Phần 2. AUTOLISP

Chương 1. CĂN BẢN VỀ AUTOLISP

LISP là một ngôn ngữ lập trình bậc cao thường được dùng cho việc nghiên cứu trí tuệ nhân tạo. **LISP** viết tắt của **List Processing** đã được **Jonh McCarthy** và các đồng nghiệp tại viện kỹ thuật **Massachusets** biên soạn từ những năm đầu của thập niên 1960. LISP là 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 danh sách đó.

Một số phiên bản của LISP được tiêu chuẩn hoá, cấu hình đầy đủ và được tiêu chuẩn hoá 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 hoạ. **AutoLISP** là ngôn ngữ thông dịch, được viết theo các 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 AutCAD.

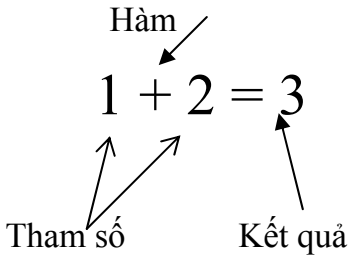
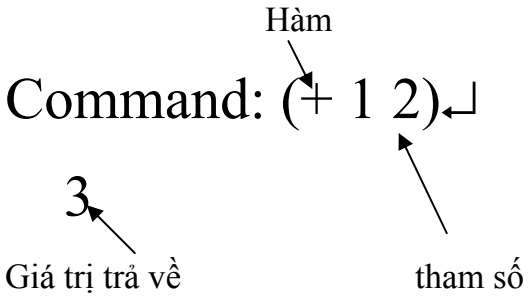
1.1. **Xây dựng biểu thức AutoLISP.**

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 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 cặp dấu ngoặc đơn (). Các phần tử tron danh sách phân cách nhau bởi các khoảng trắng.

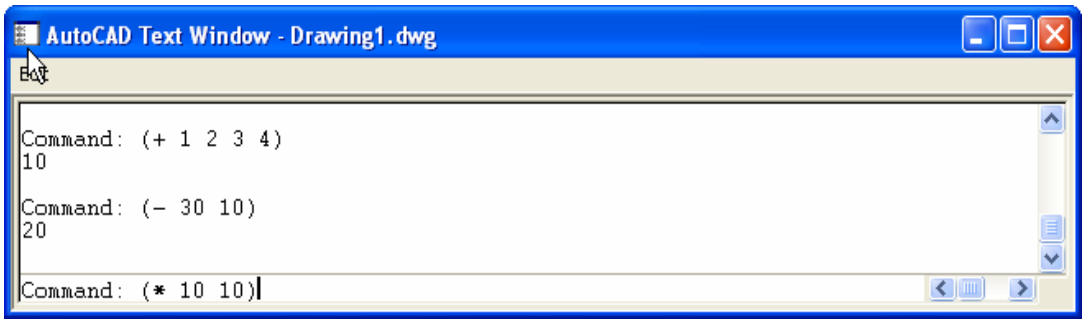
Có hai loại danh sách: biểu thức (*expression*) và danh sách dữ liệu (*data list*). Biểu thức là thành phần cơ bản trong tất cả các chương trình **AutoLISP**. Phần tử đầu tiên của biểu thức là một hàm (**function**). Hàm này sẽ được **AutoLISP** định giá trị và trả về kết quả. Các phần tử tiếp theo là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

Sự khác nhau cơ bản giữa một biểu thức **AutoLISP** với một biểu thức toán học là các phần tử **AutoLISP** có thứ tự khác với đẳng thức và chúng phải được chứa trong cặp dấu ngoặc đơn.

ĐẲNG THỨC TOÁN HỌC	BIỂU THỨC AUTOLISP
	

1.2. Cách nhập biểu thức AutoLISP

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



Hinh1.1. Cửa sổ AutoCAD Text Window.

Khi nhập biểu thức tại dòng lệnh, ta nên dùng cửa sổ AutoCAD Text Window (nhấn phím F2) nhờ đó ta thấy được giá trị trả về hoặc các thông báo lỗi nếu có.
Khi biểu thức được đưa vào từ dòng nhắc lệnh, kết quatrar về ngay tại cửa sổ dòng nhắc lệnh. Nếu ta kết thúc biểu thức bằng nhấn phím ↵, kết quả trả về dòng nhắc tiếp theo. Nếu ta kết thúc biểu thức bằng phím SPACEBAR, kết quả trả về trên cùng một dòng.

Ví dụ:

Command: (* 10 10) 100 (kết thúc biểu thức bằng phím SPACEBAR)
Command: (- 30 10)↵ (kết thúc biểu thức bằng nhấn phím ↵)
20

Nếu biểu thức không bị lỗi kết quả 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:

Command: (+ 1 2 3 4)↵ 10	Không bị lỗi, kết quả trả về là 10
Command: (+6 8) ; error: bad function: +6	Biểu thức này cần một khoảng trống để ngăn cách tên hàm + với tham số 6
Command: (+ 8 .45) ; error: misplaced dot on input	Tham số .45bị sai, cần phải ghi đầy đủ là 0.45
Command: (+ 6 8 (_>) 14	Biểu thức chưa được đóng lại bằng dấu ngoặc đơn. Dòng kế tiếp máy báo thiếu dấu và cần đóng lại.
Command: (+ 6 8 (_> *Cancel* ; error: Function cancelled	Thay vì đóng dấu ngoặc đơn ta thoát khỏi hàm.

1.3. **Các hàm số học.**

a) Hàm cộng (+).

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

(+ [number number] ...)

Examples

(+ 1 2) returns 3
(+ 1 2 3 4.5) returns 10.5
(+ 1 2 3 4.0) returns 10.0

b) Hàm trừ (-):

(– [number number] ...)

