

THE CONTENTS

MỤC LỤC

1. The contents (Mục lục)	Trang 1
2. General View (Tổng quan về tài liệu)	Trang 2
3. The Autolisp Beginner (Autolisp Sơ cấp)	Trang 3
3.1 The first thing you should know about AutoLisp (Điều đầu tiên cần biết về Autolisp)	Trang 3
3.2 Variables (Các biến số)	Trang 4
3.3 Functions (Các hàm)	Trang 9
4. The Autolisp Immediate (Autolisp Trung cấp)	Trang 14
4.1 Data Types (Các loại dữ liệu)	Trang 15
4.2 String Functions (Các hàm xử lý chuỗi)	Trang 18
4.3 Number Functions (Các hàm xử lý số)	Trang 25
4.4 List Functions (Các hàm xử lý danh sách)	Trang 31
4.5 Entity DXF Group Codes (Các mã nhóm DXF của các đối tượng)	Trang 42
4.6 Entity DXF Group Code Table (Bảng mã nhóm DXF của các đối tượng)	Trang 44
4.7 Conversion Functions (Các hàm chuyển đổi)	Trang 50
4.8 Math Functions (Các hàm toán học)	Trang 58
4.9 Selecting Entities (Các hàm lựa chọn đối tượng)	Trang 68
4.10 Selection Set Functions (Các hàm xử lý bộ lựa chọn)	Trang 73
4.11 Entity Functions (Các hàm xử lý đối tượng)	Trang 80
4.12 Read/Write & File Functions (Các hàm xử lý đọc và viết tập tin)	Trang 86
5. The Autolisp Advanced (Autolisp cao cấp)	Trang 91
5.1 Loop Functions (Các hàm lặp)	Trang 92
5.2 Conditional Statements (Các thông báo điều kiện)	Trang 95
5.3 Entity DXF Group Code Descriptions For (Mô tả các mã nhóm DXF của một số đối tượng)	Trang 102
6. The Autolisp Extreme (Autolisp siêu cấp)	Trang 152
7. Dialog Control Language (Ngôn ngữ điều khiển hộp thoại)	Trang 155
7.1 Getting Started (Mở đầu)	Trang 156
7.2 Overview (Tổng quan)	Trang 161
7.3 The basic DCL model (Mẫu tập tin DCL cơ bản)	Trang 164
7.4 Rows and Columns (Các hàng và các cột)	Trang 167
7.5 Dialog Control Language–Controls (Các nút điều khiển trong DCL)	Trang 172
7.6 Dialog Control Language – Image (Hình ảnh trong DCL)	Trang 177
7.7 Dialog Control Language – Action (Các hành động trong DCL)	Trang 180
7.8 Dialog Control Language - Set and Mode Tile (Đặt và tạo chức năng cho phần tử trong DCL)	Trang 182
7.9 Dialog Control Language – List (Danh sách trong DCL)	Trang 184
7.10 Dialog Control Language–SaveVars(Lưu biến trong DCL)	Trang 187
7.11 Dialog Control Language - Parts (Phần thực hành về DCL)	Trang 196

The Ultimate AutoLisp Tutorial

Tài liệu tự học Autolisp tốt nhất

What the hell is a car of a cdr?

Car và crd là cái quái gì vậy?

Beginner - You know how to spell AutoLisp and that is about it. From opening notepad, writing your first program, saving your first program (this is an important step), to executing your first program and checking the variables inside AutoCAD. I will cover it all. Step by step.

Sơ cấp: - Ở đây bạn sẽ biết cấu trúc của Autolisp và mọi điều về nó. Hãy mở phần mềm Notepad viết chương trình đầu tiên của bạn, lưu lại chương trình này (đây là một bước rất quan trọng), chạy chương trình đầu tiên này và kiểm soát sự thay đổi bên trong AutoCad. Ta sẽ thực hiện tất cả những điều đó từng bước từng bước một.

Intermediate - Enough of the notepad crap. How do I actually do something? What the hell is a car of a cdr?

Trung cấp: - Mọi thứ nhỏ nhãng với notepad đã đủ. Làm thế nào để ta thực sự làm được một cái gì đó với Autolisp? Car và cdr là cái quái gì vậy?

Advanced - Enough clowning around. Get serious. Let's do something.

Cao cấp: - Đùa vậy đủ rồi. Ta phải nghiêm chỉnh thôi. Nào hãy thử làm một cái gì đó với Autolisp.

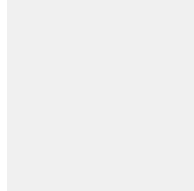
Extreme - Need more specifics? The hands on approach for the know it all. *Work in progress...*

Siêu cấp: - Phải thêm chút gì đặc biệt chứ? Hãy tiếp tục tiến tới để hiểu nó hoàn toàn. *Công việc đang tiến triển đây*

Dialog Control Language - Almost Completed! **New!**

Ngôn ngữ điều khiển hộp thoại: - Hầu hết mọi thứ đã hoàn tất! Khoái thiết!

Support this Site!



[Back to AutoLisp Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated February 26th, 2005

Copyright 2002-2007 JefferyPSanders.com. All rights reserved.

The AutoLisp Beginner

Autolisp sơ cấp

Welcome to the world of AutoLisp. Soon you will know what the hell a car of a cdr is.

Xin chào bạn đã mò mẫm với thế giới của Autolisp. Bạn sẽ biết car và cdr là cái gì mốc gì ngay ấy mà.

Let's get to it. But how?

Ta bắt đầu nhé. Nhưng bắt đầu thế nào đây?

When I say "type this at the command line" I will put the actual key strokes in red except for the enter key. This will mean that you should type the **red letters** in at the Command Prompt inside AutoCAD as if you were

typing the command "Line". To clarify, If I say "type this at the command line: **Line** and press the enter key, you would type the word "**Line**" at the command prompt inside of AutoCAD and press the enter key afterwards. Got it? No? Oh, never mind. Just keep reading. I won't leave you behind.

Khi tôi nói “đánh máy điều này vào dòng lệnh” tôi sẽ đặt các nút bàn phím hoạt động theo các ký tự màu đỏ ngoại trừ phím Enter. Điều đó có nghĩa là bạn sẽ gõ các **chữ cái màu đỏ** vào dòng nhắc lệnh trong AutoCad chẳng hạn như bạn gõ lệnh “ **Line**”. Để rõ ràng hơn, khi tôi nói “ gõ điều này vào dòng lệnh : **Line** và nhấn phím Enter, bạn gõ từ “**Line**” vào dòng nhắc lệnh trong AutoCad và sau đó nhấn tiếp phím Enter. Hiểu chưa nào? Chưa hả? Ồ không sao, chỉ cần bạn tiếp tục đọc. Tôi sẽ không bỏ rơi bạn đâu mà sợ.

The first thing you should know about AutoLisp:

Điều đầu tiên bạn cần biết về Autolisp:

AutoLisp always has as many opens "(" as it does closes ")".

Autolisp sử dụng rất nhiều dấu ngoặc mở “(“ cũng như ngoặc đóng “)”. ;(Trong mỗi chương trình số lượng ;ngoặc mở được sử dụng luôn bằng số lượng các ngoặc đóng, nếu không chương trình sẽ bị lỗi “**bad argument**” ;không chạy được

Every AutoLisp program should look similar to this:

Mỗi chương trình Autolisp luôn có dạng tương tự như chương trình dưới đây

```
(defun myProg() ;; Hàm defun xác định tên chương trình myProg không có
;biến số. Biến số nếu có sẽ được đặt trong ngoặc đơn
    (princ "Jeff") ;; Hàm princ hiển thị các ký tự Jeff ra màn hình
    (princ) ;; Hàm princ không hiển thị gì thêm và kết thúc việc thực
;hiện chương trình
) ;; Ký tự bắt buộc có biểu thị kết thúc của chương trình
```

You can execute AutoLisp at the Command Prompt inside AutoCAD. This is where we will start.

Bạn có thể thực hiện các chương trình Autolisp tại dòng nhắc lệnh trong AutoCad. Và ta sẽ bắt đầu như vậy

AutoLisp is a language that returns a value after executing. Every function that executes will return or echo it's answer. Let's test this. Open AutoCAD and type this at the command line: **(+ 1 2)** and press the enter key. (The **+** function adds numbers together). Look at the command line. You should see something that looks like this:

Autolisp là một ngôn ngữ mà nó trả về một giá trị sau khi thực hiện lệnh. Mỗi hàm được thực hiện sẽ trả về kết quả hoặc không trả. Ta sẽ kiểm chứng điều này. Hãy mở AutoCad và gõ vào dòng lệnh điều này: **(+ 1 2)** rồi

nhấn Enter. (Ký tự “+” biểu thị hàm cộng các số với nhau). Hãy xem dòng lệnh. Bạn sẽ thấy biểu thị giống như sau:



Notice the **3** below the line you typed and the Command Prompt? AutoLisp returned the value of the function you typed in. Try it again. This time change the **1** to **3** and the **2** to **5**. As in **(+ 3 5)**. What did it return? AutoLisp ALWAYS returns the answer.

Bạn có chú ý tới số **3** bên dưới dòng mà bạn đã gõ vào dòng nhắc lệnh không? Autolisp đã trả về giá trị của hàm mà bạn đã nhập. Hãy thử lần nữa nhé. Lần này ta đổi **1** thành **3** và **2** thành **5**. Và được **(+ 3 5)**. Nó trả về kết quả gì nhỉ? Vậy đó Autolisp **LUÔN LUÔN** trả về kết quả của việc thực hiện

Variables

Các biến số

What the hell is a variable? Variables are names or placeholders if you will, for values. Remember algebra? (Your going to wish you would have paid attention to Mrs. Johnson instead of that cheerleader on the first row with the blond hair and the big...well ...nevermind.) $X=1$ $Y=2$. Look familiar? X and Y are variables. They could mean anything. If you need to store data, such as a string or integer, you will need to assign a variable name to the data. The way to do this is to use the **setq** function. Ex. **(setq a 1)**. This set's the variable **a** to the value **1**. Anytime you use the variable **a** in your program, it will actually use the value **1**. If you typed **(+ a 2)** and pressed enter, AutoLisp would return **3**.

Biến số là cái gì khô gì vậy? Các biến số là các tên hay các ký tự thay thế nếu bạn muốn mà nó có giá trị. Bạn còn nhớ chút gì về đại số học không? (Này nhé bạn muốn chú ý tới một Quý Mợ Jonhson , bạn sẽ thay cái từ Cheerleader ở dòng đầu tiên bằng từ Tóc vàng và bị..... Vậy đó đừng có lo lắng nhé). $X=1$ $Y=2$. Trông có quen không? X và Y là các biến số. Chúng có thể có nghĩa là bất cứ điều gì. Nếu bạn muốn lưu trữ dữ liệu, chẳng hạn như một chuỗi hay một số, bạn sẽ phải gán một tên biến cho dữ liệu đó. Cách thực hiện điều này là sử dụng hàm **setq** . Ví dụ : **(setq a 1)**. Hàm này sẽ đặt biến **a** về giá trị **1**. Mỗi lần bạn sử dụng biến **a** trong chương trình của bạn, nó sẽ sử dụng giá trị thực là **1**. Nếu bạn nhập **(+ a 2)** và nhấn Enter Autolisp sẽ trả về giá trị **3**

You can use any variable name you want. (Practically) Upper case and lower case does not matter. I would suggest you do not use any names that AutoCAD has in use. How do you know which ones are being used by AutoCAD? Good question Jeff. The simplest method to find out if AutoCAD is using a name is to type in the name at the Command Prompt inside of AutoCAD and press the enter key. If nothing happens, use it.

Bạn có thể sử dụng bất cứ tên biến nào mà bạn khoái. Thật đấy. Chữ hoa hay chữ thường đều được. Tôi muốn lưu ý bạn rằng đừng sử dụng bất kỳ tên nào mà AutoCad đã sử dụng. Làm sao để biết một cái tên đã được AutoCad sử dụng hay chưa? Ắi chà, câu hỏi ác đấy chứ. Phương pháp đơn giản nhất để biết là bạn chỉ việc gõ cái tên đó vào dòng nhắc lệnh trong AutoCAD rồi nhấn Enter. Nếu chả có gì xảy ra cả thì cái tên đó dùng tốt.

If you were to write a program named Line.lsp and load the program, AutoCAD's built in LINE command will seem to be replaced. When you type **Line** you will run your program instead of the Line command. You will get the built in functions back next time you open a new drawing or you could use a period in front of the command to make it execute an AutoCAD built in function. Ex. **.LINE** would execute the AutoCAD built in function instead of your program. I do not suggest toying with this.

Nếu bạn viết một chương trình mang tên Line.lsp và tải nó, lệnh LINE nội trú trong AutoCad sẽ bị thay thế. Và khi bạn gõ **Line**, bạn sẽ chạy chương trình Line của bạn chứ không phải thực hiện lệnh LINE của AutoCad. Bạn sẽ chỉ có được lệnh LIND nội trú trong AutoCad khi bạn mở một bản vẽ mới ở lần kế tiếp hoặc bạn phải sử dụng dấu “.” phía trước lệnh để bắt chạy lệnh nội trú trong AutoCAD. Ví dụ: lệnh **.Line** sẽ thực hiện lệnh LINE nội trú trong AutoCad chứ không phải chạy chương trình Line của bạn đã viết. Tôi chẳng muốn bạn đùa với trò này đâu.

Name your variables and programs something unique. AutoCAD does not have any "built in" functions that are a single character. So **a** through **z** are fair game. (Unless they are assigned in the programs that automatically load during startup such as the ACAD.lsp, ACADDOC.lsp, and ACAD.PGP files. Better check your variable names at the command line as discussed earlier. Also, the letter **T** is used by autolisp as a TRUE symbol.)

Hãy đặt tên các biến và các chương trình của bạn bằng các ký tự đơn. AutoCad chẳng có bất kỳ lệnh nội trú nào mà có tên chỉ có ký tự đơn cả. Từ a đến Z chơi được tuốt. (Trừ phi chúng đã được gán trong các chương trình chạy tự động khi khởi động hằng hạn như ACAD.lsp, ACADDOC.lsp và ACAD.PGP. Tốt hơn là hãy kiểm tra các tên biến của bạn tại dòng nhắc lệnh trong AutoCAD như tôi đã bật mí cho bạn ở phần trước cho chắc ăn trước khi sử dụng. Cũng như vậy, chữ cái “ T” là đã được Autolisp sử dụng như một dấu hiệu của giá trị TRUE cho các biến logic)

[Thanks Ethan!] [On chúa lòng lành!]

Some safe examples of variable names are:

Một số mẫu an toàn (xài tốt) cho các tên biến như sau:

```
(setq a 1)
```

```
(setq b1 3)
```

```
(setq aa1 4)
```

```
(setq jeffsVarForHisProgram 1)
```

```
(setq thisVarA 2)
```

Some bad examples are:

Vài cái tên dở hơi (không xài được) như sau:

```
(setq line 1)
```

```
(setq acad 4)
```

```
(setq Itscale 7)
```

```
(setq car 12)
```

```
(setq cdr 400)
```

Again with the car and cdr stuff? What is it man?

Lại là anh quậy car và cdr ư? Nó là thằng nào vậy?

