên tập tin chứa dữ liệu vừa được mở bằng lệnh `open`

Nếu có ghi fp hàm sẽ đọc một tập tin này và hoàn trả 1 chuỗi ký tự

Nếu không có hàm sẽ đọc một chuỗi ký tự được gõ vào từ bàn phím

Mỗi lần gọi hàm `(read-line...)` một dòng dữ liệu tiếp theo được đọc vào, khi nào không còn dữ liệu thì kết quả là nil

**f. Hàm** `(read...)`

CHỨC NĂNG: Đọc một chuỗi ký tự và hoàn trả giá trị tham số phù hợp với kiểu dữ liệu nhập vào

CÚ PHÁP:

```
(read str)
```

GIẢI THÍCH:

`str` : Chuỗi ký tự hoặc một biến kiểu chuỗi

**g. Hàm** `(read-char...)`

CHỨC NĂNG: Đọc một ký tự từ bàn phím hoặc từ tệp, kết quả trả về là một số nguyên mã ASCII của ký tự vừa đọc

CÚ PHÁP:

```
(read-char [fr])
```

GIẢI THÍCH:

`[fr]` : Tên tập tin chứa dữ liệu được mở bằng lệnh open, nếu không có máy chờ người dùng nhập vào từ bàn phím

**h. Hàm** `(write-char...)`

CHỨC NĂNG: Ghi một ký tự ra vùng dòng lệnh trên màn hình hoặc vào tập tin

CÚ PHÁP:

```
(write-char n [fr])
```

GIẢI THÍCH:

`n` : Số nguyên mã ASCII của ký tự ghi vào

`[fr]` : Tên tập tin chứa dữ liệu được mở bằng lệnh open, nếu không có máy chờ người dùng nhập vào từ bàn phím

**i. Hàm** `(write-line...)`

CHỨC NĂNG: Ghi một chuỗi ký tự ra vùng dòng lệnh trên màn hình hoặc vào tệp

CÚ PHÁP:

```
(write-line str [fr])
```

GIẢI THÍCH:

`str` : chuỗi

`[fr]` : Tên tập tin chứa dữ liệu được mở bằng lệnh open, nếu không có máy chờ người dùng nhập vào từ bàn phím

**k. Các hàm** `prin1, princ, print`

CHỨC NĂNG: In kết quả ra vùng dòng lệnh trên màn hình hoặc vào tệp

CÚ PHÁP:

```
(prin1 expr [fr])
(princ expr [fr])
(print expr [fr])
```

GIẢI THÍCH:

`expr` : Biểu thức

`[fr]` : Tên tập tin chứa dữ liệu được mở bằng lệnh open, nếu không có máy chờ người dùng nhập vào từ bàn phím

Sự khác nhau giữa các hàm:

- Các ký tự điều khiển như: `"\n", "\r"...` không có tác dụng đối với hàm `prin1` mà có tác dụng đối với hàm `princ`
- Hàm `print` luôn ghi kết quả xuống dòng mới và sau kết quả có một dấu trống

**8. Các hàm kiểm tra dữ liệu**

**a. Hàm** `(type...)`

CHỨC NĂNG: Xác định kiểu dữ liệu

CÚ PHÁP:

```
(type item)
```

GIẢI THÍCH:

`item` : Là biến hay biểu thức

**b. Hàm** `(atom...)`

CHỨC NĂNG: Nếu tham số không phải là danh sách trả ra T, ngược lại Nil

CÚ PHÁP:

```
(atom item)
```

GIẢI THÍCH:

`item` : Là tham số

**c. Hàm** `(listp...)`

CHỨC NĂNG: Nếu tham số là danh sách trả ra T, các kiểu còn lại Nil

CÚ PHÁP:

```
(listp item)
```

GIẢI THÍCH:

`item` : Là tham số

**d. Hàm** `(numberp...)`

CHỨC NĂNG: Nếu tham số là số trả ra T, các kiểu còn lại Nil

CÚ PHÁP:

```
(numberp item)
```

GIẢI THÍCH:

`item` : Là tham số

**e. Hàm** `(nminusp...)`

CHỨC NĂNG: Nếu tham số là số âm trả ra T, các trường hợp khác Nil

CÚ PHÁP:

```
(nminusp number)
```

**f. Hàm** `(zerop...)`

CHỨC NĂNG: Nếu tham số là số 0 trả ra T, các trường hợp khác Nil

CÚ PHÁP:

```
(zerop number)
```

**9. Các hàm chuyển đổi kiểu dữ liệu**

**a. Hàm** `(angtof...)`

CHỨC NĂNG: Chuyển đổi chuỗi ký tự (chứa các chữ số) thành số thực dùng cho góc và trả về số đo góc bằng `radian`

CÚ PHÁP:

```
(angtof str [mode])
```

GIẢI THÍCH:

`str` : Là chuỗi chứa chữ số

`mode`: Là số nguyên xác định dạng đơn vị nhập vào, nếu không có sẽ tuân theo các cài đặt của lệnh UNITS

`Mode` nhận các giá trị sau:

0 : Dạng thập phân (độ)

1 : Dạng độ/ phút/ giây

2 : Dạng Grads

3 : Dạng Radian

4 : Dạng trắc đạc có kèm theo chỉ hướng

**b. Hàm** `(angtos...)`

CHỨC NĂNG: Chuyển đổi số đo góc thành chuỗi ký tự (chứa các chữ số)

CÚ PHÁP:

```
(angtos angle [mode] [precision]])
```

GIẢI THÍCH:

`angle` : Là số đo góc bằng `radian`

`mode` : Là số nguyên xác định dạng đơn vị xuất ra, nếu không có sẽ tuân theo các cài đặt của lệnh UNITS. Các giá trị của `mode` tương tự như hàm `(angtof...)`

`precision` : Là độ chính xác (số chữ số sau dấu phẩy thập phân)

**c. Hàm** `(atof...)`

CHỨC NĂNG: Chuyển đổi chuỗi ký tự thành số thực

CÚ PHÁP:

```
(atof str)
```

GIẢI THÍCH:

`str` : Là chuỗi ký tự (chứa các chữ số)

**d. Hàm** `(atoi...)`

CHỨC NĂNG: Chuyển đổi chuỗi ký tự thành số nguyên

CÚ PHÁP:

```
(atoi str)
```

GIẢI THÍCH:

`str` : Là chuỗi ký tự (chứa các chữ số)

**e. Hàm** `(itoa...)`

CHỨC NĂNG: Chuyển đổi số nguyên thành chuỗi ký tự

CÚ PHÁP:

```
(itoa int)
```

GIẢI THÍCH:

`int` : Là số nguyên, nếu cho kiểu khác hàm sẽ báo lỗi

**f. Hàm** `(rtos...)`

CHỨC NĂNG: Chuyển đổi số thực thành chuỗi ký tự

CÚ PHÁP:

```
(rtos number [mode [precision]])
```

GIẢI THÍCH:

`number` : Là một số

`mode`: Là mã điều khiển dạng xuất ra chuỗi ký tự

`Mode` nhận các giá trị sau:

1 : Dạng khoa học

2 : Dạng thập phân

3 : Dạng kỹ thuật

4 : Dạng kiến trúc

5 : Dạng hữu tỷ (phân số)

`precision` : Là độ chính xác (số chữ số sau dấu phảy thập phân)

**g. Hàm** `(distof...)`

CHỨC NĂNG: Chuyển đổi chuỗi ký tự (chứa các chữ số) thành số thực

CÚ PHÁP:

```
(distof str [mode])
```

GIẢI THÍCH:

`str` : Là chuỗi ký tự chứa các chữ số

`mode`: Là mã điều khiển nhập vào của số thực nhận các giá trị sau:

1 : Dạng khoa học

2 : Dạng thập phân

3 : Dạng kỹ thuật

4 : Dạng kiến trúc

5 : Dạng hữu tỷ (phân số)

**h. Hàm** `(fix...)`

CHỨC NĂNG: Chuyển đổi số thực thành số nguyên, cắt bỏ phần thập phân

CÚ PHÁP:

```
(fix number)
```

GIẢI THÍCH:

`number` : Là một số thực

**i. Hàm** `(fload...)`

CHỨC NĂNG: Chuyển đổi số thành số thực

CÚ PHÁP:

```
(fload number)
```

GIẢI THÍCH:

`number` : Là một số nguyên hoặc số thực

**10. Các hàm xử lý chuỗi ký tự**

Trong AutoLisp chuỗi là các ký tự được viết trong ngoặc kép " ". Dấu \ kèm theo các chữ cái có tác dụng điều khiển:

| Cách viết | | ý nghĩa |
|---|---|---|
| `\\` | : | \ |
| `\n` | : | Xuống dòng |
| `\t` | : | Tab |
| `\e` | : | ESC |
| `\"` | : | " |
| `\r` | : | ENTER |

**a. Hàm** `(strcat...)`

CHỨC NĂNG: Nối các chuỗi thành phần thành một chuỗi chung

CÚ PHÁP:

```
(strcat str1 str2 [str3]...)
```

GIẢI THÍCH:

`str1, str2, str3...` : Là các chuỗi thành phần cần được ghép lại

**b. Hàm** `(strcase...)`

CHỨC NĂNG: Chuyển chuỗi ra chữ in hoa hoặc chữ thường

CÚ PHÁP:

```
(strcase str which)
```

GIẢI THÍCH:

`str`: Là chuỗi cần chuyển đổi

`which`: Nếu không có, tất cả chuỗi biến thành chữ in hoa, nếu là chữ T tất cả chuỗi trở thành chữ thường

**c. Hàm** `(strlen...)`

CHỨC NĂNG: Đếm số ký tự trong chuỗi

CÚ PHÁP:

```
(strlen str)
```

GIẢI THÍCH:

`str`: Là chuỗi cần đếm

**d. Hàm** `(substr...)`

CHỨC NĂNG: Trích một phần của chuỗi ký tự thành một chuỗi khác

CÚ PHÁP:

```
(substr str start [length])
```

GIẢI THÍCH:

`str`: Là chuỗi cần trích

`start`: Vị trí ký tự trong chuỗi cần trích ra

`length`: Chiều dài của chuỗi mới (số ký tự), nếu không có thì chuỗi mới bắt đầu từ vị trí `start` cho đến hết

## 11. Các hàm xử lý chuỗi ký tự

### *11.1. CÁC HÀM THAO TÁC VỚI CÁC ĐỐI TƯỢNG ĐỘC LẬP*

**a. Hàm** `(entlast)`

CHỨC NĂNG: Chọn đối tượng vẽ cuối cùng trong bản vẽ và trả về mã ENAME của nó

CÚ PHÁP:

```
(entlast)
```

GIẢI THÍCH: Hàm này không có đối số

**b. Hàm** `(entnext...)`

CHỨC NĂNG: Chọn đối tượng vẽ đầu tiên trong bản vẽ hoặc đối tượng tiếp sau một đối tượng khác và trả về mã ENAME của nó

CÚ PHÁP:

```
(entnext [ename])
```

GIẢI THÍCH:

`ename`: Nếu không có hoàn trả thực thể đầu tiên của bản vẽ

Nếu có hoàn trả tên thực thể tiếp sau thực thể có tên là `ename`

VÍ DỤ:

```
(setq e1 (entnext))     ;Gán tên thực thể đầu tiên cho e1
(setq e2 (entnext e1)) ;Gán tên thực thể tiếp sau e1 cho e2
```

**c. Hàm** `(entsel...)`

CHỨC NĂNG: Chờ người dùng chọn đối tượng và trả về mã ENAME của nó

CÚ PHÁP:

```
(entsel [promt])
```

GIẢI THÍCH:

`promt`: Lời nhắc, nếu không có dòng nhắc sẽ là : `Select object:`

VÍ DỤ:

```
(setq e1 (entsel "Chon doi tuong thu nhat:"))
(setq e2 (entsel "Chon doi tuong thu hai:"))
(command "EXTEND" e1 "" e2 "")
```

**d. Hàm** `(nentselp...)`

CHỨC NĂNG: Chương trình tự động chọn đối tượng đi qua một điểm cho trước

CÚ PHÁP:

```
(nentselp [promt] [pt])
```

GIẢI THÍCH:

`promt`: Lời nhắc

`pt`: Điểm mà đối tượng đi qua

**e. Hàm** `(entdel...)`

CHỨC NĂNG: Xoá một đối tượng trong bản vẽ hoặc khôi phục lại đối tượng vừa bị xoá

CÚ PHÁP:

```
(entdel ename)
```

GIẢI THÍCH:

`ename`: Mã tên đối tượng, hàm này chỉ dùng cho thực thể chính, tên thực thể do hàm `(entlast)`, `(entnext)`

**f. Hàm** `(entget...)`

CHỨC NĂNG: Hoàn trả danh sách liệt kê các thông tin về đối tượng có trong bản vẽ được gọi

CÚ PHÁP:

```
(entget ename [applist])
```

GIẢI THÍCH:

`ename`: Mã tên đối tượng

`applist`: Nếu có, danh sách các chuỗi ký tự chứa các thông tin mở rộng có liên quan sẽ được gọi ra và hoàn trả.

**g. Hàm** `(entmod...)`

CHỨC NĂNG: Cập nhật lại thông tin của đối tượng đã bị thay đổi bằng các hàm và vẽ ra đối tượng với các thông tin đã cập nhật. Hàm này thường được dùng với hàm `(entget)`

CÚ PHÁP:

```
(entmod entlist)
```

GIẢI THÍCH:

`entlist`: Là danh sách chứa thông tin của đối tượng lấy từ hàm `(entget)`

**h. Hàm** `(entmake...)`

CHỨC NĂNG: Tạo một thực thể mới bằng cách xây dựng một danh sách các cớ dữ liệu của nó, không cần qua các lệnh của AutoCad

CÚ PHÁP: `(entmake edata)`

GIẢI THÍCH:

`edata`: Là danh sách các cơ sở dữ liệu của đối tượng cần tạo.

VÍ DỤ: Vẽ đường tròn

```
(entmake
    '((0 . "CIRCLE") ; Kiểu đối tượng đường tròn
    (62 . 1)         ; Màu đỏ
    (10 4.0 4.0 0.0) ; Tâm (4.0 4.0 0.0)
    (40 . 1.0)
    (40 . 1.0)       ; Bán kính 1.0
    )
)
```

## 11.2. CÁC HÀM THAO TÁC VỚI BẢNG CÁC ĐỐI TƯỢNG

Bảng các thực thể là tập hợp các thực thể trong một khối đồ hoạ chuẩn của AutoCad. Các thực thể này không thấy hiện hữu trên bản vẽ mà nằm trong các bảng dữ liệu. Các bảng dữ liệu đó bao gồm: LAYER, BLOCK, LTYPE, STYLE, DIMSTYLE, UCS, VPORT...

Mỗi bảng đối tượng nói trên có 3 dữ liệu chính:

+ Tên bảng như: "LAYER", "BLOCK"...

+ Tên đối tượng trong bảng: Lớp "0", khối "A1"

+ Dữ liệu của đối tượng: màu sắc, đường nét....

**a. Hàm** `(tblnext...)`

CHỨC NĂNG: Tìm bảng dữ liệu và trả ra dữ liệu của bảng đó

CÚ PHÁP:

```
(tblnext tblname [rewind])
```

GIẢI THÍCH:

`tblname`: Chuỗi ký tự chứa tên bảng

`rewind`: Nếu có giá trị khác nil, sẽ trả về tên và các dữ liệu của bảng các đối tượng đầu tiên mà AutoCad tìm thấy.Nếu không có hoặc nhận giá trị nil, sẽ trả về tên và các dữ liệu của bảng các đối tượng tiếp theo mà AutoCad tìm thấy

**b. Hàm** `(tblsearch...)`

CHỨC NĂNG: Tìm thành phần được chỉ định của bảng đối tượng và trả ra dữ liệu của thành phần đó

CÚ PHÁP:

```
(tblsearch tblname symbol [setnext])
```

GIẢI THÍCH:

`tblname`: Chuỗi ký tự chứa tên bảng

`symbol`: Tên thành phần của bảng

Nếu tìm thấy, sẽ trả về tên và các dữ liệu của tên gọi này

Nếu không tìm thấy sẽ trả về nil

`setnext`: Là lực chọn dùng khi có một lệnh `tblnext` tiếp sau lệnh này

## 11.3. CÁC HÀM THAO TÁC VỚI NHÓM CÁC ĐỐI TƯỢNG

Trong AutoCad người ta sử dung chữ viết tắt "ss" (Selection Set) để gọi tên nhóm đối tượng (Object) hoặc nhóm thực thể (entities) được chọn sau dòng nhắc "Select object:" của AutoCad. Kiểu của nhóm là PICKSET. Có thể coi nhóm đối tượng là một mảng các phần tử, mỗi phần tử là tên một đối tượng

**a. Hàm** `(ssget...)`

CHỨC NĂNG: Chọn các đối tượng trong bản vẽ và trả về nhóm đối tượng đã chọn. Nếu không có đối tượng nào hàm trả kết quả là nil

CÚ PHÁP:

```
(ssget [mode] [pt1 [pt2]] [pt-list] [filter-list])
```

GIẢI THÍCH:

`mode`: Chuỗi ký tự chứa các mã điều khiển việc lựa chọn, nhận các mã sau:

"L" : Last - đối tượng vẽ cuối cùng còn tồn tại trong bản vẽ được chọn

"P": Previous – Tập hợp đối tượng vừa chọn trước đó

"W" : Windows – Các đối tượng nằm trong cửa sổ chọn

"C" : Cross – Các đối tượng nằm trong và cắt qua cửa sổ chọn

"X" : Tất cả các đối tượng trong bản vẽ được chọn

"WP" : Các đối tượng nằm trong một đa giác cho trước được chọn

"WC" : Các đối tượng nằm trong và cắt qua một đa giác cho trước được chọn

`pt1 pt2`: Hai điểm là 2 đỉnh đối nhau của một cửa sổ hình chữ nhật dùng cho chọn đối tượng bằng Crossing và windows

`pt-list`: Các điểm tạo thành đa giác dùng cho lựa chọn WP, WC

`filter-list`: Là danh sách chứa các tiêu chuẩn lọc cho việc chọn nhóm các đối tượng

**b. Hàm** `(ssadd...)`

CHỨC NĂNG: Thêm một đối tượng vào tập đối tượng

CÚ PHÁP:

```
(ssadd [ename] [ss])
```

GIẢI THÍCH:

`ename`: Chuỗi ký tự chứa tên mã đối tượng cần thêm vào, nếu đối tượng ename đã có trong tập đối tượng, không thêm vào được nữa

`ss`: Tên tập đối tượng

Nếu cả 2 tham số này không có, sẽ trả ra một tập rỗng

Nếu ename khác nil tập đối tượng là nil, hàm sẽ trả ra một tập đối tượng chỉ có một phần tử là ename đã cho

Nếu cả 2 đều khác nil, hàm sẽ trả ra tập đối tượng có thêm phần tử ename vừa thêm vào

**c. Hàm** `(sslength...)`

CHỨC NĂNG: Trả ra số nguyên là số các đối tượng có trong tập các đối tượng

CÚ PHÁP:

```
(sslength ss)
```

GIẢI THÍCH:

`ss`: Tên tập đối tượng có trong bản vẽ

**d. Hàm** `(ssdel...)`

CHỨC NĂNG: Loại bỏ một đối tượng ra khỏi tập đối tượng

CÚ PHÁP:

```
(ssdel ename ss)
```

GIẢI THÍCH:

`ename`: Chuỗi ký tự chứa mã tên đối tượng cần loại bỏ. Nếu đối ename này đã có trong tập đối tượng, thì nó bị loại bỏ, nếu không có trả nguyên tập đối tượng

`ss`: Tên tập đối tượng có trong bản vẽ

**e. Hàm** `(ssname...)`

CHỨC NĂNG: Trả về ename của một đối tượng có trong tập các đối tượng

CÚ PHÁP:

```
(ssname ss index)
```

GIẢI THÍCH:

`ss`: Tên tập đối tượng có trong bản vẽ

`index`: Số thứ tự của các đối tượng trong tập hợp các đối tượng được tính từ 0, 1, 2....