Thực hiện phép trừ number thứ nhất cho các number còn lại.

Examples

(- 50 40) returns 10
(- 50 40.0) returns 10.0
(- 50 40.0 2.5) returns 7.5
(- 8) returns -8

c) Hàm nhân (*):

(* [number number] ...)

Nhân các number lại với nhau.

Examples

(* 2 3) returns 6
(* 2 3.0) returns 6.0
(* 2 3 4.0) returns 24.0
(* 3 -4.5) returns -13.5
(* 3) returns 3

d) Hàm chia (/):

(/ [number number] ...)

Thực hiện phép chia số thứ nhất cho các số còn lại.

Examples

(/ 100 2) returns 50 (khi các số đều nguyên thi kết quả là phần nguyên của phép chia)
(/ 100 2.0) returns 50.0
(/ 100 20.0 2) returns 2.5
(/ 100 20 2) returns 2
(/ 4) returns 4

1.4. Các biến và ký hiệu trong AutoLISP.

Các giá trị tĩnh không đổi như tên hàm của **AutoLISP** (+, -, *, /, …) và các tên hàm tự tạo, hoặc các hằng số (như Pi) gọi chung là các ký hiệu (symbol). Các dữ liệu thay đổi trong chương trình gọi là biến (variable). Dữ liệu trong các biến thay đổi tuỳ 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 nhau. Tên gọi phụ thuộc vào giá trị của chúng là tĩnh hay động.
Tên biến và và ký hiệu (cũng như tên hàm) không phân biệt chữ hoa chữ thường.

1.4.1. Gán giá trị cho các biến.

Dùng hàm Setq để gán giá trị cho các biến, hàm sẽ trả kết quả bằng giá trị gán. Hàm **setq** có thể gán mọi kiểu dữ liệu cho bất kỳ biến nào: số nguyên, số thực, chuỗi, danh sách, hoặc các kiểu khác.
Cú pháp hàm Setq:

(setq sym expr [sym expr]...)

Hàm **setq** có thể gán một lúc nhiều giá trị cho nhiều biến.
Ví dụ:

Command: (setq a 5.0)

5.0

Command: (setq b 123 c 4.7)

4.7 (gán giá trị cho nhiều biến, kết quả trả về là giá trị biến cuối cùng)

Command: (setq x '(a b)) (gán giá trị biến x bằng danh sách (A B)

(A B)

Chương 2. FILE CHƯƠNG TRÌNH AutoLISP

2.1. File chương trình AutoLISP

Khi cần định giá một biểu thức AutoLISP đơn giản, ta nhập trực tiếp tại dòng nhắc lệnh AutoCAD. Khi cần thực hiện liên tiếp nhiều biểu thức phức tạp, ta có thể lưu chúng vào một file văn bản ASCII, và sau đó gọi file này để thực hiện. File này gọi là file chương trình AutoLISP.

Các ưu điểm khi sử dụng file chương trình AutoLISP:

- Ta chỉ cần một lần tạo ra các biểu thức AutoLISP và sau đó có thể sử dụng chúng được nhiều lần.

- Khi gọi thực hiện file chương trình, ta có thể an tâm là nó đã được kiểm tra lỗi cẩn thận.

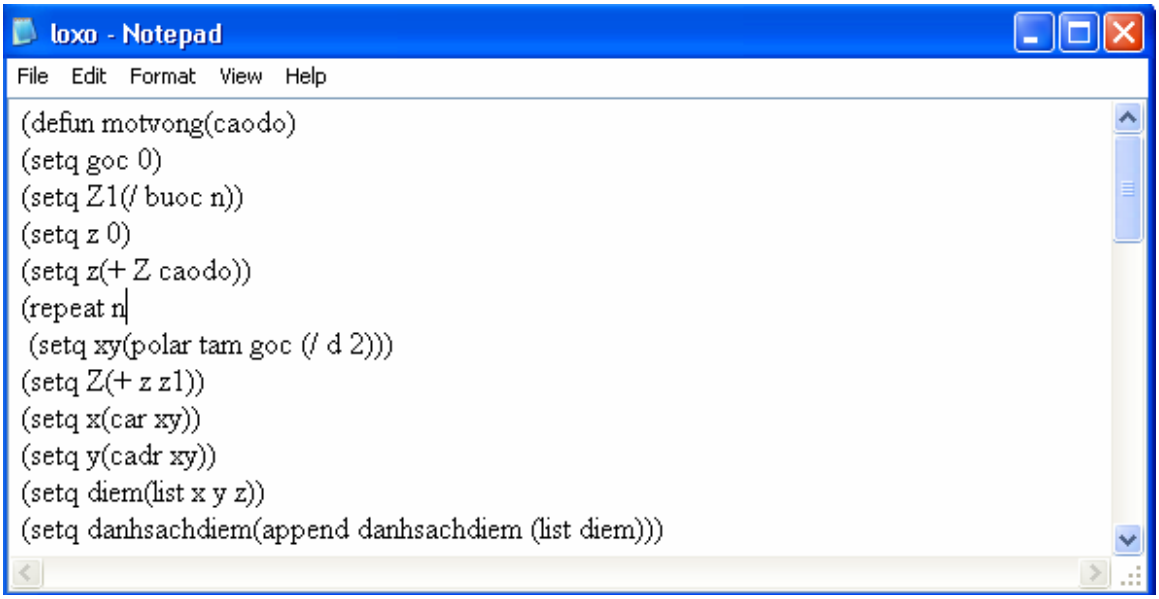
- **AutoCAD** tính toán các biểu thức trong file chương trình nhanh hơn khi chúng được nhập từ dòng nhắc lệnh.