I suggest you use variable names that are somewhat descriptive. If you had to name a variable for a string that contains people's names then I would use something like this `(setq strName "Jeff")`.

Tôi mách nước cho bạn là hãy sử dụng các tên biến mà nó mô tả được chút gì đó của biến. Nếu bạn phải đặt tên cho một biến dạng chuỗi có chứa tên người, vậy bạn nên sử dụng như sau : `(setq strName "Jeff")`

If you have a variable name for a integer that holds the data for a value that represents a title block selection then I would use something like this `(setq intTBSelection 3)`.

Còn nếu bạn phải đặt tên biến cho một số mà nó mang dữ liệu về giá trị đại diện cho một lựa chọn title block , thế thì bạn nên sử dụng như sau: `(setq intTBSelection 3)`

;(À mà bạn chớ có quên rằng tên biến phải là các ký tự liền nhau, không được có khoảng trắng ở giữa chúng nhé)

A couple of important last minute things:

Vài điều quan trọng cuối cùng:

While inside AutoCAD you can check your variables to see what they are set to. This is very handy when debugging a program. Type `(setq aa1 454)` and press enter. To check the value of the variable we just declared, simply type `!aa1` at the command prompt. Go ahead. Try it. AutoLisp echo's the value to the command line. Using an exclamation point in front of the variable prints the value of the variable to the command line.

Trong khi bạn còn ở trong AutoCad bạn có thể kiểm tra các biến của bạn để xem chúng được đặt là cái gì. Đây

là điều rất cần khi kiểm duyệt từng bước một chương trình. Hãy gõ (setq aa1 454) và nhấn Enter. Để kiểm tra giá trị của biến mà ta vừa đưa ra, đơn giản là gõ !aa1 vào dòng nhắc lệnh. Nào ta hãy thử nhé. Autolisp lặp lại giá trị của biến trên dòng lệnh. Luôn nhớ phải sử dụng dấu chấm than “!” phía trước biểu thị tên biến trên dòng lệnh.

New! Have you ever seen an AutoLisp program crash? Ever try to debug it? It looks like a bunch of garbled mess. How in the heck can you cipher through all of this:

Mới đây! Bạn đã từng thấy một chương trình Autolisp gặp nạn chưa? Bạn đã cố kiểm duyệt nó từng bước chưa? Nó trông giống như một mớ bòng bong các thông báo. Bằng cách nào mà bạn có thể lần mò từng mã cho hết chúng?

```
Select objects: error: bad argument type
(+ CANT 1)
(SETQ CNT (+ CANT 1))
(WHILE (< CNT LEN) (SETQ EN (SSNAME OSET CNT)) (SETQ ENLIST (ENTGET EN)) (SETQ
TXT (CDR (ASSOC 1 ENLIST))) (SETQ NWTXT (+ (ATOF TXT) NMADD)) (SETQ NWTXT (RTOS
NWTXT 2 RNDDEC)) (SETQ ENLIST (SUBST (CONS 1 NWTXT) (ASSOC 1 ENLIST) ENLIST))
(ENTMOD ENLIST) (SETQ CNT (+ CANT 1)))
(PROGN (WHILE (< CNT LEN) (SETQ EN (SSNAME OSET CNT)) (SETQ ENLIST (ENTGET EN))
(SETQ TXT (CDR (ASSOC 1 ENLIST))) (SETQ NWTXT (+ (ATOF TXT) NMADD)) (SETQ NWTXT
(RTOS NWTXT 2 RNDDEC)) (SETQ ENLIST (SUBST (CONS 1 NWTXT) (ASSOC 1 ENLIST)
ENLIST)) (ENTMOD ENLIST) (SETQ CNT (+ CANT 1))))
(IF (/= LEN nil) (PROGN (WHILE (< CNT LEN) (SETQ EN (SSNAME OSET CNT)) (SETQ
ENLIST (ENTGET EN)) (SETQ TXT (CDR (ASSOC 1 ENLIST))) (SETQ NWTXT (+ (ATOF TXT)
NMADD)) (SETQ NWTXT (RTOS NWTXT 2 RNDDEC)) (SETQ ENLIST (SUBST (CONS 1 NWTXT)
(ASSOC 1 ENLIST) ENLIST)) (ENTMOD ENLIST) (SETQ CNT (+ CANT 1))))))
(C:ADD)
*Cancel*

Command: |
```

It's not as tough as it looks. You only have to worry about the first or second lines returned. In this case the culprit that crashed the program is (+ CANT 1). Let's look at the variable CANT. At the command prompt type !CANT<enter> It will return nil. You can't add nil to 1. Why is CANT set to nil? Let's look at the next line. (SETQ CNT(+ CANT 1)) This looks like a counter of some type. I'll bet the variable name CANT should have been CNT. A typo! At this point I would open the program with NotePad and search for the text string CANT. If this is the only place it shows up then my first explanation would be correct. I would retype the CANT variable to be CNT. Save the program and try it again.

Ồ, nó không rắc rối như bạn tưởng đâu. Bạn chỉ cần chú ý tới kết quả của các dòng thứ nhất hoặc thứ hai mà thôi. Trong trường hợp này, thủ phạm gây họa cho chương trình là (+ CANT 1). Ta hãy xem biến CANT. Tại dòng nhắc lệnh, ta nhập !CANT rồi Enter. Nó sẽ trả kết quả nil. Bạn không thể cộng nil với 1. Vậy tại sao biến CANT lại bị đặt thành nil? Ta hãy xử dòng tiếp theo. (SETQ CNT (+ CANT 1)). Điều này trông giống như một việc đếm cái gì đó. Vì thế tôi đoán chừng là tên biến CANT phải là CNT. Tại đánh máy chắc!.Bây giờ tôi mở Notepad và tìm kiếm chuỗi lý tự CANT. Nếu chỉ có duy nhất một chỗ chứa chuỗi này vậy thì lời giải thích ban đầu của tôi là đúng. Tôi sửa lại tên biến CANT thành CNT. Lưu lại chương trình và chạy thử lại.

Okay, enough for variables for now. Let's move on.

Ô kê xa lem, xem xem giống Autolisp chưa? Vậy là đủ với các biến rồi. Tiếp tục chứ. ; (Một lần nữa nhắc ;lại là tên biến không được chứa khoảng trắng, bằng không thì nó chẳng giống Autolisp đâu, nó giống Ổn Tà ;Roan)

Functions

Các hàm ;(Hàm là tôi gọi bậy, còn là tùy bạn gọi nhé, Tí hay Tèo thì nó vẫn là Function thôi)

Let's take it a step further and create our very own function. Every defined function begins with the declaration **defun**.

Mở thêm tí nữa và ta sẽ tạo ra chương trình rất riêng của mình. Mỗi chương trình xác định đều bắt đầu bằng một tuyên bố rất Autolisp **defun** . ;(Nhớ là phải tuyên bố sau khi đã mở ngoặc nhé, nếu không Autolisp bảo là ;láo, không thừa nhận đâu. Ở đây tôi gọi function là chương trình nghe cho có vẻ hoành tráng chứ nó cũng là ;function thôi)

Ex. `(defun myProg())` or `(defun C:myProg())`

Ví dụ: `(defun myProg())` hay `(defun C:myProg())` ;; Chỗ này chưa đóng ngoặc vì ta còn phải tuyên bố các nội hàm (internal function) trong chương trình của ta đã chứ , xong rồi đóng một thể.

The difference between using the C: and not using the C: will be explained later. For now. let's learn another AutoLisp function to use in our first program. The function **princ** simply prints to the command line. Ex. `(princ "Jeff")` would print **Jeff** on the command line. Alrighty then, let's type this at the command prompt:

Sự khác nhau giữa việc sử dụng và không sử dụng ký tự “ c: “ sẽ được giải thích sau. Còn bây giờ ta sẽ học một hàm Autolisp khác để sử dụng trong chương trình đầu tiên của ta. Hàm **princ** đơn giản là in ra dòng lệnh. Ví dụ; `(princ “Jeff”)` sẽ in ra các ký tự **Jeff** trên dòng lệnh. Mọi sự hoàn hảo rồi, ta nhập điều này vào dòng nhắc lệnh rồi Enter

`(defun myProg()) (princ "Jeff")` and press enter.

;(Lúc này ta phải nhớ đóng ngoặc lại kết thúc chương trình vì đã hết trò cần tuyên bố)

You should see something that looks like this:

Bạn sẽ thấy vài thứ giống như sau:

```
Command: (defun myProg()(princ "Jeff"))
MYPROG
Command:
Object Properties toolbar
```

Now type **(myprog)** at the command prompt and press the enter key. You should see something that looks like this:

Bây giờ, nhập **(myProg)** vào dòng nhắc lệnh rồi nhấn Enter. Bạn sẽ thấy như sau:

```
Command: (myprog)
Jeff"Jeff"
Command: |
1'-4 1/8", 0'-0 15/16", 0'-0" SNAP GRID ORTHO OSNAP MODEL TILE
```

The program executed and printed "Jeff" on the command line. Then the program echoed the last statement to the command line. AutoLisp always returns the answer right? That is why you ended up with **Jeff"Jeff"** on the command line. The way to get rid of the echo is to use a **princ** statement without any parameters as the last statement in your program. Ex. **(princ)**.

Chương trình đã thực hiện và in “Jeff” lên dòng lệnh. Sau đó chương trình lặp lại thông báo cuối cùng trên dòng lệnh. Autolisp là luôn trả về kết quả, đúng không? Đó chính là lý do bạn đã kết thúc với **Jeff"Jeff"** trên dòng lệnh. Cách để loại bỏ sự lặp lại là sử dụng một thông báo **princ** không có bất kỳ thông số nào như là một thông báo cuối cùng trong chương trình của bạn. Ví dụ: **(princ)**

Let's rewrite the program using the **(princ)** function to stop the echo. Type this is in at the command prompt:

Hãy viết lại chương trình sử dụng hàm **(princ)** để dừng sự lặp lại. Nhập điều sau vào dòng nhắc lệnh:

```
(defun myProg() (princ "Jeff") (princ))
```

Then type **(myprog)** and press enter. What happened? No echo. Cool.

Rồi nhập **(myProg)** và nhấn Enter. Điều gì xảy ra thế? Không có sự lặp lại nữa. Chà khoái ghê.

Let's back track and explain the C: we mentioned earlier. The C: tells AutoCAD that you want this program to be executed at the Command Prompt like a built in function. Let's redo our program. Type this at the command line:

Ta hãy quay lại và giải thích về ký tự “C:” đã được đề cập ở trên. Ký tự “c:” nói với AutoCad rằng bạn muốn chương trình này được thực hiện tại dòng nhắc lệnh giống như một hàm nội trú trong AutoCad. Ta hãy

làm lại chương trình của chúng ta. Nhập dòng sau vào dòng lệnh:

```
(defun C:myProg() (princ "Sanders") (princ))
```

Now type **(myprog)** and press enter. What happened? Why did it print Jeff? We used the C: in front of the function. We do not have to put the program inside of quotes to execute it. (There are some other differences, we will get to that in the advanced levels.) To execute a function that was declared with a C: you only have to type the name of the program at the command prompt. Let's do that now. Type **myProg** at the command prompt and press enter. It printed "**Sanders**" to the command line. Wow! Don't rush off to show everyone yet. The next time you open a drawing your new function will disappear and you will have to type it in again. Dang!

Bây giờ nhập **(myProg)** và nhấn Enter. Điều gì xảy ra vậy? Sao lại in ra Jeff? Ta đã sử dụng ký tự "c:" phía trước tên chương trình. Ta không cần phải đặt tên chương trình vào trong ngoặc để thực hiện nó. (Còn những điều khác nữa, ta sẽ biết được trong phần Autolisp cao cấp). Để thực hiện một chương trình mà nó đã được tuyên bố với ký tự "c:" bạn chỉ cần nhập tên của chương trình vào dòng nhắc lệnh. Ta sẽ thực hiện điều đó ngay bây giờ. Nhập **myProg** vào dòng nhắc lệnh và nhấn Enter. Nó sẽ in "**Sanders**" trên dòng lệnh. Wow! Đừng vội khoe với mọi người. Lần sau khi bạn mở một bản vẽ, chương trình mới của bạn sẽ biến mất và bạn sẽ phải gõ lại chúng lần nữa. Chết tiệt thật

Wait! We could save it to disk and reload it whenever we needed it. Let's do that. I will assume you have a windows operating system for the following instructions. Yes. I know what happens when you assume something. Let's not go there.

Ày, đừng nóng. Ta sẽ lưu lại chương trình vào đĩa và tải lại nó mỗi khi ta cần. Hãy làm thế nhé. Tôi sẽ giả sử rằng bạn có một hệ thống các cửa sổ đang hoạt động để thực hiện các chỉ dẫn sau đây. Đúng vậy. Tôi biết điều gì sẽ xảy ra khi bạn giả sử cái gì đó. Ta sẽ không làm điều đó ở đây.

Click on the **Start** button. Go to **Programs**. Go to **Accessories**. Go to **NotePad**.

Click nút **Start**. Vào **Programs**. Vào **Accessories**. Vào **NotePad**

Notepad should open. When it does, type in:

NotePad sẽ mở ra và bạn hãy nhập vào đó những dòng sau:

```
(defun C:myProg()  
  (princ "Jeff")  
  (princ "Sanders")  
  (princ "AutoLisp")  
  (princ)  
)
```

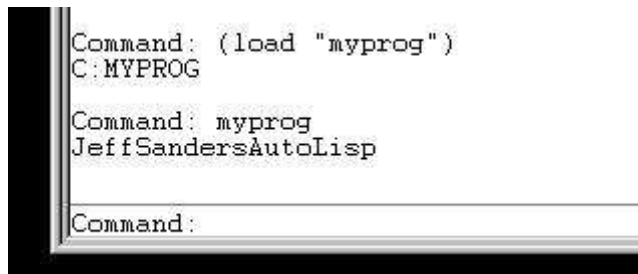
Now, go to the "**File**" drop down menu in NotePad and select "SaveAs". Type in **myProg.lsp** for the name of the program. Wait! Don't hit the SAVE button yet. NotePad has a bad habit of assuming you want to save it as a text file with the extension txt. Go to the "SAVE AS TYPE" drop down box and select "ALL FILES".

Look at the top of NotePad in the "Save in:" location. Make sure you save the file in a location where AutoCAD can find your file in it's search path. I suggest creating a directory on your hard drive to keep all of your AutoLisp files in. I would suggest a directory such as : "C:\ACAD\lsp". Change the "Save in: " location to point to this directory. Now press the SAVE button. Go back to AutoCAD. Go to the "Tools" drop down menu and go to "Preferences". Click the "Files" tab. Click the + sign in front of "Support File Search Path". Click the "ADD" button. Type in the directory you saved the "myProg" lisp file in. Ex. "C:\ACAD\lsp" . Click the "APPLY" button. Click OK to the alert box. Click the OK button to exit the dialog box.