## IV> HÀM DO NGƯỜI LẬP TRÌNH ĐỊNH NGHĨA

Các hàm này tương tự như hàm (Function) trong ngôn ngữ lập trình C và C++, thủ tục (Procedure) trong Pascal, hoặc hàm con (Subroutine) trong Fortran

Cú pháp chung của hàm do người dùng định nghĩa:

CÚ PHÁP:

```
(defun tên_hàm (tham_số/ biến_số_riêng)
.........................; Thân hàm
.........................; Thân hàm
)
Hoặc:
(defun tên_hàm ()
.........................; Thân hàm
.........................; Thân hàm
)
```

GIẢI THÍCH:

`defun`: Hàm định nghĩa của AutoLisp

`Tên_hàm`: Do người lập trình đặt, tên hàm nên viết bằng chữ hoa và không có khoảng trống

`Thâm hàm`: là các lệnh xử lý của AutoLisp. Một hàm có thể triệu gọi nhiều hàm khác

Trong thân hàm bao gồm có các `tham số, biến_số_riêng, biến_số_chung`

`tham số`: Là một hoặc nhều biến hình thức, các biến này chung cho cả chương trình, tham gia vào các biểu thức trong thân hàm và được tồn tại khi thoát ra khỏi AutoCad

Khi được triệu gọi, các biến hình thức này được thay bằng các giá trị

Trong hàm có thể không có tham số

`biến_số_riêng`: Trong hàm có thể có một hay nhiều biến số riêng, được phân cách nhau bằng dấu trống

Các biến số này nhận giá trị hoặc định nghĩa và chỉ tồn tại bên trong hàm, khi ra khỏi hàm, giá trị các biến này sẽ được xoá khỏi bộ nhớ

`biến_số_chung`: Khi hàm không có cả tham số lẫn biến riêng, thì các biến số của nó mặc nhiên là biến chung

VÍ DỤ:

**a. Ví dụ về tham số:**

```
(defun VD1(x y)
.............................; Thân hàm
.............................; Thân hàm
)
```

Hàm này có 2 tham số x và y. Hàm được triệu gọi như sau:

```
(VD1 x y)
```

**b. Ví dụ về biến số riêng:**

```
(defun VD2(/ x y)
.............................; Thân hàm
.............................; Thân hàm
)
```

Hàm này có 2 biến số riêng x và y. Hàm được triệu gọi như sau:

```
(VD2)
```

**c. Ví dụ về biến số chung:**

```
(defun VD3()
.............................; Thân hàm
.............................; Thân hàm
)
```

Hàm này không có tham số và biến riêng. Hàm được triệu gọi như sau:

```
(VD3)
```

**d. Tên hàm là một lệnh của AutoCad:**

```
(defun C:VD4()
.............................; Thân hàm
.............................; Thân hàm
)
```

Khi chương trình AutoLisp chứa hàm trên được gọi vào AutoCad thì 'vd4" là một lệnh của AutoCad, có thể gọi trên dòng lện Command: vd4↵

## 1. Các trường hợp đặc biệt của tên hàm:

### *1.1. Tên hàm trở thành một lệnh của AutoCad*

```
(defun C:tên_hàm(...)
.........................; Thân hàm
.........................; Thân hàm
)
```

### *1.2. Hàm được thực hiện ngay sau khi khởi động AutoCad*

```
(defun S::tên_hàm(...)
.........................; Thân hàm
.........................; Thân hàm
)
```

## 2. Một số ví dụ minh hoạ

### *2.1. Tạo Layer*

```
(defun C:TAOLOP()
  (command "-layer" "n" "dut" "l" "DASHED" "dut" "c" 3 "dut" "")
  (command "-layer" "n" "tam" "l" "CENTER" "tam" "c" 1 "tam" "")
  (command "Ltscale" 300)
  (command "ZOOM" "a")
); Kết thúc hàm
```

Chương trình khi chạy trong AutoCad sẽ tạo 2 lớp có tên là “dut” và “tam”