2.1.1. Tên file AutoLISP

Tên file AutoLISP phụ thuộc vào hệ điều hành. Khi dùng **Windows 95, 98, 2000, NT** và các phiên bản mới hơn ta có thể đặt tên file dài đến 256 ký tự. Phần mở rộng mặc định của file là **.LSP.**

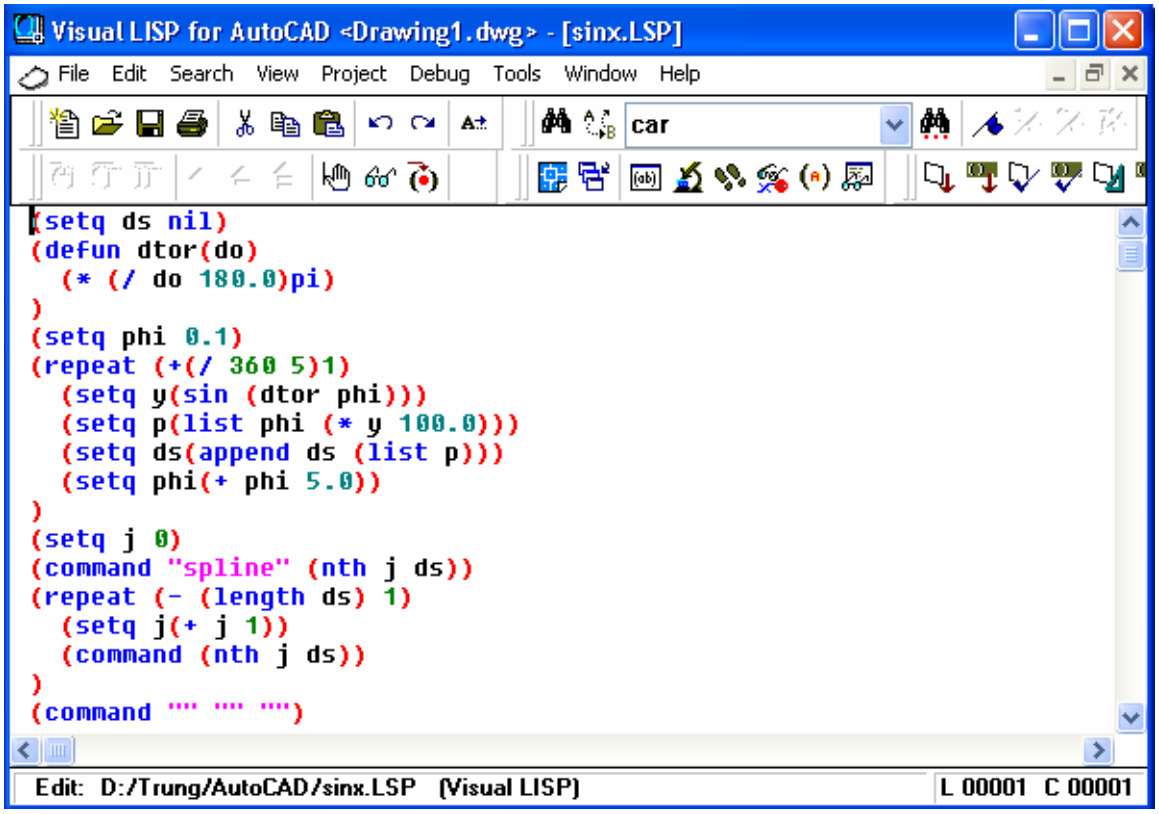
2.1.2. Tạo file chương trình.

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* .



Hình 2.1. Soạn thảo chương trình **AutoLISP** trong **Notepad.**

Nhưng công cụ tốt nhất cho việc soạn chương trìng **AutoLISP** là dùng phần mềm *Visual LISP*. Được gọi bằng lệnh Vlisp hoặc chọn từ *Tools menu -> AutoLISP->Visual LISPEditor* sẽ xuất hiện màn hình soạn thảo chương trình AutoLISP như hình dưới đây.



Hình 2.2. Soạn thảo chương trình **AutoLISP** bằng phần mềm**Visual LISP**.

√ **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.

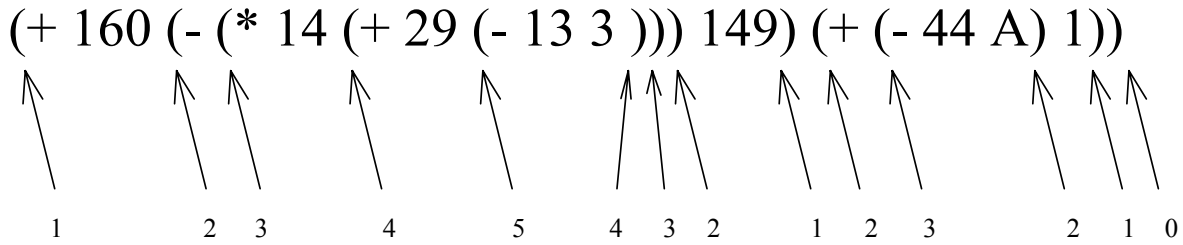
2.1.3. Các dấu ngoặc đơn.

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

Ví dụ:

```
(setq ds nil)
(defun dtor(do)
(* (/ do 180.0)pi)
)
```

Khi các biểu thức lồng nhau , các cặp dầu ngoặc mở và đóng phải đặt đúng vị trí. Có một phương pháp nhanh để kiểm tra các dấu ngoặc là đếm 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. Nếu kết quả khác không thì biểu thức ta viết đã bị lỗi. Tuy nhiên phương pháp này chỉ giúp ta 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** xem đó là tên hàm và tất nhiên là bị lỗi.