Bây giờ từ danh mục các lệnh mở rộng của “File” menu trong NotePad chọn Save as. Nhập **myProg.lsp** làm tên của chương trình. Ấy! Họ đã! Đừng vội click nút Save. NotePad có thói quen dở hơi là cứ coi rằng tất cả mọi thứ bạn đều muốn lưu lại như một file text với phần mở rộng là txt. Vậy nên bạn phải vào hộp thoại “Save as type” mở ra và chọn mục “All Files”. Ngó lên trên đầu hộp thoại “Save as” của NotePad, hãy nhập vào hộp thoại “Save in” địa chỉ mà bạn muốn lưu chương trình. Phải chắc chắn là địa chỉ này có trong đường dẫn tìm kiếm hỗ trợ của AutoCad. Tôi mách nước cho bạn là nên tạo một thư mục trên ổ đĩa cứng để lưu toàn bộ các file chương trình Autolisp vào đó, chẳng hạn như: “C:\ACAD\lsp”. Bây giờ thì nhấn nút “Save” được rồi. ;(Nhấn thoải mái, bao lâu cũng được nhưng một lần thôi). Trở lại với AutoCad. Mở menu “Tools” và tìm tới tab “Preferences”, click tab “Files“, mở rộng nút “Support Files Search Path”, click tab “Add”, nhập vào đường dẫn tới thư mục chứa file chương trình Autolisp của bạn. Chẳng hạn là “c:\ACAD\lsp”. Click nút “Apply”. Click nút “OK” của hộp thoại cảnh báo. Click nút “OK” để thoát khỏi hộp thoại

At the command prompt inside AutoCAD type (**load "myProg"**) and press enter. Then type **myprog** and press enter. If you hit the F2 button to bring up the text screen, you should see something that looks like this:

Tại dòng nhắc lệnh trong AutoCad nhập (**load myProg**) và nhấn Enter. Sau đó nhập **myprog** và nhấn Enter. Nhấn phím F2 để kéo màn hình lên phía trên bạn sẽ thấy nó hiển thị như sau:



```
Command: (load "myprog")
C:MYPROG

Command: myprog
JeffSandersAutoLisp

Command:
```

You have successfully created an AutoLisp program. Congratulations!

Bạn đã thành công trong việc tạo một chương trình Autolisp cho riêng bạn. Quá giỏi rồi, chúc mừng nhé, khao đi.

One last thing about functions:

Điều cuối cùng về các hàm:

We discussed the **defun()** statement and what it meant. We did not discuss the open and close parenthesis **"()**" after the defun statement. We will do that in the intermediate tutorial level. The point I wanted to get to here is, if you put your variable names inside these parenthesis and after a / you will reset these variables to nothing...or nil as AutoLisp defines it. Therefore the exclamation point before the variable name will return nil. This saves room in RAM and also keeps your program from getting corrupted by other programs that do not reset the variables to nil. Always include the variables you use inside the parenthesis **after** debugging your program.

Chúng ta đã bàn về cái tuyên bố defun() và nghĩa của nó. Nhưng vẫn chưa sờ mó gì tới mấy cái ngoặc đóng mở "()" ở sau tuyên bố này. Điều này sẽ được đề cập tới trong phần Autolisp trung cấp. Điều tôi muốn nói ở đây là nếu bạn đặt các tên biến của bạn trong mấy cái ngực này và sau ký tự "/" thì bạn sẽ đặt lại các biến này về nil hoặc chả là gì cả khi Autolisp xác định chúng. Do vậy dấu chấm than trước tên biến sẽ trả về nil. Điều này sẽ giúp tiết kiệm không gian cho bộ nhớ RAM và cũng giữ cho chương trình của bạn không bị đột quỵ bởi các chương trình khác mà nó không đặt lại biến về nil. Luôn luôn gộp các biến của bạn vào trong ngoặc sau khi chạy kiểm duyệt chương trình của bạn từng bước một.

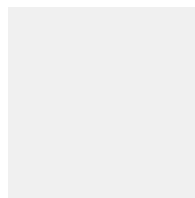
An example of this is (defun C:myProg(/ varA varB varC).

Một ví dụ về điều này là (defun C: myprog (/ varA varB varC)

You are so, so, close to uncovering the mysteries surrounding the car and cdr quandaries. Move on to the next level.

Vậy là bạn cũng được đẩy chớ, sắp túm được bí mật xung quanh mấy thằng khó chơi car và cdr rồi. Leo tiếp thôi.

Support this Site!



[AutoLisp Tutorial Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated May 24th, 2006

Copyright 2002-2007 JefferyPSanders.com. All rights reserved.

The AutoLisp Intermediate Tutorial

Tự học Autolisp Trung cấp

Welcome to the Intermediate Tutorial. You know the beginners steps to writing an AutoLisp program. You don't? See the [Beginner's Tutorial](#). Now all you need to know is what to write. What functions are available? How do you manipulate entities inside of AutoCAD? What the heck is a car of a cdr?

Chào bạn đã mò tới phần Autolisp Trung cấp. Hẳn bạn đã biết các bước để viết một chương trình Autolisp. Bạn chưa biết ư? Hãy xem phần Autolisp Sơ cấp. Giờ đây điều bạn cần biết là viết cái gì. Các chương trình có khả năng làm gì? Làm sao để vận dụng các đối tượng trong AutoCad? Mấy cái anh quậ car và cdr là cái cóc khô gì?

I believe that you not only need to know what a function returns if it is successful, but also what the function will return if it fails. This will help you troubleshoot your programs. That is why I believe this will become the Ultimate AutoLisp Tutorial. In the examples of the functions below, I've included the

most common reasons why AutoLisp programs crash along with the error message you will receive. Remember, when AutoLisp has an error, execution is aborted.

Tôi tin rằng bạn không chỉ cần biết các kết quả trả về khi các hàm hoạt động thành công mà còn cả các kết quả trả về khi các hàm thất bại. Điều đó sẽ giúp bạn xử lý khắc phục các chương trình của bạn. Vậy nên tôi mới tin rằng đây là cuốn tài liệu tự học Autolisp tốt nhất. Trong các ví dụ về các hàm dưới đây, tôi sẽ bao gồm hầu hết các lý do làm cho các chương trình Autolisp bị đột quy cùng với các thông báo lỗi mà bạn sẽ gặp phải. Cần nhớ rằng khi Autolisp có lỗi, việc thực hiện chương trình sẽ ngừng lại và thoát ra.

Let's begin with Data Types:

Ta sẽ bắt đầu với các loại dữ liệu:

Data Types:

Các loại dữ liệu:

String

Chuỗi

A string is an AutoLisp data type consisting of alphanumeric characters inside quotes.

Chuỗi là một loại dữ liệu của Autolisp bao gồm các ký tự chữ cái và số đặt trong dấu ngoặc kép “ “

eg. "Jeff" "Sanders" "1245r" "3"

Ví dụ: “Jeff” Sanders” 1245r” “3”

Integers

Số nguyên

An integer is a number without a decimal place.

Số nguyên là một loại dữ liệu Autolisp dạng số không có phần thập phân .

eg. 1 23 456 67890 -5 -687

Ví dụ: 1 23 456 67890 -5 -687

Real

Số thực

A real number is a number that has a decimal place.

Số thực là một loại dữ liệu Autolisp dạng số có chứa phần thập phân

eg. 1.0 23.034 456.2 -56.345

Ví dụ: 1.0 23.034 456.2 -56.345

List

Danh sách

A list is an AutoLisp data type that has parenthesis around it. Multiple data types can reside inside one list. This will also unravel the mystery behind the **car** and the **cdr**.

Danh sách là một loại dữ liệu Autolisp được đặt trong dấu ngoặc đơn. Có thể có nhiều loại dữ liệu ở trong một danh sách (list). Điều đó cũng mở ra bí ẩn đằng sau mấy chú car và cdr .

eg. (1 2 3) ("Jeff" 156.0 45.0 3 "Sanders") (1) () ((1 2 3) (4 5 6) ("A" "B" "C"))

Ví dụ: (1 2 3) ("Jeff" 156.0 45.0 3 "Sanders") (1) () ((1 2 3) (4 5 6) ("A" "B" "C"))

Variables or Symbols

Các biến hay các ký hiệu biến

We covered these in the Beginner's Tutorial. A variable is a symbol that is created by you, the user. eg. (setq a 1) (setq pt1(getpoint "\n Select Point: ")) - **a** and **pt1** are the variables.

Ta đã biết những điều này trong phần Autolisp sơ cấp. Một biến là một ký hiệu do bạn, người sử dụng biến tạo ra. Chẳng hạn như: (setq a 1) (setq pt1(getpoint "\n Select Point: ")) – **a** và **pt1** là các biến

Associated List

Danh sách tương tác

An associated list is a list that has a value associated with it. The association and the data are separated by a period. eg. (1 . 455) (34 . "Sanders") You can find "Sanders" by looking up group code 34. We will get more into this when we take a look at the DXF Group Codes later in this Tutorial.

Danh sách tương tác là một danh sách có một giá trị tương tác với danh sách đó. Sự tương tác và dữ liệu

tương tác được tách rời nhau bởi dấu “.”. Ví dụ: (1 . 455) (34 . "Sanders") . Bạn có thể tìm thấy chuỗi “Sanders” bằng cách tìm kiếm mã nhóm 34. Ta sẽ đi sâu hơn vào vấn đề này khi học tới phần Các mã nhóm DXF trong tài liệu này.

Let's get started with the basic functions. Remember, you don't need to know all of this. You don't need to memorize all of this. This is a reference. There will be no test. It will help you to read through it once to see what functions are available. When you are writing a program and get stuck, come back here to find the correct function to solve your problem. Put down your pencil. Really.... no test.

Ta sẽ bắt đầu với các hàm cơ bản. Nhớ rằng, bạn không cần phải biết hết về nó. Bạn cũng chẳng cần nhớ mọi thứ về nó làm gì. Nó chỉ để tham khảo mà thôi. Cũng chẳng cần thử cho mất thời gian. Bạn chỉ cần đọc qua nó một lần để biết có những hàm gì mà thôi. Khi nào viết chương trình và gặp phiền toái, hãy quay trở lại đây và chọn lấy hàm nào có thể giải quyết được vấn đề của bạn. Quảng bút đi, khỏi cần thử mà.

String Functions substr strlen strcase strcat

Number Functions abs atof atoi fix float itoa

List Functions car cdr cadr caddr caar cddr foreach list cons nth

Group Codes/Associated List assoc Entity_DXF_Group_Codes

Conversion Functions fix float itoa atoi atof rtos angtos

Math Functions + - * / +1 -1 cos atan sin sqrt expt

Selecting Entities entsel ssget

Selection Set Functions ssadd ssdel sslength ssname

Entity Functions entget entlast entnext entdel entmod entupd

File Functions open close read-line write-line **New!**

DXF Group Code Table

String Functions

Các hàm xử lý chuỗi

Functions - [substr](#) [strcase](#) [strlen](#) [strcat](#) [Example Programs](#)

substr - This function retrieves part of a string.

Hàm này xử lý việc cắt bớt một **chuỗi**

Syntax : (**substr** "string" startPoint numberOfCharacters)

Cú pháp: (**substr** “chuỗi” Vịtríbắtđầu Sốkýtự)

"string" - any valid string or variable representing a string. If you use a variable name omit the quotes.

“chuỗi” - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

startPoint - an integer or variable representing an integer that corresponds to the position in the string to start the substring at. The first character is character 1.

Vịtríbắtđầu - một **số** hay một biến đại diện cho số mà nó tương ứng với vị trí trong chuỗi để bắt đầu xác định chuỗi kết quả. Vị trí ký tự đầu tiên của chuỗi là 1

numberOfCharacters - an integer or variable representing an integer that corresponds to the length of string to return. Length being the number of characters.

Sốkýtự - một số hay một biến đại diện cho số mà nó tương ứng với độ dài của chuỗi kết quả. Độ dài của chuỗi là số ký tự có trong chuỗi.

Returns a partial string starting at the character represented by startPoint and ending at the character represented by the numberOfCharacters or the end of the string. Whichever comes first..

Giá trị trả về của hàm là một **chuỗi** thành phần bắt đầu từ ký tự tại **Vịtríbắtđầu** với độ dài của chuỗi bằng **Sốkýtự** hoặc kết thúc với ký tự cuối cùng của chuỗi ban đầu tùy theo giá trị của **Sốkýtự**

(**substr** "Jeff Sanders" 1 6) would return "Jeff S" (trả về giá trị “Jeff S”)

`(substr "Jeff Sanders" 6 3)` would return (trả về giá trị) "San"

`(substr "Jeff Sanders" 64 456)` would return (trả về giá trị) ""

`(substr "Jeff Sanders" -4 3)` would return (trả về giá trị) "Error: Bad argument"
[No negative character positions] (Không có số vị trí âm)

`(substr "Jeff Sanders" 4 -3)` would return (trả về giá trị) "Error: Bad argument"
[No negative string lengths] (Chuỗi không có độ dài âm)

`(substr "Jeff" 6 4)` would return (trả về giá trị) ""

`(substr "" 9 3)` would return (trả về giá trị) ""

`(substr "Jeff Sanders" 9)` would return (trả về giá trị) "ders"

If you omit the last parameter, the remainder of the string from the startPoint will be returned..

Nếu bạn bỏ qua thông số cuối cùng **Số ký tự**, hàm sẽ trả về chuỗi còn lại từ **Vị trí bắt đầu**

strcase - This function converts to upper or lower case.

Hàm này đổi toàn bộ **chuỗi** thành các ký tự in hoa hoặc ký tự in thường

Syntax : `(strcase "string" flag)`

Cú pháp: `(strcase “chuỗi” Giá trị Logic)`

"string" - any valid string or variable representing a string. If you use a variable omit the quotes.

“chuỗi” - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

flag - a flag that sets uppercase or lowercase.

Giá trị Logic - một ký hiệu để thực hiện việc đặt chuỗi thành các ký tự in hoa hoặc in thường

(Giá trị Logic) flag = T or nil

T = True nil = False

T = LowerCase (**Ký tự in thường**) nil or empty = UpperCase (**ký tự in hoa**)

Returns a string that is either upper or lower case depending on the flag setting.

Giá trị trả về của hàm là chuỗi với toàn bộ các ký tự in thường hoặc in hoa tùy theo **Giá trị Logic**

`(strcase "Jeff" T)` returns (trả về giá trị) "jeff"

`(strcase "Jeff" nil)` returns (trả về giá trị) "JEFF"

`(strcase "123a" nil)` returns (trả về giá trị) "123A"

`(strcase "" nil)` returns (trả về giá trị) ""

`(strcase 123 T)` returns (trả về giá trị) "Error: Bad Argument Type" [123 is an integer, not a string] (123 là một số nguyên chứ không phải chuỗi)

`(strcase 1.0 nil)` returns (trả về giá trị) "Error: Bad Argument Type" [1.0 is a real number, not a string] (1.0 là một số thực chứ không phải chuỗi)

`(strcase "Jeff")` returns (trả về giá trị) "JEFF"

Omitting the flag is the same as having the flag set to nil or false.

Bỏ qua **Giá trị Logic** là tương tự việc đặt **Giá trị Logic** về giá trị nil hay false

strlen - This function returns the length of a string in characters.

Hàm này trả về giá trị **số** là độ dài của chuỗi hay số lượng các ký tự của chuỗi

Syntax : `(strlen "string")`

Cú pháp: `(strlen “chuỗi”)`

"string" - any valid string or variable representing a string. If you use a variable omit the quotes.