### *2.2. Vẽ khung tên*

```
(defun c:KBV(/ tyle loai KBV1 KBV2)
(setq tyle (getreal "\Cho ty le ban ve : "))
(setq loai (getstring "\Cho kho giay in ban ve A0,A1,A2,A3,A4:
<A3> "))
(setq  KBV1 (getpoint "\Cho diem chen khung ban ve: "))
     (if (= loai "a0")
     (setq  KBV2 (list (+ (car KBV1) (* 1188.0 tyle)) (+ (cadr
      KBV1) (* 840.0 tyle))))
          (if (= loai "a1")
          (progn
          (setq  KBV2 (list (+ (car KBV1) (* 840.0 tyle)) (+
          (cadr KBV1) (* 594.0 tyle)))))
```

```
            (if (= loai "a2")
            (progn
             (setq  KBV2 (list (+ (car KBV1) (* 560.0 tyle)) (+
             (cadr KBV1) (* 418.0 tyle)))))
            (if (= loai "a4")
            (progn
             (setq  KBV2 (list (+ (car KBV1) (* 285.0 tyle)) (+
             (cadr KBV1) (* 198.0 tyle)))))

            (if (= loai "a3")
            (setq  KBV2 (list (+ (car KBV1) (* 396.0 tyle)) (+
            (cadr KBV1) (* 285.0 tyle))))
            )
            )
            )
            )
      )
(command "-layer" "n" "Dam" "l" "CONTINUOUS" "Dam" "c" 5 "Dam"
"")
(command "-layer" "s" "Dam" "")
(command "Rectangle" KBV1 KBV2 )
); Hết hàm
```

Chương trình chạy trong AutoCad sẽ vẽ kích thước khung bản vẽ theo khổ bản vẽ mà người dùng yêu cầu

### *2.3. Đổi màu của đối tượng*

```
(defun C:1()
            (prompt "_.change ")
            (princ "\n Change Colour to Red - 1")
            (setq sset (ssget))
            (if (null sset)
                  (progn
                    (princ "\nERROR: Nothing selected.")
                    (exit)
                  )
                )
                (command "_.change" sset "" "P" "C" "1" "")
```

```
                (princ)
); Hết chương trình
```

Chương trình chạy trong AutoCad sẽ đổi màu của đối tượng chọn sang màu đỏ

### *2.4. Đổi lớp của đối tượng*

```
(defun C:DAM()
       (prompt "_.change ")
       (princ "\n Change Layer Dam")
       (setq sset (ssget))

       (if (null sset)
         (progn
           (princ "\nERROR: Nothing selected.")
           (exit)
         )
       )
        (command "_.change" sset "" "P" "LA" "Dam" "LT"
        "BYLAYER" "C" "BYLAYER" "")
      (princ)
      ); Hết chương trình
```

Chương trình chạy trong AutoCad sẽ đổi lớp hiện hành của đối tượng sang lớp “dam”

### *2.5. Tạo underline cho một nhóm text được chọn*

```
(defun c:UU( / sset ssl temp ed old new )
(text_uu_ssget); Gọi hàm con
(setq ssl (sslength sset))
(while (> ssl 0)
        (progn
        (setq temp (ssname sset (setq ssl (1- ssl))))
        (setq ed (entget temp))

         (setq old (cons 1 (DXF 1 ed))
               new (cons 1 (strcat "%%u" (DXF 1 ed)))
               ed (subst new old ed)
             )
        (entmod ed)
```

```
            )
      (princ)
  )

 );Hết hàm chính
;-------------------------------------------------------------
(defun text_uu_ssget ( / ssl  nsset temp ed ); Hàm con
       (setq sset (ssget))
       (setq ssl (sslength sset)
             nsset (ssadd)
       )

       (print ssl)
       (while (> ssl 0)
              (progn
                (setq temp (ssname sset (setq ssl (1- ssl))))
                (setq ed (entget temp))
                (if (= (DXF 0 ed) "TEXT") (ssadd temp nsset))
              )
       )
       (setq ssl (sslength nsset)
             sset nsset
       )
       (print ssl)
       (princ)
       (print)
);Hết hàm con
```