2.1.5. Các dòng chú thích.

Trong chương trình chúng 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 bên phải dấu chấm phẩy (;) cho đến hết một dòng đều được xem là chú thích. Các chú thích có thể bắt đầu ở vị trí đầu dòng hoặc đứng phía sau biểu thức.

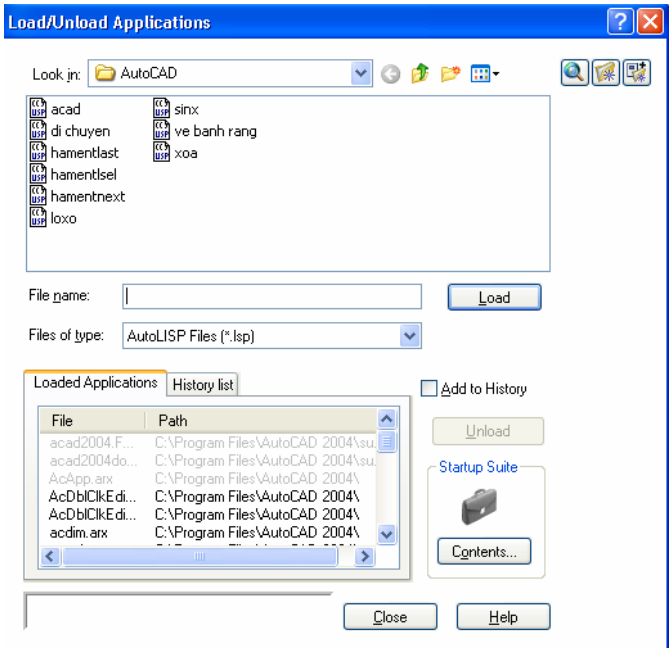
Ví dụ:

```
; chương trình tạo thủ tục chuyển số đo của góc từ độ sang radian
    (defun dtor(do)
        (* (/ do 180.0)pi) ; radian = (do/180)*pi
    ); kết thúc thủ tục
```

Các chú thích có thể đứng giữa biểu thức, bắt đầu bằng ký hiệu **;|** và kết thúc bằng ký hiệu **|;. Đ**iều này khiến chúng ta có thể tạo chú thích trên nhiều dòng liên tiếp mà không phải dùng dấu chấm phẩy trước mỗi dòng, mà chỉ cần đặt đoạn chú thích trong cặp dấu ;| và |;.

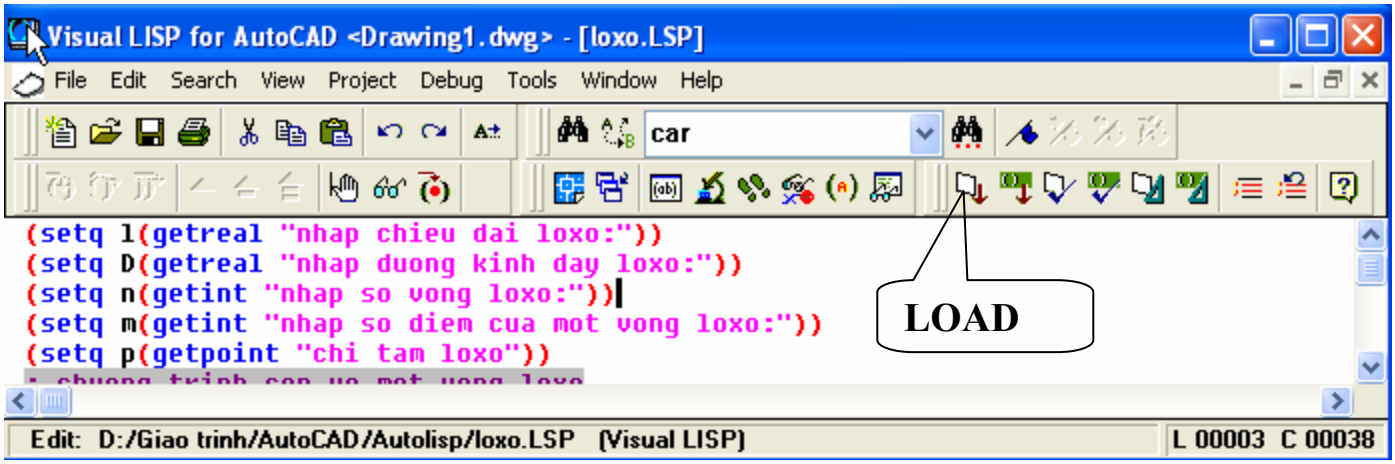
2.1.6. Gọi thực thi chương trình AutoLISP bằng lệnh Appload

Ta dùng hàm **Appload** (gõ lệnh từ dòng lênh của AutoCAD) hoặc vào **Tools** chọn **Load Aplication…**để tải một file chương trình **AutoLISP** vào **AutoCAD** để thực thi. Khi đó xuất hiện hộp thoại và ta phải chọn file sau đó chọn **Load** rồi **Close** và chương trình sẽ được thực hiện.



Hình 2.3. Hộp thoại chọn file bởi lệnh **Load.**

Hoặc đang trong chương trình Visual Lisp thì ta kích vào  **(có vị trí như hình 2.4) trên thanh công cụ để tiến hành load file hiện hành vào AutoCAD thực hiện.**



Hình 2.4. Giao diện chương trình Visual Lisp và nút lệnh LOAD

2.2. Các hàm tự tạo.

AutoLISP cho phép chúng ta tạo ra 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 chức năng như yêu cầu người sử dụng nhập các 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. Tạo hàm mới bằng hàm Defun.