“chuỗi” - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

Returns an integer that represents the length of the string. The length is the number of characters.

Giá trị trả về của hàm là một số đại diện cho độ dài của chuỗi tức số lượng các ký tự của chuỗi

`(strlen "Jeff")` returns (trả về giá trị) 4

`(strlen "Jeff Sanders")` returns (trả về giá trị) 12

`(strlen "")` returns (trả về giá trị) 0

`(strlen 123)` returns (trả về giá trị) "Error: Bad Argument Type [123 is an integer, not a string] (123 là một số nguyên chứ không phải là một chuỗi)

`(strlen 1.0)` returns (trả về giá trị) "Error: Bad Argument Type [1.0 is a real number, not a string] (1.0 là một số thực chứ không phải một chuỗi)

`(strlen (list 1.0 2.0))` returns (trả về giá trị) "Error: Bad Argument Type [(list 1.0 2.0) is a list, not a string] ((list 1.0 2.0) là một danh sách chứ không phải chuỗi)

strcat - This function adds multiple strings together to form one string.

Hàm này cộng các **chuỗi** đã có thành một **chuỗi** mới

Syntax : `(strcat "string1" "string2" "string3"ect.)`

Cú pháp: `(strcat “chuỗi1” “chuỗi2” “chuỗi3”)`

"string1" - any valid string or variable representing a string. If you use a variable omit the quotes.

“chuỗi1”, “chuỗi2”, “chuỗi3” - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

Returns a string that includes all of the strings in the list.

Giá trị trả về của hàm là một **chuỗi** bao gồm tất cả các chuỗi trong hàm theo thứ tự của nó.

`(strcat "Jeff" "Sanders")` returns (trả về giá trị) "JeffSanders"

`(strcat "Jeff " "Sanders")` returns (trả về giá trị) "Jeff Sanders"

`(strcat "" "56" "Jeff" "abcdefg")` returns (trả về giá trị) "56Jeffabcdefg"

`(strcat "" 5 "Jeff")` returns (trả về giá trị) "Error: Bad argument type". [5 is an integer, not a string] (5 là số nguyên chứ không phải chuỗi)

`(strcat "" 3.1 "Jeff")` returns (trả về giá trị) "Error: Bad argument type". [3.1 is a

real number, not a string] (3.1 là số thực chứ không phải chuỗi)

This function can only accept strings or a variable representing a string as its arguments. (parameters)

Hàm này **chỉ nhận** các đối số (tham số) là chuỗi hoặc biến đại diện cho chuỗi

End of String Functions

Kết thúc các hàm xử lý chuỗi

Example Program 1:

Chương trình mẫu 1:

```
(defun C:myProg() ; Đặt tên chương trình là myProg không có biến số

  (setq str1 "Jeffery"); Đặt biến str1 là chuỗi Jeffery

  (setq str2 "P"); Đặt biến str2 là chuỗi P

  (setq str4(substr str1 1 4)) ;sets str4 to 1st 4 characters of str1
                               (Đặt biến str4 là chuỗi gồm 4 ký tự đầu của biến str1)

  (princ str4)                ;Prints "Jeff" to the command line.
                               (Hiển thị giá trị biến str4 ("Jeff") ra dòng lệnh của AutoCad)

  (setq str4(strcase str4))    ;Sets str4 to uppercase (Đặt lại biến
                               str4 thành chuỗi các ký tự in hoa)

  (princ str4)                ;Prints "JEFF" to the command line.
                               (Hiển thị giá trị biến str4 ("JEFF") ra dòng lệnh của AutoCad)

  (setq int1(strlen str4))     ;Sets int1 to 4 (length of string) (Đặt
                               biến int1 là độ dài chuỗi của biến str4 (4))

  (princ int1)                ;Prints 4 to the command line. (Hiển
                               thị giá trị biến int1 (4) ra dòng lệnh của AutoCad)

  (setq str5 (strcat str4 " " str2)) ;sets str5 to "JEFF P" (Đặt biến
  str5 là chuỗi tổ hợp của biến str4 chuỗi " " và biến str2 ("JEFF P"))

  (princ str5)                ;Prints "JEFF P" to the command line.
                               (Hiển thị giá trị biến str5 ("JEFF P") ra dòng lệnh của AutoCad)
```

```
(princ) ;clean exit (supresses echo) (Thoát êm  
kết thúc việc thực hiện chương trình (loại trừ việc lặp lại))  
) ;close the program (Đóng chương trình)
```

(Nhập vào dòng nhắc lệnh của AutoCad sau khi đã lưu chương trình vào thư mục trên đĩa cứng như sau:)

Command: (load "myProg")<enter>

Command: myProg<enter>

(Xem hiển thị trong dòng lệnh của AutoCad)

Command: JeffJEFF4JEFF P

Program Example 2:

Chương trình mẫu 2:

```
(defun C:NameMix() ; Đặt tên chương trình là NameMix không có biến số  
  
(setq strName(getstring T "\n Enter your name: ")) ; Đặt biến strName  
;là kết quả trả về của hàm getstring với tham số T và chuỗi  
;hiển thị là Enter your name (Ở đây thú thật là tôi chả  
;quen biết gì cái chú getstring này, cứ nói đại như vậy cho  
;oai thôi, ai bà con với chú ta thì giới thiệu giùm nhé)  
  
(setq keepGoing 1); Đặt biến keepGoing bằng 1  
  
(while (<= keepGoing (strlen strName)); (Hì hì, cái chú while này ra  
;sớm quá, lại còn bồng theo thằng "<=" gây rắc rối nữa chú.  
;Thôi thì mặc xác chú ta, để đó rồi trị chú ta sau hé.  
;Cứ tạm hiểu thế này đã: trong khi biến keepGoing còn có  
;giá trị còi hơn là giá trị trả về của hàm strlen với biến  
;strName ta tha hồ thực hiện các phi vụ bên dưới mà không  
;sợ nó ụynh (Nó bự hơn thì nó đêch cho ta làm)  
  
(setq char1(substr strName 1 keepGoing)); Đặt biến char1 là kết quả  
;trả về của hàm substr với biến strName và các thông số là  
;1 và biến keepGoing  
  
(setq char1 (strcase char1)); Đặt lại biến char1 thành chuỗi với các  
;ký tự in hoa
```

```

(setq strName2(strcat char1 (substr strName (+ keepGoing 1)))) ; Đặt
;biến strName2 là chuỗi tổ hợp gồm biến char1 và kết quả
;trả về của hàm substr với biến strName và một tham số
;(keepGoing+1)

(setq keepGoing(+ keepGoing 1)); Cái vụ này là do chú While đề ra
;đây. Đã chót thì phải chét dành để nó đẩy rồi xử sau. Ta
;cứ chấp nhận là nó ra đòi để nhắc ta rằng mỗi khi thực
;hiện xong mấy cái phi vụ bên trên của chú While thì phải
;lại quả cho nó 1 đơn vị nữa nó mới cho ta lặp lại các phi
;vụ trên với giá trị mới của nó

(princ (strcat "\n " strName2)); Hiển thị giá trị biến strName2 ra
;dòng lệnh của AutoCad

) ; Ký tự này bảo ta nghỉ chơi với chú while

(princ "\n Program Complete.") ; Hiển thị chuỗi Program Complete ra
;dòng lệnh của AutoCad

(princ) ; Kết thúc việc thực thi nhiệm vụ để thoát êm

) ; Kết thúc chương trình

```

(Nhập vào dòng nhắc lệnh của AutoCad sau khi đã lưu chương trình vào thư mục trên đĩa cứng như sau:)

Command: (load "namemix")<enter>

Command: namemix<enter>

(Xem hiển thị trong dòng lệnh của AutoCad)

Command: Enter Your Name: jeff<enter>

Command: Jeff

Command: JEff

Command: JEFf

Command: JEFF

Command: Program Complete.



Number Functions

Các hàm xử lý số

[abs](#) [atof](#) [atoi](#) [fix](#) [float](#) [itoa](#) [Example Program](#)

abs - This function returns the absolute value of a number.

Hàm này trả về giá trị tuyệt đối của một số.

Syntax : (**abs number**)

Cú pháp: (**abs Số**)

number - any valid number.[integer or real number]

Số - là một số bất kỳ có giá trị (số thực hoặc số nguyên)

Returns an integer or a real number.

Giá trị trả về của hàm là một số nguyên hoặc một số thực

(**abs 345**) returns (trả về giá trị) 345

(**abs 345.5**) returns (trả về giá trị) 345.5

(**abs -345**) returns (trả về giá trị) 345

(**abs -345.5**) returns (trả về giá trị) 345.5

(**abs "345JEFF"**) returns (trả về giá trị) "Error: Bad Argument Type" ["345JEFF" is a string, not a number] ("345JEFF" là một chuỗi chứ không phải số)

atoi - This function converts a string to a integer.

Hàm này đổi **một chuỗi** thành **số nguyên**

Syntax : (**atoi "string"**)

Cú pháp: (**atoi** “**chuỗi**”)

"string" - any valid string or variable representing a string. If you use a variable omit the quotes.

“chuỗi” - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

Returns an integer.

Giá trị trả về của hàm là một số nguyên

(**atoi** "Jeff345") returns (trả về giá trị) 0

(**atoi** "5") returns (trả về giá trị) 5

(**atoi** "5.6") returns (trả về giá trị) 5

(**atoi** "") returns (trả về giá trị) 0

(**atoi** "345JEFF") returns (trả về giá trị) 3 ; Chỗ này cụ Jeff lầm lẫn cần nên bỏ quên mất hai ký tự 4 và 5. Kết quả đúng phải là 345. Bạn kiểm lại mà xem

(**atoi** 5) returns (trả về giá trị) "Error: Bad argument type". [5 is an integer, not a string]
(5 là một số nguyên chứ không phải chuỗi)

(**atoi** 5.6) returns (trả về giá trị) "Error: Bad argument type". [5.6 is a real number, not a string]
(5.6 là một số thực chứ không phải chuỗi)

This function looks at the first character, then the next, then the next, ect. until it finds a character that cannot be part of an integer.

Hàm này tìm kiếm bắt đầu từ ký tự đầu tiên của chuỗi rồi lần lượt qua các ký tự tiếp theo cho tới khi tìm được ký tự mà nó không thể là một phần của số nguyên thì dừng lại và trả kết quả.

itoa - This function converts an integer to a string.

Hàm này đổi **một số** nguyên thành **một chuỗi**.

Syntax : (**itoa integer**)

Cú pháp: (**itoa Sốnguyên**)

integer - Any valid integer.

Sốnguyên – là một **số** nguyên bất kỳ có giá trị

Returns a string.

Giá trị trả về của hàm là một chuỗi

`(itoa 345)` returns (trả về giá trị) "345"

`(itoa 5)` returns (trả về giá trị) "5"

`(itoa 5.6)` returns (trả về giá trị) "Error: Bad argument type". [5.6 is a real number, not an integer] (5.6 là một số thực không phải số nguyên)

`(itoa "5")` returns (trả về giá trị) "Error: Bad argument type". ["5" is a string not an integer] ("5" là một chuỗi chứ không phải là số nguyên)

`(itoa "345JEFF")` returns (trả về giá trị) "Error: Bad argument type". ["345JEFF" is a string, not an integer] ("345JEFF" là một chuỗi chứ không phải là số nguyên)

This function can only accept an integer or a variable representing an integer as it's parameter.

Hàm này **chỉ nhận một** đối số là số nguyên hoặc **một** biến đại diện cho số nguyên.

atof - This function converts a string to real.

Hàm này đổi một **chuỗi** thành một **số** thực

Syntax : `(atof "string")`

Cú pháp: `(atof "chuỗi")`

"string" - any valid string or variable representing a string. If you use a variable omit the quotes.

"chuỗi" - một **chuỗi** có giá trị hay một biến đại diện cho chuỗi. Nếu sử dụng tên biến thì bỏ qua dấu ngoặc kép.

Returns an real number.

Giá trị trả về của hàm là một số thực.

`(atof "Jeff345")` returns (trả về giá trị) 0.0

`(atof "5")` returns (trả về giá trị) 5.0

`(atof "5.6")` returns (trả về giá trị) 5.6

`(atof "5'3-1/2'")` returns (trả về giá trị) 63.5

`(atof "3-1/2")` returns (trả về giá trị) 3.5

`(atof "")` returns (trả về giá trị) 0.0

`(atof "345JEFF")` returns (trả về giá trị) 345.0

`(atof 345)` returns (trả về giá trị) "Error: Bad Argument Type" [345 is an integer, not a string] (345 là một số nguyên chứ không phải chuỗi)

`(atof 3.4)` returns (trả về giá trị) "Error: Bad Argument Type" [3.4 is a real number, not a string] (3.4 là một số thực chứ không phải một chuỗi)

`(atof (list 3 4))` returns (trả về giá trị) "Error: Bad Argument Type" [(list 3 4) is a list, not a string] ((list 3 4) là một danh sách chứ không phải một chuỗi)

This function looks at the first character, then the next, then the next, ect. until it finds a character that cannot be part of an real number.

Hàm này tìm kiếm bắt đầu từ ký tự đầu tiên của chuỗi rồi lần lượt qua các ký tự tiếp theo cho tới khi tìm được ký tự mà nó không thể là một phần của số thực thì dừng lại và trả kết quả.

fix - This function converts a real to an integer.

Hàm này đổi một **số thực** thành một **số nguyên**

Syntax : **(fix real)**

Cú pháp: **(fix Sốthực)**

real - Any valid real number.

Sốthực – là một **số thực** bất kỳ có giá trị

Returns an integer.

Giá trị trả về của hàm là một **số nguyên**

`(fix 345.0)` returns (trả về giá trị) 345

`(fix 5.0)` returns (trả về giá trị) 5

`(fix 5.6)` returns (trả về giá trị) 5

`(fix "5.0")` returns (trả về giá trị) "Error: Bad Argument Type" ["5.0" is a string, not a real number] ("5.0" là một chuỗi chứ không phải một số thực)

`(fix "5")` returns (trả về giá trị) "Error: Bad Argument Type ["5" is a string, not a real number] ("5" là một chuỗi chứ không phải một số thực)

This function takes the whole number part of a decimal number and returns it as an integer.

Hàm này lấy đi tất cả phần thập phân của số thực và chỉ trả về phần nguyên của nó như một số nguyên

float - This function takes a number (integer or real) and converts it to a real number.

Hàm này lấy một số (số nguyên hoặc số thực) và đổi nó thành một số thực.

Syntax : **(float integer) (float real)**

Cú pháp: **(float Số)**

Integer - Any valid integer.

Số - là một số nguyên hay số thực bất kỳ có giá trị

real - Any valid real number.

`(float 5)` returns (trả về giá trị) 5.0

`(float 345)` returns (trả về giá trị) 345.0

`(float 3.5)` returns (trả về giá trị) 3.5 [No effect, but no error] (Không có hiệu quả nhưng không sai)

`(float "3")` returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] ("3" là một chuỗi chứ không phải một số)

`(float "3.5")` returns (trả về giá trị) "Error: Bad Argument Type" ["3.5" is a string, not a number] ("3.5" là một chuỗi chứ không phải một số)

`(float "abc")` returns (trả về giá trị) "Error: Bad Argument Type" ["abc" is a string, not a number] ("abc" là một chuỗi chứ không phải một số)

Example Program 1:

Chương trình mẫu: ; Những gì đã giải thích ở các phần trước có thể sẽ không được nhắc lại ở đây.

```
(defun C:myProg()

  (setq intAge(getint "\n Enter your Age: ")); Đặt biến intAge là kết
    ;quả của hàm getint. Cái câu getint này có bà con với chú
    ;getString ở phần trước nên tôi chịu, phải để lại ngâm cứu
    ;sau. Cứ tạm hiểu là nó dùng để nhập dữ liệu dạng số sau dòng
    ;thông báo (ở đây là "Enter your Age")

  (setq intDays(* intAge 365)) ;Đặt biến intDays là kết quả của hàm
    ;nhân biến intAge với 365. Cái hàm nhân (*) này sẽ được trình
    ;nhìn ở phần sau. Bạn cứ yên trí nhé.

  (princ (strcat "\n You are " (itoa intDays) " Days old!")); Hiển thị
    ;các ký tự của chuỗi tổ hợp "You are" với chuỗi kết quả của
    ;hàm itoa áp dụng cho biến intDays và thêm "Days old". Chú ý
    ;tới nhóm ký tự "\n" ở đây có nghĩa là hiển thị kết quả ở một
    ;dòng riêng dưới dòng lệnh của AutoCad

  (princ) ; Cái này..... Biết rồi! Khổ lắm... Nói mãi...

)
```

Execution: Chơi thử coi:

Command: (load "myProg")<enter>

Command: myProg<enter>

Command: Enter your Age: 39<enter>

Command: You are 14235 Days old!

List Functions:

Các hàm xử lý danh sách:

[car](#) [cdr](#) [cadr](#) [caddr](#) [caar](#) [cddr](#) [foreach](#) [list](#) [cons](#) [nth](#)

[Example Programs](#)

You are about to unravel the mystery behind the car of a cdr. Better sit down.

Bạn sắp lột truồng hai thằng car và cdr rồi. Tốt hơn là hãy ngồi xuống đã. Đừng nóng

car - This function returns the first item of a list. The item can be a list as shown in example 3 below.

Đây là hàm trả về giá trị là **khoản mục** đầu tiên của một **danh sách**. Khoản mục có thể là một **danh sách**, một **số** hay một **chuỗi** như các ví dụ dưới đây

Syntax : (**car list**)

Cú pháp: (**car Danh sách**)

list - Any valid list.

Danh sách – là một **danh sách** có giá trị bất kỳ

(car (1 2 3)) returns (trả về giá trị) 1

(car ("ab" "cde" "Jeff")) returns (trả về giá trị) "ab"

(car ((1 2 3) (4 5 6))) returns (trả về giá trị) (1 2 3)

(car 1) returns (trả về giá trị) "Error: Bad Argument Type" [1 is an integer, not a list] (1 là một số chứ không phải một danh sách)

(car "Jeff") returns (trả về giá trị) "Error: Bad Argument Type" ["Jeff" is a string, not a list] ("Jeff" là một chuỗi chứ không phải là danh sách)

(car (1 (2 3))) returns (trả về giá trị) 1

(car ((1 2) 3)) returns (trả về giá trị) (1 2)

This function is mainly used to get the x coordinate of a point. [(car (x y z)) returns x]

Hãy nhớ bừu bối này nhé (**car (x y z)**) . Hàm này trả về giá trị toạ độ x của một điểm cho trước

cdr - This function returns a list that includes everything but the first item in a list. The item can be a list as shown in example 3 below.

Hàm này trả về một **danh sách** bao gồm mọi khoản mục trừ khoản mục đầu tiên của một danh sách cho trước. Khoản mục có thể là một danh sách , một số hay một chuỗi như trong các ví dụ dưới đây.

Syntax : (**cdr list**)

Cú pháp: (**cdr Danh sách**)

list - Any valid list.

Danh sách – là một **danh sách** có giá trị bất kỳ

`(cdr (1 2 3))` returns (trả về giá trị) `(2 3)`

`(cdr ("ab" "cde" "Jeff"))` returns (trả về giá trị) `("cde" "Jeff")`

`(cdr ((1 2 3) (4 5 6)))` returns (trả về giá trị) `(4 5 6)` [since `(4 5 6)` is the second item in the list]. (Do danh sách `(4 5 6)` là khoản mục thứ hai của danh sách đã cho)

`(cdr 1)` returns (trả về giá trị) `"Error: Bad Argument Type"` [1 is an integer, not a list] (`1` là một số nguyên chứ không phải là một danh sách)

`(cdr "Jeff")` returns (trả về giá trị) `"Error: Bad Argument Type"` [`"Jeff"` is a string, not a list] (`"Jeff"` là một chuỗi chứ không phải là một danh sách)

`(cdr (1 (2 3)))` returns (trả về giá trị) `(2 3)`

`(cdr ((1 2) 3))` returns (trả về giá trị) `3`

Mystery HINT (Mách lẻo tí) : `(car (cdr (1 (2 3))))` returns (trả về giá trị) `2` . [So, that's what a car of a cdr is] (Vậy đó, đó là kết quả trả về khi thẳng car kéo bè thêm thẳng cdr)

if the cdr of `(1 (2 3))` returns `(2 3)` and the car of `(2 3)` returns `2` then the mystery is solved.

Vì `(cdr (1 (2 3)))` trả về giá trị `(2 3)` và `(car (2 3))` trả về giá trị `2` , thế là bí mật bị bật mí nhé.

The car of a cdr is the first item in the second item in the list.

Vậy là car rồi cdr sẽ cho kết quả là khoản mục đầu tiên trong trong khoản mục thứ hai của danh sách có hai khoản mục cho trước.

cadr - This function returns the second item in a list. The item can be a list as shown in example 3 below.

Hàm này trả về giá trị là **khoản mục** thứ hai của một danh sách cho trước. Khoản mục có thể là một danh sách, một số hay một chuỗi như trong các ví dụ bên dưới.

Syntax : **(cadr list)**

Cú pháp: **(cadr Danh sách)**

list - Any valid list.

Danh sách – là một danh sách có giá trị bất kỳ

`(cadr (1 2 3))` returns (trả về giá trị) 2

`(cadr ("ab" "cde" "Jeff"))` returns (trả về giá trị) "cde"

`(cadr ((1 2 3) (4 5 6) (7 8 9)))` returns (trả về giá trị) (4 5 6)

`(cadr 1)` returns (trả về giá trị) "Error: Bad Argument Type" [1 is an integer, not a list] (1 là một số nguyên chứ không phải là danh sách)

`(cadr "Jeff")` returns (trả về giá trị) "Error: Bad Argument Type" ["Jeff" is a string, not a list] ("Jeff" là một chuỗi chứ không phải một danh sách)

`(cadr (1 (2 3)))` returns (trả về giá trị) (2 3)

`(cadr ((1 2) 3))` returns (trả về giá trị) 3

`(cadr (1))` returns (trả về giá trị) nil [There isn't a second item] (Không có khoản mục thứ hai)

This function is mainly used to get the y coordinate of a point. [`(cadr (x y z))` returns y]

Hàm chủ yếu được sử dụng để xác định tọa độ y của một điểm cho trước là **(cadr (x y z))**

; Có thể coi rằng cadr là một thằng car cặp thêm thằng cdr nữa: `(cadr(list))=(car(cdr(list)))`

caddr - This function returns the third item in a list. The item can be a list as shown in example 3 below.

Hàm này trả về giá trị là **khoản mục** thứ ba của một danh sách cho trước. Khoản mục có thể là một danh sách, một số hay một chuỗi như trong các ví dụ bên dưới.

Syntax : **(caddr list)**

Cú pháp: **(caddr Danh sách)**

list - Any valid list.

Danh sách – là một danh sách có giá trị bất kỳ

`(caddr (1 2 3))` returns (trả về giá trị) 3

`(caddr ("ab" "cde" "Jeff"))` returns (trả về giá trị) "Jeff"

`(caddr ((1 2 3) (4 5 6) (7 8 9)))` returns (trả về giá trị) (7 8 9)

`(caddr 1)` returns (trả về giá trị) "Error: Bad Argument Type" [1 is an integer, not a list] (1 là một số nguyên chứ không phải một danh sách)

`(caddr "Jeff")` returns (trả về giá trị) "Error: Bad Argument Type" ["Jeff" is a string, not a list] ("Jeff" là một chuỗi chứ không phải một danh sách)

`(caddr (1 (2 3)))` returns (trả về giá trị) nil [There isn't a third item] (không có khoản mục thứ 3)

`(caddr ((1 2) 3 (5 6) 7))` returns (trả về giá trị) (5 6)

`(caddr (1))` returns (trả về giá trị) nil [There isn't a third item] (Không có khoản mục thứ 3)

This function is mainly used to get the z coordinate of a point. [`(caddr (x y z))` returns z]

Hàm chủ yếu để xác định tọa độ z của một điểm cho trước là **`(caddr (x y z))`**

; Cùn một tí nhé `(caddr(list) = (car (cadr(list)) = (car(cdr(cdr(list))))`

caar - This function returns the first item of the first item in a list. The item can be a list as shown in example 3 below.

Hàm này trả về giá trị là **khoản mục** đầu tiên của khoản mục thứ nhất trong danh sách đã cho. Khoản mục có thể là một danh sách, một số hay một chuỗi như trong các ví dụ bên dưới

Syntax : **`(caar list)`**

Cú pháp: **`(caar Danhsách)`**

list - Any valid list.

Danhsách – là một danh sách có giá trị bất kỳ

`(caar (1 2 3))` returns (trả về giá trị) "Error: Bad Argument Type" [The first item in the list is not a list] (Khoản mục đầu tiên trong danh sách đã cho là một số nguyên chứ không phải là danh sách)

`(caar ((1 2 3) (4 5 6)))` returns (trả về giá trị) 1

`(caar (("ab" "cde") ("Jeff")))` returns (trả về giá trị) "ab"

`(caar ("Jeff" 1 2 3) ("x" "y" "z") (4 5 6))` returns (trả về giá trị) "Jeff"

`(caar 1)` returns (trả về giá trị) "Error: Bad Argument Type" [1 is an integer, not a list] (1 là một số chứ không phải danh sách)

`(caar "Jeff")` returns (trả về giá trị) "Error: Bad Argument Type" ["Jeff" is a string, not a list] ("Jeff" là một chuỗi chứ không phải một danh sách)

`(caar (1 (2 3)))` returns (trả về giá trị) "Error: Bad Argument Type" [The first item in the list is not a list] (Khoản mục đầu tiên trong danh sách đã cho là số nguyên chứ không phải là một danh sách)

`(caar ((1 2) 3))` returns (trả về giá trị) 1

; Theo kiểu cùn : `(caar(list)) = (car(car(list)))` . Vậy nên khi `(car(list))` trả về giá trị là một số hay một chuỗi thì `(car(car(list)))` ngỏm là phải lý quá còn gì.

cddr - This function returns a list that includes everything after the second item in a list.

Hàm này trả về giá trị là một **danh sách** bao gồm mọi thứ linh tinh sau khoản mục thứ hai của một danh sách đã cho.

Syntax : **(cddr list)**

Cú pháp: **(cddr Danhsách)**

list - Any valid list.

Danhsách – là một danh sách có giá trị bất kỳ

`(cddr (1 2 3))` returns (trả về giá trị) (3)

`(cddr ("ab" "cde" "Jeff" "Sanders"))` returns (trả về giá trị) ("Jeff" "Sanders")

`(cddr ((1 2 3) (4 5 6) (7 8 9) (10 11 12)))` returns (trả về giá trị) ((7 8 9)(10 11 12))

`(cddr 1)` returns (trả về giá trị) "Error: Bad Argument Type" [1 is an integer, not a list] (1 là một số nguyên chứ không phải một danh sách)

`(cddr "Jeff")` returns (trả về giá trị) "Error: Bad Argument Type" ["Jeff" is a string, not a list] ("Jeff" là một chuỗi chứ không phải một danh sách)

`(cddr (1 (2 3)))` returns (trả về giá trị) nil [There isn't a third item] (Không có khoản mục thứ ba)

`(cddr ((1 2) 3 (5 6) 7))` returns (trả về giá trị) `((5 6) 7)`

`(cddr (1))` returns (trả về giá trị) `nil` [There isn't a third item] (Không có khoản mục thứ ba)

foreach - This function steps through each item in the list and returns the last value.

Hàm này yêu cầu **thực hiện cùng một nhiệm vụ** đối với mỗi một chú chàng bị tóm trong một danh sách cho trước và trả về **kết quả thực hiện** với chú chàng sau rốt.

; Ôi giỏi ời! Cái cụ JEFF này lảm cẩm quá rồi. Cái anh **foreach** này có họ hàng với chú **While** mà ; ta đã gặp ở phần trước. Nhẽ ra cụ phải gom lại rồi giới thiệu một thể, chứ làm kiểu này, con cháu ; nó ra họ nhà tôm cả thì chết. BỐ ai mà nhớ nổi.

Syntax : (**foreach** **varName** **list** **yourFunctions**)

Cú pháp: (**foreach** **Tênbiến** **Danhsách** **Cáchànthựchiện**)

varName - Any valid variable name that you make up.

Tênbiến – là **một tên biến** có giá trị bất kỳ được lần lượt gán cho giá trị của những chú bị tóm.

list - Any valid list.

Danhsách – là một danh sách các chú chàng bị tóm

yourFunctions - Any valid autolisp functions.

Cáchànthựchiện – là các hàm Autolisp có giá trị bất kỳ cần được thực hiện với biến đã chọn

`(foreach a (list 1 2 3) (princ a))` prints 123 to the screen and returns 3 (hiển thị 123 ra màn hình và trả về giá trị 3 [same as (princ 1) (princ 2) (princ 3) except that it only returns the last value.] (tương tự như thực hiện (princ 1)(princ 2)(princ 3) ngoại trừ việc nó chỉ trả về giá trị thực hiện cuối cùng)

`(foreach a (list 1 2 3) (princ (+ a 5)))` prints 678 to the screen and returns 8 (Hiển thị 678 ra màn hình và trả về giá trị 8 [same as (princ (+ 1 5)) (princ (+ 2 5)) (princ (+ 3 5)) except that it only returns the last value.] (tương tự như (princ (+1 5)) (princ (+ 2 5)) (princ (+ 3 5)) ngoại trừ việc nó trả về giá trị thực hiện cuối cùng)

list - This function creates a list.

Hàm này trả về giá trị là một **danh sách** gồm các khoản mục đối số của hàm

Syntax : (**list** **Item**)

Cú pháp: (list **Cáckhoảnmục**)

Syntax : (list **firstItem secondItem**)

Syntax : (list **firstItem secondItem thirdItem...**)

Item - Any valid item including a list.