+ Cú pháp:

```
(defun ten-ham (x1 x2 … / y1 y2 …)
 (bt1)
 (bt2)
 …
)
```

- Defun: hàm để tạo hàm mới của AutoLISP;

- ten-ham: do người tạo tự đặt không trùng với tên hàm đã có trong AutoLISP;

- x1, x2,…: các biến cần thiết khi gọi hàm;

- y1, y2, …:các biến cục bộ của hàm, phân biệt với biến dùng khi gọi hàm băng ký tự chia (/)

- bt1, bt2, …: các biểu thức hoặc lệnh thực thi của hàm.

Giá trị trả về của hàm là giá của biểu thức cuối cùng.

Ví dụ: tạo hàm chuyển đổi đơn vị đo của góc từ độ sang radian như sau:

```
(defun dtor(do)
        (* (/ do 180.0)pi)
)
```

Trong ví dụ này hàm **dtor** gồm 1 biến do và kết quả trả về là giá trị ứng với đơn vị radian.

2.2.2. Tạo các lệnh AutoCAD mới.

Bằng cách thêm **C:** vào trước tên hàm tự tạo thì khi thực thi chương trình AutoLISP sẽ tạo ra lệnh mới cho AutoCAD có tên trùng tên hàm. Cú pháp như sau:

```
(defun C:ten-ham ( / y1 y2 …)
 (bt1)
 (bt2)
 …
)
```

Như vậy hàm chỉ chứa biến cục bộ và có thể không cần các biến cục bộ này.

Ví dụ:

```
(defun c:hv(/ p a)
        (setq p(getpoint  "\n nhap diem goc trai phia duoi cua hinh vuong:"))
        (setq a(getreal  "\n nhap chieu dai canh hinh vuong:"))
        (command "rectangle" p (polar (polar p 0 a) (/ pi 2.0) a))
)
```

3. CÁC KIỂU DỮ LIỆU VÀ NHẬP DỮ LIỆU

3.1. Dữ liệu số nguyên (**Integer):**

Intiger (INT) là kiểu số nguyên không chứa dấu thập phân, gồm 32-bit có giá trị từ +2,147,483,647đến –2,147,483,648. Khi dùng hàm Getint để nhập một số nguyên thì hàm này chỉ chấp nhận 16-bit có giá trị từ +32767 đến -32678 (sẽ báo lỗi và chờ nhập lại). Khi dùng một số integer trong biểu thức của Autolisp giá trị này được biết như là một hằng số (Const). các số như 2, -56 và 1,200,196 là các biến số nguyên của Autolisp.

Nếu nhập vào một số mà giá trị lớn hơn giá trị max của kiểu integer thì Autolisp tự động chuyển thành kiểu số thực (Real). Nếu một phép tính được thực hiện từ hai biến kiểu Integer nhưng kết quả nằm ngoài giới hạn của kiểu Integer thì sẽ cho kết quả sai.

Các trường hợp cụ thể như sau:

Giá trị của số nguyên dương lớn nhất được chấp nhận:2147483647

Nếu bạn nhập vào một số nguyên mà có giá trị lớn hơn 2147483647 thì AutoLISP tự động chuyển thành số thực: 2147483648 chuyển thành 2.14748e+009

kết quả của phép cộng lớn hơn 2147483647 cho kết quả sai là một số nguyên âm:

_$ (+ 2147483646 3)-> -2147483647 (giá trị sai)

Tự động chuyển 2147483648 thành số thực trước khi cộng thêm 2:

_$ (+ 2147483648 2)

2.14748e+009

Giá trị của số nguyên âm nhỏ nhất được chấp nhận: -2147483647 và các lỗi tương tự như trên.

_$ -2147483648

-2.14748e+009 (trả về số thực)

_$ (- -2147483648 1)

-2.14748e+009 (kết quả sai)

Thông thường số nguyên được dùng làm giá trị cho biến đếm. Nếu không cần thiết thì nên khai báo ở dạng số thực để tranh những sai sót.

3.2. Dữ liệu số thực (Real)

Số thực là một kiểu số có chứa dấu thập phân. Những số từ -1 đến 1 phải có chữ số 0 đứng đầu. Số thực gồm phần thực và phần thập phân cách nhau bởi dấu chấm(.), phần thập phân ít nhất là 14 chữ số. Chú ý là Autolisp không hiện tất cả các con số thập phân.

Kiểu số thực có thể thể hiện dưới dạng kiểu dấu phẩy động. (for example, 0.0000041 is the same as 4.1e-6). Các số như 3.1; 0.23; –56.123; 21,000,000.0 là các số thực.

3.3. Dữ liệu kiểu chuỗi (String)

Kiểu chuỗi (String) là tập hợp các ký tự và được đặt trong dấu ngoặc kép. String là kiểu dữ liệu tạo ra giao diện giữa người và máy.

Khi sử dụng kiểu dữ liệu ta phải sử dụng ký tự kết hợp với các ký tự khác

\\	\ character (ký tự \ xuất hiện tại dòng string)
\"	" character (ký tự " xuất hiện tại dòng string)

\e	Escape character (tạo dấu cách tại vị trí ký tự)
\n	Newline character (xuống dòng mới)
\r	Return character
\t	Tab character (tạo ký tự Tab tại vị trí chèn ký tự)

Ví dụ:

Khi không dùng ký tự \ :

```
(setq a(getreal "cho biet he so a="))
(setq b(getreal "cho biet he so b="))
```

Kết quả trả về:

Command: cho biet he so a=10 *(ghi tiếp vào mà không xuống dòng)*

cho biet he so b=10 *(xuống dòng là vì nhấn Enter khi nhập 10 cho biến a)*

Khi dùng ký tự \ :

```
(setq a(getreal "\n cho biet he so a="))
(setq b(getreal "\n cho biet he so b="))
```

Kết quả trả về:

```
Command:
cho biet he so a=10
cho biet he so b=20
```

3.4. Dữ liệu kiểu file

Kiểu dữ liệu **file** là một **con trỏ** dùng để mở một file bằng hàm **open** trong AutoLISP. Hàm open trả về cho con trỏ giá trị ký tự và số của file được mở. Khi đó phải cung cấp biến như các hàm khác của AutoLISP để thực hiện đọc và ghi file được mở.

Ví dụ ta dùng chương trình sau để đọc file bac-hai.txt trong ổ đĩa C:

```
(setq f(open "c:\\bac-hai.txt" "r"))
        (while (/= (setq s(read-line f))nil)
                (if (/= s 32) (setq t1 s))
                 (princ t1)
        )
(close f)
```

Như vây ta có thể sử dụng dữ liệu từ file, và sau khi thoát phải đóng file lai bằng hàm (close)

4. DANH SÁCH VÀ XỬ LÝ DANH SÁCH

4.1. **Danh sách**

Danh sách (LIST) là tập hợp gồm các phần tử là hàm, ký tự, số và và list được chứa trong cặp dấu ngoặc đơn (). Đây là hình thức lưu trữ và xử lý dữ liệu của AutoLISP; các biểu thức, hàm,… chính là các List và phần tử đầu tiên của List là hàm.

Ví dụ:

(1 2 3 4 5) → danh sách chứa các số;

(“a” “b” “c”) → danh sách chứa các ký tự;

(“a” 1 “b” 2 “c” 3) → danh sách chứa ký tự và số

(+ 1 2 3 4) → danh sách chứa biểu thức: 1+2+3+4

4.1.1. Phân loại

Danh sách được phân thành 3 loại:

- Biểu thức (*Expression List*): chứa tên hàm và các thông số của hàm;

- Tọa độ điểm (*Point Coordinate List*): chứa tọa độ X,Y hoặc X,Y,Z của một điểm;

- Kho dữ liệu (*Data Storage List*): Chứa bất kỳ kiểu dữ liệu nào.

4.1.2. Tạo danh sách

Việc lưu trữ dữ liệu bằng danh sách sẽ đon giản đi rất nhiều so với việc lưu trữ bằng các biến. Có hai hàm để tạo danh sách,đó là:

Hàm **LIST**: (list bt1 bt2 bt3 …)

Trong đó: bt1, bt2, bt3 là các giá trị chuỗi, số, hoặc là các biểu thức đã được định giá trị.

Ví dụ: (setq p(list 10 20 30)) → (10 20 30)

(setq x(list (list 10 20) 10 “danang”)) → ((10 20) 10 “danang)

Hàm **QUOTE**: (setq a(quote (x y))) hoặc (setq a'(x y)) (X Y)→ (x y): a là list chứa hai biến x,y chưa được định giá trị.

4.2. Các hàm xử lý danh sách

4.2.1. Hàm CAR

Cú pháp: (car list)

list: là một danh sách

Kết quả trả về là phần tử đầu tiên của danh sách, nếu list rỗng thì thì kết quả nil.

Ví dụ:

Command: (car '(a b c)) → A

Command: (car '((a b) c)) →(A B)

Command: (car '()) →nil.

Thường ứng dụng lấy hoành độ x của điểm trong autoCAD.

4.2.2. Hàm CDR

Cú pháp: (cdr list)

list: là một danh sách

Kết quả trả về: Một danh sách được loại bỏ phần tử đầu. Nếu list rỗng thì trả về kết quả nil.

Ví dụ:

Command: (cdr '(a b c)) →(B C)

Command: (cdr '((10 20) 30)) →(30)

Command: (cdr '()) →nil

4.2.3. Hàm CADR

Cú pháp: (cadr list)

list: là một danh sách

Giá trị trả về: Phần tử thứ hai của danh sách (list), nếu list rỗng hoặc chỉ có một phần tử thì trả về nil.

Hàm Cadr thường dùng để lấy tọa độ y của điểm.

Ví dụ:

Command: (setq pt2 '(5.25 1.0)) →(5.25 1.0)

Command: (cadr pt2) →1.0

Command: (cadr '(4.0)) →nil

Command: (cadr '(5.25 1.0 3.0)) →1.0

4.2.4. Hàm CADDR

Cú pháp: (caddr list)

list: là một danh sách

Giá trị trả về: Phần tử thứ ba của danh sách (list), nếu list rỗng hoặc chỉ có ít hơn hai phần tử thì trả về nil.

Hàm Caddr thường dùng để lấy tọa độ z của điểm 3D.

Ví dụ:

Command: (setq pt3 '(5.25 1.0 3.0)) → (5.25 1.0 3.0)

Command: (caddr pt3) → 3.0

Command: (caddr '(5.25 1.0)) → nil

4.2.5. Hàm LAST

Cú pháp: (last list)

list: là một danh sách

Giá trị trả về: Phần tử cuối cùng của danh sách, nếu list rỗng thì trả về nil.

Ví dụ:

Command: (last '(a b c d e)) → E

Command: (last '(a b c (d e))) → (D E)

4.2.6. Hàm LENGTH

Cú pháp: (length list)

list: là một danh sách

Giá trị trả về: Một số kiểu Integer là số phần tử có trong list.

Ví dụ:

Command: (length '(a b c d)) → 4

Command: (length '(a b (c d))) → 3

Command: (length '()) → 0

4.2.7. Hàm NTH

Cú pháp: (nth n list)

n: số nguyên là số của phần tử lấy ra từ danh sách (n=0 là phần tử đầu tiên)

list: là một danh sách.