Cáckhoảnmục – là những **khoảnmục** được đặt cách nhau một khoảng trắng như các đối số của hàm list .Mỗi **khoảnmục** có thể là một chuỗi, một số hay một danh sách

(list) returns (trả về giá trị) nil [an Empty list] (Một danh sách rỗng)

(list 1 2 3) returns (trả về giá trị) (1 2 3)

(list "ab" "cde" "Jeff" "Sanders") returns (trả về giá trị) ("ab" "cde" "Jeff" "Sanders")

(list (list 1 2 3) (list 4 5 6)) returns (trả về giá trị) ((1 2 3) (4 5 6))

(list 1) returns (trả về giá trị) (1)

(list "Jeff") returns (trả về giá trị) ("Jeff")

(list 1 (list 2 3)) returns (trả về giá trị) (1 (2 3))

(list (list 1 2) 3 (list 5 6) 7) returns (trả về giá trị) ((1 2) 3 (5 6) 7)

cons - This function takes an item and a list, and returns the addition of that item to the beginning of the list. The first item can be an atom or a list.

Hàm này nhận **một khoảnmục** và **một danh sách** rồi trả về **một danh sách** bắt đầu với khoảnmục đã chọn rồi tiếp theo là danh sách đã chọn. Khoảnmục chọn có thể là một phần tử hoặc một danh sách

Syntax : (cons **Item list**)

;thanks tim!

Cú pháp: (cons **Khoảnmục Danhsách**)

; Cám ơn bỏ!

Item - Any valid item including a list.

Khoảnmục – là một khoảnmục có giá trị bất kỳ kể cả một danh sách

list - Any valid list.

Danhsách – là một danh sách có giá trị bất kỳ

`(cons)` returns (trả về giá trị) "Error: Too Few Arguments" [cons requires an item and a list] (hàm cons đòi hỏi phải có một khoản mục và một danh sách làm đối số)

`(cons 1 2 3)` returns (trả về giá trị) "Error: Too Many Arguments" [cons requires an item and a list] (hàm cons đòi hỏi phải có một khoản mục và một danh sách làm đối số)

`(cons "ab" (list "cde" "Jeff" "Sanders"))` returns (trả về giá trị) ("ab" "cde" "Jeff" "Sanders")

`(cons (list 1 2 3) (list 4 5 6))` returns (trả về giá trị) ((1 2 3) 4 5 6)

`(cons 1)` returns (trả về giá trị) "Error: Too Few Arguments" [cons requires an item and a list] (hàm cons đòi hỏi phải có một khoản mục và một danh sách làm đối số)

`(cons "Jeff")` returns (trả về giá trị) "Error: Too Few Arguments" [cons requires an item and a list] (hàm cons đòi hỏi phải có một khoản mục và một danh sách làm đối số)

`(cons 1 (list 2 3))` returns (trả về giá trị) (1 2 3) [notice the difference here from the list function above] (Chú ý tới sự khác nhau ở đây với hàm list ở trên)

;Hàm `(list 1 (list 2 3))` trả về giá trị `(1 (2 3))`

`(cons "Jeff" (list 1 2 3 5 6 7))` returns (trả về giá trị) ("Jeff" 1 2 3 5 6 7)

nth - This function returns the Nth item in a list. The first item in a list is item zero.

Hàm này trả về giá trị là **khoản mục** thứ n của một danh sách cho trước. Khoản mục đầu tiên của danh sách là khoản mục thứ 0.

; Ấy da, cái vụ này khác với số vị trí của ký tự trong một chuỗi à nha. Làm lộn là làm lại đó.

Syntax : **(nth integer list)**

Cú pháp: **(nth Sốnguyên Danhsách)**

integer - Any valid integer.

Sốnguyên – là một số nguyên có giá trị bất kỳ

list - Any valid list.

Danhsách – là một danh sách có giá trị bất kỳ

```

(nth) returns (trả về giá trị) "Error: Too Few Arguments" [nth requires an integer and a
list] (Hàm nth đòi hỏi một số nguyên và một danh sách làm đối số)

(nth 2 1 2 3) returns (trả về giá trị) "Error: Bad Argument Type" [nth requires an integer
and a list] (Hàm nth đòi hỏi một số nguyên và một danh sách làm đối số)

(nth 2 (list "cde" "Jeff" "Sanders")) returns (trả về giá trị) "Sanders"

(nth 0 (list 1 2 3)) returns (trả về giá trị) 1

(nth 1 (list 1 2 3)) returns (trả về giá trị) 2

(nth 2 (list 1 2 3)) returns (trả về giá trị) 3

(nth 1 (list "a" (list "b" "c") "d")) returns (trả về giá trị) ("b" "c")

(nth 4 (list 1 2 3 5 6 7)) returns (trả về giá trị) 6

```

Example Program 1:

Chương trình mẫu 1:

```

(defun C:myProg() [define program] (xác định tên chương trình)

  (setq pt1(getpoint "\n First Corner: ")) [get first point] (chọn
                                              điểm thứ nhất)

  ; Cái chú getpoint này cũng có bà con với mấy chú get..... ở các
  ;phần trước, ai quen biết thì chỉ giùm nhé. Tớ cứ hiểu đại
  ;khái là chú bắt ta phải chọn một điểm nào đó trong AutoCad và
  ;chú trả lại ta các toạ độ của điểm vừa chọn để ta xài chùa .
  ;Do vậy dòng lệnh trên đây có nghĩa là gán cho biến pt1 bộ toạ
  ;độ của điểm mà bạn sẽ chọn trong AutoCad

  (setq pt2(getcorner pt1 "\n Last Corner: ")) [get other point on a
  square] (Chọn điểm khác ở góc vuông đối diện với điểm pt1)

  ; Ấy, tớ nói vậy chả biết có phải vậy không? Nhỡ có sai thì
  ;điều chỉnh lại giùm nhé. Ở đây lại tòi ra một chú get..... nữa
  ;là getcorner. Đáng GHÉT thật!!!Chú này tớ hiểu là chú ấy bảo
  ;tớ nhét giùm chú một điểm pt2 nữa trong AutoCad rồi chú sẽ trả
  ;công tớ là một miền hình chữ nhật tưởng tượng trong AutoCad

```

;nhận pt1 và pt2 làm hai đỉnh đối xứng qua tâm và các cạnh của
;nó thì song song với các trục tọa độ của AutoCad. Đồng thời
;chú cũng biểu tỏ luôn bộ tọa độ của pt2 để sử dụng dưới đây.

```
(setq x1(car pt1)) [get x coordinate of pt1] (lấy tọa độ x của pt1)
(setq x2(car pt2)) [get x coordinate of pt2] (lấy tọa độ x của pt2)
(setq y1(cadr pt1)) [get y coordinate of pt2] (lấy tọa độ y của pt1)
(setq y2(cadr pt2)) [get y coordinate of pt2] (lấy tọa độ y của pt2)
(setq hor1(- x2 x1))[get horizontal distance] (lấy khoảng cách theo
trục x giữa hai điểm pt1 và pt2)
(setq ver1(- y2 y1))[get vertical distance] (lấy khoảng cách theo
trục y giữa hai điểm pt1 và pt2)
```

; Hàm trừ (-) cũng như hàm nhân (*) ở phần trước ta để dành xoi
;sau vì nó không xương lăm. Đừng vội, nó có chạy đằng trời.

```
(princ "\n Horizontal Distance = ") (princ hor1) [print to screen]
```

(Dùng các hàm princ để in ra màn hình dòng thông báo Horizontal
Distance = giá trị biến hor1)

```
(princ "\n Vertical Distance = ") (princ ver1) [print to screen]
```

(Dùng các hàm princ để in ra màn hình dòng thông báo Vertical
Distance = giá trị biến ver1)

```
(princ) [supress echo..clean exit] ;Khỏ lăm
```

```
) [close program] ; Nói mãi
```

Execution: Đùa với AutoCad

Command: (load "myProg")<enter>

Command: myProg<enter>

Command: First Corner: <pick a point>

Command: Last Corner: <pick a point>

Command: Horizontal Distance = 23.5

Command: Vertical Distance = 11.5

Example Program 2:

Chương trình mẫu 2:

```
(defun C:AddText() [define program]

  (setq myList(list "e" "b" "c" "m" "at")) [make a list of 6 letters]

  (Đặt biến myList là một danh sách có 6 chữ cái như mô tả)

  (foreach myVar myList [start the foreach loop]

    (Bắt đầu vòng lặp foreach với biến myVar nhận các giá trị trong danh sách myList)

    (princ (strcat myVar (nth 4 myList))) [define a valid function]

    (Hiển thị các ký tự của chuỗi tổ hợp bởi biến myVar và chuỗi "at")

    ; Chả biết cụ Jeff có nhầm không chứ theo tớ thì phải có thêm chuỗi
    ; ký tự "\n" vào trong hàm strcat mới có được kết quả đúng khi chơi.

  ) [close the foreach statement] (Đóng vòng lặp foreach)

  (princ) [supress echo..clean exit]

) [close program]
```

Execution: Chơi với AutoCad

Command: (load "AddText")<enter>

Command: AddText<enter>

Command: eat

Command: bat

Command: cat

Command: mat

Command: atat

End of List Functions

Kết thúc các hàm xử lý danh sách

Entity DXF Group Codes

Các mã nhóm DXF của các đối tượng trong AutoCad

All entities inside an AutoCAD drawing has DXF Group codes. The group codes define the properties of each entity. Each code inside the group is actually an associated list. The group codes for a **line** may look something like this:

Tất cả các đối tượng trong một bản vẽ AutoCad đều có các mã nhóm DXF. Các mã nhóm đó xác định các thuộc tính của mỗi một đối tượng. Mỗi một mã trong nhóm là một danh sách tương tác chặt chẽ. Các mã nhóm của một **đoạn thẳng** trông giống như sau:

```
((-1 . <Entity name: 3b40868>) (0 . "LINE") (5 . "BD") (100 .  
"AcDbEntity") (67 . 0) (8 . "DIM") (100 . "AcDbLine") (10 200.25  
316.75 0.0)  
(11 242.25 316.75 0.0) (210 0.0 0.0 1.0))
```

Where: Trong đó:

Group code -1 is the name of the entity. (Mã nhóm -1 là tên của đối tượng)

Group code 0 is the type of entity. (Mã nhóm 0 là loại đối tượng)

Group code 8 is the layer name. (Mã nhóm 8 là tên lớp đặt đối tượng)

Group code 10 is the start point of the line. (Mã nhóm 10 là tọa độ điểm bắt đầu của đoạn thẳng)

Group code 11 is the end point of the line. (Mã nhóm 11 là tọa độ điểm cuối của đoạn thẳng)

The group codes for a **circle** may look something like this: (Các mã nhóm cho **đường tròn** như sau:)

```
((-1 . <Entity name: 3b40cf0>) (0 . "CIRCLE") (5 . "146") (100 .  
"AcDbEntity") (67 . 0) (8 . "HOL") (100 . "AcDbCircle") (10 163.135  
367.479 0.0) (40 . 2.56277) (210 0.0 0.0 1.0))
```

Where: (Trong đó:)

Group code -1 is the name of the entity. (Mã nhóm -1 là tên của đối tượng)

Group code 0 is the type of entity. (Mã nhóm 0 là loại đối tượng)

Group code 8 is the layer name. (Mã nhóm 8 là tên lớp đặt đối tượng)

Group code 10 is the center of the circle. (Mã nhóm 10 là tọa độ tâm của đường tròn đối tượng)

Group code 40 is the radius of the circle. (Mã nhóm 40 là bán kính của đường tròn đối tượng)

How do you list the codes for an entity? Type this at the command line, press enter, and select an entity.

Làm sao để tạo danh sách các mã nhóm của một đối tượng? Hãy nhập điều này vào dòng nhắc lệnh, nhấn Enter, và chọn một đối tượng.

```
(entget (car (entsel)))
```

Entsel selects the entity, **car** gets the entity name from entsel, and **entget** returns the Group Codes of the entity.

Hàm **entsel** để lựa chọn đối tượng, hàm **car** lấy giá trị tên đối tượng được chọn từ hàm entsel, hàm **entget** trả về giá trị các mã nhóm của đối tượng đưa vào. (Một danh sách tương tác)

; Vậy là hàm entsel phải trả về một danh sách bắt đầu bằng tên đối tượng được chọn trong AutoCad, còn hàm entget sẽ lấy đối số là tên của đối tượng được chọn.

What are the numbers in the group code for? They aide in extracting data from the group codes. Take for instance this code:

Các số trong mã nhóm để làm gì vậy? Chúng sẽ hỗ trợ cho việc trích xuất các dữ liệu từ các mã nhóm. Ví dụ dòng mã sau đây:

```
(setq entList(entget (car (entsel))))
```

If I selected a circle, this code would set entList to something like this:

Nếu tôi chọn một vòng tròn, dòng mã (lệnh) trên sẽ đặt biến entList thành một danh sách tương tác sau:

```
((-1 . <Entity name: 3b40cf0>) (0 . "CIRCLE") (5 . "146") (100 .
```

```
"AcDbEntity") (67 . 0) (8 . "HOL") (100 . "AcDbCircle") (10 163.135
367.479 0.0) (40 . 2.56277) (210 0.0 0.0 1.0))
```

If I wanted to extract data from this group code list I would simply type:

Nếu tôi muốn trích xuất các dữ liệu từ danh sách tương tác của các mã nhóm, rất đơn giản tôi sẽ nhập:

; Ở đây phải dùng hàm assoc với một số nguyên và một danh sách tương tác làm đối số của hàm

(assoc 40 entList) ;would return (trả về giá trị) 2.56277

```
(assoc 0 entList) ;would return (trả về giá trị) "CIRCLE"
```

(assoc 10 entList) ;would return (trả về giá trị) (163.135 367.479 0.0)

```
(assoc 8 entList) ;would return ( trả về giá trị ) "HOL"
```

An associated list can make extraction a very easy thing to accomplish.

Có thể trích xuất dễ dàng các dữ liệu muốn lấy từ một danh sách tương tác bằng cách sử dụng hàm `assoc`.