Giá trị trả về: Phần tử thứ (n +1) của list. Nếu n lớn hơn số phần tử có trong list thì trả về nil

Ví dụ:

Command: (nth 3 '(a b c d e)) → D

Command: (nth 0 '(a b c d e)) → A

Command: (nth 5 '(a b c d e)) → nil

4.2.8. Hàm APPEND

Cú pháp: (append list1 list2 ...])

list1, list2,…: các danh sách.

Giá trị trả về:

Là một danh sách gồm tất cả các phần tử của list1,list2,…

Ví dụ:

Command: (append '(a b) '(c d)) →(A B C D)

Command: (append '((a)(b)) '((c)(d))) →((A) (B) (C) (D))

5. CÁC HÀM TOÁN HỌC

5.1. **Các hàm điều khiển**

Khi tính toán trong AutoLISP thì kiểu số (real, integer) là kiểu dữ liêu cơ bản, đôi lúc ta phải chuyển đổi qua lại để đáp ứng lập trình. Ta thường sử dụng các hàm xử lý số sau để làm điều đó:

a) **Hàm ABS**: lấy trị tuyệt đối

Cú pháp: (ABS num)

Num: một số bất kỳ.

Giá trị trả về: trị tuyệt đối của num.

Ví dụ:

```
(abs 100)              → 100
(abs -100)             → 100
(abs -99.25)           → 99.25
```

b) **Hàm FIX**: Chuyển số thực thành số nguyên gần nhất.

Cú pháp: (fix number)

Number: số thực

Số nguyên tạo thành phải thuộc (+2,147,483,647 ÷ –2,147,483,648)

Ví dụ:

Command: (fix 3) →3

Command: (fix 3.7) →3

c) **Hàm FLOAT**: Chuyển số bất kỳ thành số thực

Cú pháp: (float number)

number: số bất kỳ.

Giá trị trả về: số thực từ số đã cho.

Ví dụ:

Command: (float 3) →3.0

Command: (float 3.75) →3.75

d) **Hàm REM**: chia lấy số dư.

Cú pháp: (rem num1 num2 …)

num1: số bất kỳ

num2, …: số nguyên

Nếu cung cấp nhiều hơn một số chia thì Rem chia liên tiếp từ trái sang phải, số dư sẽ được chia cho số tiếp theo và lấy số dư cuối cùng.

Giá trị trả về: số dư phụ thuộc vào số bị chia nếu số bị chia là real thi số dư là real, còn số bị chia là Integer thì số dư là integer.

A number. If any number argument is a real, rem returns a real; otherwise, rem returns an integer. If no arguments are supplied, rem returns 0. If a single number argument is supplied, rem returns number.

Ví dụ:

Command: (rem 42 12) →6

Command: (rem 12.0 16) →12.0

Command: (rem 26 7 2) →1

Command: (rem 26) →26

Command: (rem) →0

e) **Hàm GCD**: Tìm ước số chung lớn nhất của hai số nguyên.

Cú pháp: (gcd int1 int2)

int1, int2: số nguyên dương

Giá trị trả về: số nguyên là ước chung lớn nhất của int1,int2.

Ví dụ:

Command: (gcd 81 57) →3

Command: (gcd 12 20) →4

f) **Hàm MIN**: tìm số có giá trị nhỏ nhất.

Cú pháp: (min number1 number2…)

Number1, number2,…: các số bất kỳ.

Giá trị trả về: số thực (nếu có một giá trị thực), số nguyên là số có giá trị nhỏ nhất trong các giá trị.

Ví dụ:

Command: (min 683 -10.0) →-10.0

Command: (min 73 2 48 5) →2

Command: (min 73.0 2 48 5) →2.0

Command: (min 2 4 6.7) →2.0

Command: (min) →0

g)**Hàm MAX:** tương tự hàm **min** nhưng đưa ra giá trị lớn nhất.

5.2. Các hàm lượng giác, các hàm xử lý khoảng cách và góc đo

5.2.1. Các hàm lượng giác

a) **Hàm SIN**: tính sin của một góc giá trị thực đơn vị radian.

Cú pháp: (sin ang)

ang: một góc (radian)

Giá trị trả về: sin(ang)

Ví dụ:

Command: (sin 1.0) →0.841471

Command: (sin 0.0) →0.0

Command: (sin pi) →1.22461e-016

b) **Hàm COS**: tính cos của một góc giá trị thực đơn vị radian.

Cú pháp: (cos ang)

ang: một góc (radian)

Giá trị trả về: cos(ang)

Ví dụ:

Command: (cos 0.0) →1.0

Command: (cos pi) →-1.0

c) **Hàm ATAN**: Trả về giá tri arctan của một số

Cú pháp: (atan num1 [num2])

num1, num2: một số

Giá trị trả về góc (radian) có giá trị bằng: $\begin{cases} \text{arctg(num1)} & \text{num1} \neq \text{nil, num2} = \text{nil} \\ \text{arctg(num1/num2)} & \text{num1} \neq \text{nil, num2} \neq \text{nil} \\ 1.570796\ (=\pi/2) & \text{num1} \neq \text{nil, num2} = 0. \end{cases}$

Ví dụ:

Command: (atan 1) →0.785398

Command: (atan 1.0) →0.785398

Command: (atan 0.5) →0.463648

Command: (atan 1.0) →0.785398

Command: (atan -1.0) →-0.785398

Command: (atan 2.0 3.0) →0.588003

Command: (atan 2.0 -3.0) →2.55359

Command: (atan 1.0 0.0) →1.5708

Sử dụng hàm Atan ta có thể ta có thể tính được hàm asin và acos (hai hàm này không có trong autoLISP). Bằng biến đổi toán học như sau:

$$\tan x = \frac{\sin x}{\cos x} = \frac{\sqrt{1-\cos^2 x}}{\cos x}$$

Lấy atan hai vế: $x = a\tan\frac{\sqrt{1-\cos^2 x}}{\cos x}$

Đặt: t=cosx $\Rightarrow$ x =acost

Suy ra: $a\cos t = a\tan\frac{\sqrt{1-t^2}}{t}$

Tương tự: $a\sin t = a\tan\frac{t}{\sqrt{1-t^2}}$

Bài tập: Xây dựng hai hàm mới: acos và asin trong autoLISP.