Entity DXF Group Code Table

Bảng mã nhóm DXF của các đối tượng

; Bảng này dùng để tham khảo khi muốn tìm hiểu rõ hơn về các mã nhóm DXF. Tôi lười dịch

The list of DXF group Codes for the entities applied in AutoCad

Entity Types: (Group Code 0)

LN = Line CI = Circle AR = Arc TR = Trace SD = Solid TX = Text 3DF = 3DFace AD = AttDef

PT = Point PL = PolyLine VT = Vertex DM = Dimension AT = Attribute IN = Insert SQ = Seqend

<div style="text-align: center;"> ----- Entity Type (Group Code 0) ----- </div>																	
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ

[illegible]

[illegible]

25	Y of Definition Point														25		
26	Y of Definition Point														26		
30	Z of Start or Insert Point	30	30					30	30	30	30	30		30			
	Z of Center Point			30	30												
	Z of Corner Point					30	30									30	
	Z of Definition Point														30		
	Z of Elevation Point (2D Poly)															30	
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ
31	Z of End Point	31															
	Z of Corner Point					31	31									31	
	Z of Alignment Point							31			31	31					
	Z of Middle Point of Dimension														31		
32	Z of Corner Point					32	32									32	
	Z of Insert Point														32		
33	Z of Corner Point					33	33									33	
	Z of Definition Point														33		
34	Z of Definition Point														34		
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ
35	Z of Definition Point														35		
36	Z of Definition Point														36		
38	Entity Elevation	38	38	38	38	38	38	38	38	38	38	38	38		38		38
39	Entity Thickness	39	39	39	39	39	39	39	39		39	39	39	39	39		39

40	Radius			40	40												
	Height, Size, or Width							40	40		40	40	40				
	Leader Length														40		
41	X Scale Factor or Width							41	41		41	41	41	41			
42	Y Scale Factor or Bulge									42					42		
43	Z Scale Factor									43							
44	Column Spacing									44							
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ
45	Row Spacing									45							
50	Rotation Angle		50					50	50	50	50	50			50		
	Start Angle				50												
	Curve Fit Tangent													50			
51	End Angle				51												
	Obliquing Angle							51	51		51	51					
	Angle from Horizontal													51			
62	Color	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62
66	Entities Follow Flag									66			66				
70	Dimension Type														70		
	Vertex or Polyline Flag												70	70			
	Attribute Flag										70	70					
	Column Count									70							
	Invisible Edges Flag															70	
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ
71	Text Generation Flag							71			71	71					

	Row Count										71						
	Mesh M Vertex Count												71				
72	Text Justification							72			72	72					
	Mesh N Vertex Count												72				
73	Field Length										73	73					
	Smooth Surface M Density												73				
74	Smooth Surface N Denstiy												74				
75	Smooth Surface Type												75				
210	X of Extrusion Point	210	210	210	210	210	210	210	210	210	210	210	210		210		
220	Y of Extrusion Point	210	210	210	210	210	210	210	210	210	210	210	210		210		
230	Z of Extrusion Point	210	210	210	210	210	210	210	210	210	210	210	210		210		
CODE	DESCRIPTION	LN	PT	CI	AR	TR	SD	TX	SH	IN	AD	AT	PL	VT	DM	3DF	SQ

The idea for this table was from the book "Inside AutoLisp - Using AutoLisp to Customize AutoCAD" by Joseph Smith and Rusty Gesner.

Conversion Functions:

Các hàm chuyển đổi:

[fix](#) [float](#) [itoa](#) [atoi](#) [atof](#) [rtos](#) [angtos](#)

; Các hàm fix, float, itoa, atoi, atof đã giới thiệu ở phần trước nên ta khỏi mất công nhé.

fix - This function returns an integer. (NOTE: This function **does not** round numbers off!)

Syntax : (**fix number**)

number - any valid number.[integer or real number]

(fix 345) returns 345

(fix 345.9) returns 345

(fix -345) returns -345

(fix -345.9) returns -345

(fix "345JEFF") returns "Error: Bad Argument Type" ["345JEFF" is a string, not a number]

float - This function returns a real number.

Syntax : (**float number**)

number - any valid number.[integer or real number]

(float 345) returns 345.0

(float 345.9) returns 345.9

(float -345) returns -345.0

(float -345.9) returns -345.9

(float "345JEFF") returns "Error: Bad Argument Type" ["345JEFF" is a string, not a number]

itoa - This function changes an integer to a string.

Syntax : (**itoa integer**)

integer - any valid integer.

(itoa 345) returns "345"

(itoa 345.9) returns "Error: Bad Argument Type" [345.9 is a real number, not an integer]

integer]

`(itoa -345)` returns "-345"

`(itoa -345.9)` returns "Error: Bad Argument Type" [-345.9 is a real number, not an integer]

`(itoa "345JEFF")` returns "Error: Bad Argument Type" ["345JEFF" is a string, not an integer]

atoi - This function changes a string into an integer.

Syntax : **(atoi string)**

string - any valid string.

`(atoi "345")` returns 345

`(atoi "345.9")` returns 345

`(atoi 345)` returns "Error: Bad Argument Type" [345 is a number, not a string]

`(atoi 345.9)` returns "Error: Bad Argument Type" [345.9 is a number, not a string]

`(atoi "-345")` returns -345

`(atoi "-345.9")` returns -345

`(atoi "345JEFF49")` returns 345

`(atoi "JEFF49")` returns 0

Note: This function starts at the beginning of the string and continues until it finds a character that cannot be an integer.

atof - This function changes a string into a real number.

Syntax : **(atof string)**

string - any valid string.

`(atof "345")` returns 345.0

`(atof "345.9")` returns 345.9

`(atof 345)` returns "Error: Bad Argument Type" [345 is a number, not a string]

`(atof 345.9)` returns "Error: Bad Argument Type" [345.9 is a number, not a string]

`(atof "-345")` returns -345.0

`(atof "-345.9")` returns -345.9

`(atof "345JEFF49")` returns 345.0

`(atof "JEFF49")` returns 0.0

Note: This function starts at the beginning of the string and continues until it finds a character that cannot be a number.

rtos - This function converts numbers to a formatted string representing a distance.

Hàm này đổi một **số** thành một **chuỗi** định dạng trước đại diện cho một khoảng cách độ dài.

Syntax : (**rtos number mode precision**)

Cú pháp: (**rtos Số Mode Độchínhxác**)

number - Any valid number. [Real or Integer]

Số - là một **số** nguyên hay **số** thực có giá trị bất kỳ

mode - This directly corresponds to LUnits system variable.

Mode – là một **số** đại diện cho mode tương ứng trực tiếp với biến hệ thống đơn vị đo độ dài

precision - This directly corresponds to the LUPrec system variable.

Độchínhxác – là một **số** đại diện cho mode tương ứng trực tiếp với biến hệ thống độ chính xác của đơn vị đo độ dài.

mode = 1 = Scientific (**Mode** = 1 tương ứng với hệ thống đo độ dài trong khoa học)

with **precision** (với **Độchínhxác**) = 1 (`rtos 456.5 1 1`) returns (trả về giá trị) "4.6E+02"

with **precision** (với Độchínhxác) = 2 (rtos 456.5 1 2) returns (trả về giá trị) "4.57E+02"
 with **precision**(với Độchínhxác) = 3 (rtos 456.5 1 3) returns (trả về giá trị) "4.565E+02"
 with **precision** (với Độchínhxác)= 4 (rtos 456.5 1 4) returns (trả về giá trị) "4.5650E+02"
 with **precision** (với Độchínhxác)= 5 (rtos 456.5 1 5) returns (trả về giá trị) "4.56500E+02"
 with **precision** (với Độchínhxác)= 6 (rtos 456.5 1 6) returns (trả về giá trị) "4.565000E+02"
 with **precision** (với Độchínhxác)= 7 (rtos 456.5 1 7) returns (trả về giá trị) "4.5650000E+02"
 ect.....

mode = 2 = Engineering (Mode = 2 tương ứng với hệ thống đo độ dài trong kỹ thuật)

with **precision** (với Độchínhxác) = 1 (rtos 456.5 2 1) returns (trả về giá trị) "456.5"
 with **precision** (với Độchínhxác) = 2 (rtos 456.5 2 2) returns (trả về giá trị) "456.50"
 with **precision** (với Độchínhxác) = 3 (rtos 456.5 2 3) returns (trả về giá trị) "456.500"
 with **precision** (với Độchínhxác) = 4 (rtos 456.5 2 4) returns (trả về giá trị) "456.5000"
 with **precision** (với Độchínhxác) = 5 (rtos 456.5 2 5) returns (trả về giá trị) "456.50000"
 with **precision** (với Độchínhxác) = 6 (rtos 456.5 2 6) returns (trả về giá trị) "456.500000"
 with **precision** (với Độchínhxác) = 7 (rtos 456.5 2 7) returns (trả về giá trị) "456.5000000"
 ect.....

mode = 3 = Decimal (Mode = 3 tương ứng với hệ thống đo độ dài trong kỹ thuật Anh-Mỹ)

with **precision** (với Độchínhxác) = 1 (rtos 456.5 3 1) returns (trả về giá trị) "38'-0.5""
 with **precision** (với Độchínhxác) = 2 (rtos 456.5 3 2) returns (trả về giá trị) "38'-0.50""
 with **precision** (với Độchínhxác) = 3 (rtos 456.5 3 3) returns (trả về giá trị) "38'-0.500""
 with **precision** (với Độchínhxác) = 4 (rtos 456.5 3 4) returns (trả về giá trị) "38'-0.5000""
 with **precision** (với Độchínhxác) = 5 (rtos 456.5 3 5) returns (trả về giá trị) "38'-0.50000""

with **precision** (với Độchínhxác) = 6 (rtos 456.5 3 6) returns (trả về giá trị) "38'-0.500000""

with **precision** (với Độchínhxác) = 7 (rtos 456.5 3 7) returns (trả về giá trị) "38'-0.5000000""

ect.....

mode = 4 = Architectural (Mode = 4 tương ứng với hệ thống đo độ dài trong kiến trúc)

with **precision** (với Độchínhxác) = 1 (rtos 37.7071 4 1) returns (trả về giá trị) "3'-1 1/2""

with **precision** (với Độchínhxác) = 2 (rtos 37.7071 4 2) returns (trả về giá trị) "3'-1 3/4""

with **precision** (với Độchínhxác) = 3 (rtos 37.9 4 3) returns (trả về giá trị) "3'-1 7/8""

with **precision** (với Độchínhxác) = 4 (rtos 37.7071 4 4) returns (trả về giá trị) "3'-1 11/16""

with **precision** (với Độchínhxác) = 5 (rtos 37.7071 4 5) returns (trả về giá trị) "3'-1 23/32""

with **precision** (với Độchínhxác) = 6 (rtos 37.7071 4 6) returns (trả về giá trị) "3'-1 45/64""

with **precision** (với Độchínhxác) = 7 (rtos 37.7071 4 7) returns (trả về giá trị) "3'-1 91/128""

ect.....

mode = 5 = Inch Fractional (Mode = 5 tương ứng với hệ thống đo độ dài theo đơn vị Inch)

with **precision** (với Độchínhxác) = 1 (rtos 37.7071 5 1) returns (trả về giá trị) "37 1/2""

with **precision** (với Độchínhxác) = 2 (rtos 37.7071 5 2) returns (trả về giá trị) "37 3/4""

with **precision** (với Độchínhxác) = 3 (rtos 37.9 5 3) returns (trả về giá trị) "37 7/8""

with **precision** (với Độchínhxác) = 4 (rtos 37.7071 5 4) returns (trả về giá trị) "37 11/16""

with **precision** (với Độchínhxác) = 5 (rtos 37.7071 5 5) returns (trả về giá trị) "37 23/32""

with **precision** (với Độchínhxác) = 6 (rtos 37.7071 5 6) returns (trả về giá trị) "37 45/64""

with **precision** (với Độchínhxác) = 7 (rtos 37.7071 5 7) returns (trả về giá trị) "37 91/128""

ect.....

angtos - This function converts a number to a formatted string representing an angle.

Hàm này đổi một **số** thành một **chuỗi** định dạng trước đại diện cho số đo một góc

Syntax : (**angtos number mode precision**)

Cú pháp: (**angtos Số Mode Độchínhxác**)

number - Any valid number representing radians. [3.14159 radians = pi =180 degrees]

Số - là một **số** có giá trị bất kỳ đại diện cho số đo theo radian

mode - This directly corresponds to AUnits system variable.

Mode – là một **số** đại diện cho mode tương ứng trực tiếp với biến hệ thống đơn vị đo góc

precision - This directly corresponds to the AUPrec system variable.

Độchínhxác – là một **số** đại diện cho mode tương ứng trực tiếp với biến hệ thống độ chính xác của đơn vị đo góc

mode = 0 = Decimal Degrees (**Mode = 0** tương ứng với đơn vị đo góc theo độ thập phân)

with **precision** (**với Độchínhxác**) = 1 (**angtos 0.627102 0 1**) returns (**trả về kết quả**) "35.9"

with **precision** (**với Độchínhxác**) = 2 (**angtos 0.627102 0 2**) returns (**trả về kết quả**) "35.93"

with **precision** (**với Độchínhxác**) = 3 (**angtos 0.627102 0 3**) returns (**trả về kết quả**) "35.930"

with **precision** (**với Độchínhxác**) = 4 (**angtos 0.627102 0 4**) returns (**trả về kết quả**) "35.9303"

with **precision** (**với Độchínhxác**) = 5 (**angtos 0.627102 0 5**) returns (**trả về kết quả**) "35.93030"

ect.....

mode = 1 = Degrees Minutes Seconds (**Mode = 1** tương ứng với đơn vị đo góc theo độ phút giây)

with **precision** (**với Độchínhxác**) = 1 (**angtos 0.627102 1 1**) returns (**trả về kết quả**) " 35d56' "

with **precision** (với Độchínhxác) = 3 (angtos 0.627102 1 3) returns (trả về kết quả) "35d55'49" "

with **precision** (với Độchínhxác) = 5 (angtos 0.627102 1 4) returns (trả về kết quả) "35d55'49.1" "

with **precision** (với Độchínhxác) = 6 (angtos 0.627102 1 5) returns (trả về kết quả) "35d55'49.07" "

with **precision** (với Độchínhxác) = 7 (angtos 0.627102 1 6) returns (trả về kết quả) "35d55'49.073" "

ect.....

mode = 2 = Grads (Mode = 2 tương ứng với đơn vị đo góc theo Grads)

with **precision** (với Độchínhxác) = 1 (angtos 0.627102 2 1) returns (trả về kết quả) "39.9g"

with **precision** (với Độchínhxác) = 2 (angtos 0.627102 2 2) returns (trả về kết quả) "39.92g"

with **precision** (với Độchínhxác) = 3 (angtos 0.627102 2 3) returns (trả về kết quả) "39.923g"

with **precision** (với Độchínhxác) = 4 (angtos 0.627102 2 4) returns (trả về kết quả) "39.9226g"

with **precision** (với Độchínhxác) = 5 (angtos 0.627102 2 5) returns (trả về kết quả) "39.92255g"

ect.....

mode = 3 = Radians (Mode = 3 tương ứng với đơn vị đo góc theo Radians)

with **precision** (với Độchínhxác) = 1 (angtos 0.627102 3 1) returns (trả về kết quả) "0.6r"

with **precision** (với Độchínhxác) = 2 (angtos 0.627102 3 2) returns (trả về kết quả) "0.63r"

with **precision** (với Độchínhxác) = 3 (angtos 0.627102 3 3) returns (trả về kết quả) "0.627r"

with **precision** (với Độchínhxác) = 4 (angtos 0.627102 3 4) returns (trả về kết quả) "0.6271r"

về kết quả) "0.6271r"

with **precision** (với Độchínhxác) = 5 (angtos 0.627102 3 5) returns (trả về kết quả) "0.62710r"

ect.....

mode = 4 = Surveyor (Mode = 4 tương ứng với đơn vị đo góc theo hướng địa cầu)

with **precision** (với Độchínhxác) = 1 (angtos 0.627102 0 1) returns (trả về kết quả) "N 54d4' E"

with **precision** (với Độchínhxác) = 2 (angtos 0.627102 0 2) returns (trả về kết quả) "N 54d4' E"

with **precision** (với Độchínhxác) = 3 (angtos 0.627102 0 3) returns (trả về kết quả) "N 54d4'11" E"

with **precision** (với Độchínhxác) = 4 (angtos 0.627102 0 4) returns (trả về kết quả) "N 54d4'11" E"

with **precision** (với Độchínhxác) = 5 (angtos 0.627102 0 5) returns (trả về kết quả) "N 54d4'10.9" E"

ect.....