5.2.2. Các hàm xử lý khoảng cách và góc đo

a) **Hàm ANGLE**: đo góc (radian) giữa đường thẳng (tạo bởi hai điểm) và trục ox.

Cú pháp: (angle pt1 pt2)

pt1, pt2: hai điểm

Giá trị trả về: giá trị góc (radian) tạo bởi đường thẳng có phương và hướng từ pt1 đến pt2 và trục ox của mặt phẳng vẽ hiện hành, có chiều dương ngược chiều kim đồng hồ. Nếu pt1 và pt2 là các điểm 3D được chiếu lên mặt phẳng vẽ hiện hành.

Ví dụ:

Command: (angle '(1.0 1.0) '(1.0 4.0)) →1.5708

Command: (angle '(5.0 1.33) '(2.4 1.33)) →3.14159

b) **Hàm DISTANCE**: Đo khoảng cách giữa hai điểm.

Cú pháp: (distance pt1 pt2)

pt1, pt2 : là điểm 2D hoặc 3D.

Giá trị trả về: giá trị khoảng cách 3D giữa hai điểm. Nếu có một hoặc cả hai là điểm 2D thì bỏ qua cao độ Z của điểm 3D còn lại và trả về khoảng cách giữa hai điểm sau khi đã chiếu chúng lên mặt phẳng vẽ hiện hành.

Ví dụ:

Command: (distance '(1.0 2.5 3.0) '(7.7 2.5 3.0)) →6.7

Command: (distance '(1.0 2.0 0.5) '(3.0 4.0 0.5)) →2.82843

c) **Hàm POLAR**: Từ một điểm dùng làm gốc tọa độ cực, cùng với một khoảng cách và một góc cho trước xác định một điểm trong không gian 3 chiều bằng phương pháp tọa độ cực.

Cú pháp: (polar pt ang dist)

pt: điểm 3D hoặc 2D.

ang: giá trị góc bằng radian được xác định tương đối so với trục x trong hệ tọa độ WCS, chiều dương ngược chiều kim đồng hồ, không phụ thuộc vào mặt phẳng vẽ hiện hành.

dist: Khoảng cách so với điểm **pt**

Giá trị trả về: là điểm 2D nếu **pt** là điểm 2D, là điểm 3D có cao độ Z với **pt** (3D)

Ví dụ:

Tạo điểm 3D bằng hàm **polar**:

Command: (polar '(1 1 3.5) 0.785398 1.414214) →(2.0 2.0 3.5)

Tạo điểm 2D bằng hàm **polar**:

Command: (polar '(1 1) 0.785398 1.414214) →(2.0 2.0)

d) **Hàm GETANGLE**: nhập giá trị góc

Cú pháp:(getangle [pt] [msg])

pt: điểm gốc trong UCS hiện hành có thể có hoặc không, nếu có thì người dùng phải nhập thêm điểm thứ hai, giá trị góc nhập vào chính là góc của đường thẳng này với trục ox của UCS hiện hành.

msg: lời nhắc.

Giá trị trả về: giá trị góc do người dùng nhập vào (có đơn vị trùng với đơn vị góc của drawing unit, thường là độ-decimal degrees) được chuyển sang radian.

Ví dụ:

Command: (setq ang (getangle)) ↵

180 ↵

3.14159 (giá trị trả về)

Command: (setq ang (getangle '(1.0 3.5))) ↵

Chọn một điểm trên màn hình

0.710481 (giá trị trả về, tùy theo điểm chọn)

Command: (setq ang (getangle "Which way? ")) ↵

Command: (setq ang (getangle "Which way? "))

Which way? 90 ↵

1.5708

Command: (setq ang (getangle '(1.0 3.5) "Which way? ")) ↵

Which way? 45 ↵

0.785398

e)**Hàm GETDIST**: Dừng lại và chờ người dùng nhập vào giá trị làm khoảng cách.Có thể nhập khoảng cách bằng cách chọn hai điểm, hoặc nhập điểm thứ hai nếu như hàm **getdist** đã định

sẵn điểm thư nhất (pt). Khoảng cách nhạp vào có đơn vị là đơn vị khoảng cách hiện hành của bản vẽ.

Cú pháp: (getdist [pt] [msg])

pt: điểm 2D hoặc 3D (có thể có hoặc không) dùng như là điểm gốc trong UCS hiện hành. Nếu hàm getdist cung cấp điểm pt thì yêu cầu nhập điểm thứ hai để xác định góc.

msg: lời nhắc

Giá trị trả về: giá trị thực. Nếu các điểm nhập vào có ít nhất một điểm 3D thì trả về khoảng cách 3D. Tuy nhiên nếu cài đặt 64 bit của hàm **initget** thì hàm **getdist** bỏ qua thành phần Z và đưa ra khoảng cách 2D.

Ví dụ:

```
(setq dist (getdist))
(setq dist (getdist '(1.0 3.5)))
(setq dist (getdist "How far "))
(setq dist (getdist '(1.0 3.5) "How far? "))
```

5.2.3. Hàm truy bắt đối tượng và truy bắt điểm

Hàm OSNAP

Hàm INTERS

5.3. Các hàm lũy thừa, khai căn , logarit

Hàm EXP

Hàm SQRT

Hàm LOG

Hàm EXPT