Math Functions

Các hàm toán học

; Cái thẳng (<=) mà ta thấy lúc trước cùng một giuộc với mấy thẳng ở đây đây nhé.

; Có rất nhiều hàm toán học có cấu trúc tương tự như một trong các hàm dưới đây

± = / * 1+ 1- cos atan sin sqrt expt [Example Program](#)

+ - Addition. (Hàm cộng +)

Syntax : (+ number number)

Cú pháp: (+ Số Số Số) ; các số phải cách nhau một khoảng trắng đầy nhé.

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số thực hoặc số nguyên)

Returns an integer or a real number.

Giá trị trả về là một số thực hoặc một số nguyên

(+ 3 4) returns (trả về giá trị) 7

(+ 3.5 4.2) returns (trả về giá trị) 7.7

(+ 3 4 5) returns (trả về giá trị) 12

(+ 1.2 1.2 1.2) returns (trả về giá trị) 3.6

(+ "3" "4") returns (trả về giá trị) "Error: Bad Argument Type" ["3" and "4" are strings, not numbers] (“3” và “4” là các chuỗi chứ không phải là số)

- - **Subtraction. (Hàm trừ -)** ; Giờ là lúc ta nấu cháo thẳng này nhé. Khởi lo nó chạy mất.

Syntax : (- number number)

Cú pháp: (- Số Số Số ...) ; Thẳng đầu tiên là số bị trừ, còn lại là số trừ tuốt tuồn tuột.

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

Returns an integer or a real number.

Giá trị trả về là một số nguyên hoặc số thực

(- 4 3) returns (trả về giá trị) 1

(- 4.5 4.2) returns (trả về giá trị) 0.3

(- 9 5 2) returns (trả về giá trị) 2

(- 40.5 10.0 5.2) returns (trả về giá trị) 25.3

(- "3" "4") returns (trả về giá trị) "Error: Bad Argument Type" ["3" and "4" are strings, not numbers] (“3” và “4” là các chuỗi chứ không phải số)

/ - Division. (Hàm chia /)

Syntax : (/ number number)

Cú pháp: (/ Số Số Số ...) ; Chỉ có số đầu tiên là số bị chia, còn lại là các số chia

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số thực hoặc số nguyên)

Returns an integer or a real number.

Giá trị trả về là một số nguyên hoặc số thực

(/ 9 3) returns (trả về giá trị) 3

(/ 5 2) returns (trả về giá trị) 2 [if both numbers are integers then it returns an integer]
(nếu cả hai số đều là số nguyên hàm sẽ trả về giá trị là một số nguyên)

(/ 5 2.0) returns (trả về giá trị) 2.5 [if either number is a real number then it returns a real number]
(nếu một số là số thực thì hàm sẽ trả về giá trị là một số thực)

(/ 5.0 2) returns 2.5 (trả về giá trị) [if either number is a real number then it returns a real number]
(nếu một số là số thực thì hàm sẽ trả về giá trị là một số thực)

(/ 12 2 3) returns (trả về giá trị) 2 [12/2 = 6 then 6/3 = 2] (thực hiện phép chia liên tiếp)

(/ "3" "4") returns (trả về giá trị) "Error: Bad Argument Type" ["3" and "4" are strings, not numbers] (“3” và “4” là các chuỗi chứ không phải số)

* - Multiplication. (Hàm nhân *) ; Đến lượt thằng này xí quách cũng chả còn

Syntax : (* number number)

Cú pháp: (* Số Số Số ...) ; Thực hiện phép nhân liên tiếp với các số

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số thực hoặc số nguyên)

Returns an integer or a real number.

Giá trị trả về là một số nguyên hoặc số thực

(* 9 3) returns (trả về giá trị) 12

(* 5 2) returns (trả về giá trị) 10 [if both numbers are integers then it returns an integer] (nếu cả hai số đều là số nguyên thì giá trị trả về sẽ là một số nguyên)

(* 5 2.0) returns (trả về giá trị) 10.0 [if either number is a real number then it returns a real number] (nếu một số là số thực thì giá trị trả về sẽ là một số thực)

(* 5.0 2) returns (trả về giá trị) 10.0 [if either number is a real number then it returns a real number] (nếu một số là số thực thì giá trị trả về sẽ là một số thực)

(* 2 3 4) returns (trả về giá trị) 24 [2*3 = 6 then 6*4 = 24] (Thực hiện nhân liên tiếp)

(* "3" "4") returns (trả về giá trị) "Error: Bad Argument Type" ["3" and "4" are strings, not numbers] ("3" và "4" là các chuỗi chứ không phải số)

1+ - Returns value increased by one. (**Hàm cộng 1** trả về giá trị số được tăng thêm 1)

Syntax : (1+ number)

Cú pháp: (1+ Số)

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

Returns an integer or a real number.

Giá trị trả về là một số nguyên hoặc số thực

(1+ 3) returns (trả về giá trị) 4

(1+ 5) returns (trả về giá trị) 6 [if the number is an integer then it returns an integer] (nếu số là số nguyên thì giá trị trả về sẽ là một số nguyên)

(1+ 5.0) returns (trả về giá trị) 6.0 [if number is a real number then it returns a real number] (nếu số là số thực thì giá trị trả về sẽ là một số thực)

(1+ "3") returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number]
(“3” là một chuỗi chứ không phải một số)

(1+ 3 4) returns (trả về giá trị) "Error: Too Many Arguments" [Only accepts one number
as argument] (Hàm chỉ chấp nhận một số làm đối số)

1- - Returns value decreased by one. (Hàm trừ 1 trả về giá trị số bị bớt đi 1)

Syntax : (1- number)

Cú pháp: (1- Số)

number - any valid number.[integer or real number]

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

Returns an integer or a real number.

Giá trị trả về là một số nguyên hoặc số thực

(1- 3) returns (trả về giá trị) 2

(1- 5) returns (trả về giá trị) 4 [if the number is an integer then it returns an integer]
(nếu số là số nguyên thì giá trị trả về sẽ là một số nguyên)

(1- 5.0) returns (trả về giá trị) 4.0 [if number is a real number then it returns a real
number] (nếu số là số thực thì giá trị trả về sẽ là một số thực)

(1- "3") returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number]
(“3” là một chuỗi chứ không phải một số)

(1- 3 4) returns (trả về giá trị) "Error: Too Many Arguments" [Only accepts one number
as argument] (Hàm chỉ chấp nhận một số làm đối số)

cos - Returns the cosine of an angle expressed in radians. (Hàm cos trả về giá trị cosin của một góc
biểu diễn dưới dạng radians)

(Note: Radians are AutoCads angular units. A circle = 2*pi or 180 degrees = pi)

(Chú ý: Radians là đơn vị đo góc của AutoCAD. Một vòng tròn = 28pi hay 180 độ = pi)

Syntax : (cos number)

Cú pháp: (cos Số)

number - any valid number.[integer or real number] that represents an angle expressed in radians.

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực) đại diện cho số đo của một góc được tính theo Radians)

Returns a real number.

Giá trị trả về là một số thực

(cos pi) returns (trả về giá trị) -1.0

(cos (+ pi pi)) returns (trả về giá trị) 1.0

(cos 5.0) returns (trả về giá trị) 0.283662

(cos "3") returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] ("3" là một chuỗi chứ không phải một số)

(cos 3 4) returns (trả về giá trị) "Error: Too Many Arguments" [Only accepts one number as argument] (Hàm chỉ chấp nhận một số làm đối số)

atan - Returns the arctangent of a number in radians. (**Hàm atan trả về giá trị arctang của một số theo radians**)

(Note: Radians are AutoCads angular units. A circle = 2*pi or 180 degrees = pi)

(Chú ý: Radians là đơn vị đo góc của AutoCAD. Một vòng tròn = 28pi hay 180 độ = pi)

Syntax : (atan number1optional number2)

Cú pháp: (atan Số Sốđịnhhướng)

number1 - any valid number.[integer or real number].

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

number2 - any valid number.[integer or real number]. This is optional and is usually omitted.

Sốđịnhhướng – là một số có giá trị bất kỳ (số nguyên hoặc số thực). Số định hướng này thường được bỏ qua

Returns a real number representing radians.

Giá trị trả về là một số thực theo Radians

(`atan 1`) returns (trả về giá trị) 0.785398

(`atan -1`) returns (trả về giá trị) -0.785398

(`atan 5.0`) returns (trả về giá trị) 1.3734

(`atan "3"`) returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] (“3” là một chuỗi chứ không phải một số)

(`atan 3 4`) returns (trả về giá trị) 0.643501 [Returns the arctangent of 3 divided by 4] (Trả về giá trị của arctang $\frac{3}{4}$)

(Note: The range of angles returned is $-\pi/2$ to $+\pi/2$ radians.)

(Chú ý: Các góc chỉ được trả về giá trị trong khoảng $-\pi/2$ đến $+\pi/2$)

sin - Returns the sine of an angle expressed in radians. **(Hàm sin trả về giá trị sine của một góc được đo theo radians)**

(Note: Radians are AutoCads angular units. A circle = 2π or 180 degrees = π)

(Chú ý: Radians là đơn vị đo góc của AutoCAD. Một vòng tròn = 2π hay 180 độ = π)

Syntax : (**sin number**)

Cú pháp: (**sin Số**)

number - any valid number.[integer or real number] that represents an angle expressed in radians.

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực) đại diện cho số đo của một góc được tính theo Radians.

Returns a real number.

Giá trị trả về là một số thực

(`sin 1.0`) returns (trả về giá trị) 0.841471

(`sin 0`) returns (trả về giá trị) 0.0

`(sin 5.0)` returns (trả về giá trị) -0.958924

`(sin "3")` returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] (“3” là một chuỗi chứ không phải một số)

`(sin 3 4)` returns (trả về giá trị) "Error: Too Many Arguments" [Only accepts one number as an argument] (Hàm chỉ chấp nhận một số làm đối số)

sqrt - Returns the square root of a number. (Hàm sqrt trả về giá trị căn bậc hai của một số)

Syntax : (**sqrt number**)

Cú pháp: (**sqrt Số**)

number - any valid number.[integer or real number].

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

Returns a real number.

Giá trị trả về là một số thực

`(sqrt 4)` returns (trả về giá trị) 2.0

`(sqrt 0)` returns (trả về giá trị) 0.0

`(sqrt 5.0)` returns (trả về giá trị) 2.23607

`(sqrt "3")` returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] (“3” là một chuỗi chứ không phải một số)

`(sqrt 3 4)` returns (trả về giá trị) "Error: Too Many Arguments" [Only accepts one number as argument] (Hàm chỉ chấp nhận một số làm đối số)

expt - Returns a number raised to a power. (Hàm expt trả về giá trị một số được nâng lũy thừa)

Syntax : (**expt number power**)

Cú pháp: (**expt Số Số mũ**)

number - any valid number.[integer or real number].

Số - là một số có giá trị bất kỳ (số nguyên hoặc số thực)

power - any valid number.[integer or real number].

Sômũ – là một số có giá trị bất kỳ (số nguyên hoặc số thực)

Returns a real number unless both number and power are integers, then it returns an integer..

Giá trị trả về là một **số thực** trừ phi cả **Số** và **Sômũ** đều là số nguyên, khi đó giá trị trả về sẽ là một số nguyên

(expt 2 2) returns (trả về giá trị) 4

(expt 2 2.0) returns (trả về giá trị) 4.0

(expt 5 2) returns (trả về giá trị) 25

(expt "3" 2) returns (trả về giá trị) "Error: Bad Argument Type" ["3" is a string, not a number] ("3" là một chuỗi chứ không phải số)

Example Program 1:

Chương trình mẫu 1: ; Ồi giờì ời! Toàn đồ cổ! Cố mà ôm.

```
(defun C:myProg()

  (setq intAge(getint "\n Enter your Age: "))

  (setq intDays(* intAge 365))

  (setq intWeeks(/ intDays 7))

  (princ (strcat "\n You are " (itoa intDays) " Days old!"))

  (princ (strcat "\n You are " (itoa intWeeks) " Weeks old!"))

  (princ)

)
```

Execution: Thử tí, vui đáo để!

Command: (load "myProg")<enter>

Command: myProg<enter>

Command: Enter your Age: 39<enter>

Command: You are 14235 Days old!

Command: You are 2033 Weeks old!

Example Program 2:

Chương trình mẫu 2: ; Có tí đuôi mới đấy, khoái thì nhào vô.

```
(defun C:Triangle()

  (setq pt1(getpoint "\n Pick First Point: "))

  (setq pt2(getpoint "\n Pick Last Point: "))

  (setq x1(car pt1))    ;x-coord of point one
  (setq x2(car pt2))    ;x-coord of point two
  (setq y1(cadr pt1))   ;y-coord of point one
  (setq y2(cadr pt2))   ;y-coord of point two

  (setq xdis(- x2 x1)) ;distance in x direction between points
  (setq ydis(- y2 y1)) ;distance in y direction between points

  (setq slpdis(sqrt(+ (expt xdis 2.0)(expt ydis 2.0))) ;Asq + Bsq =
    Csq ;(Tại ông Pitagor bảo thế! Vậy nên biến slpdis là khoảng cách
    ;giữa hai điểm pt1 và pt2)

  (princ (strcat "\n Distance = " (rtos slpdis 4 4)))

  ;;; Hiện thị khoảng cách giữa hai điểm sau các ký tự Distance = theo
  ;kiểu 4 (kiến trúc) với độ chính xác 4

  (princ)

)
```

Execution: Chơi đi, khoái không?

Command: (load "Triangle")<enter>

Command: Triangle<enter>

Command: Pick First Point:<pick>

Command: Pick Last Point:<pick>

Command: Distance = 3'-4 1/2"

Command:

Selecting Entities

Các hàm lựa chọn đối tượng

Functions - [entsel](#) [ssget](#) [Example Programs](#)

entsel - This function prompts for the user to pick one entity.

Hàm này nhắc người sử dụng chọn **một đối tượng** trong AutoCad

Syntax : (**entsel**)

Cú pháp: (**entsel**) ; hàm này không có các đối số kèm theo

Returns the entity's name and the point selected in a list.

Giá trị trả về là **một danh sách** gồm tên của đối tượng và tọa độ của điểm chọn

Example (**Ví dụ**): (<Entity name: 3b40a08> (230.666 285.862 0.0))

(**car**(**entsel**)) returns the entity name if entsel does not return nil.. (trả về giá trị tên của đối tượng nếu hàm entsel không trả về nil)

(**cadr**(**entsel**)) returns the point selected if entsel does not return nil.. (trả về danh sách tọa độ của điểm chọn nếu hàm entsel không trả về giá trị nil)

Note: Selecting an area in the drawing where there are no entities would return nil. You will get an error if you try to get the car or cadr of nil as shown above. The best thing to do is to save the return value of the entsel function, then check to see if it is a valid selection.

Chú ý: Việc lựa chọn một vùng trên bản vẽ không có đối tượng nào sẽ trả về kết quả nil. Bạn sẽ gặp lỗi nếu bạn cố dùng hàm car hay cadr với đối số nil như đã nói ở phần trên. Tốt nhất bạn nên lưu lại giá trị trả về của hàm entsel, sau đó kiểm tra lại xem nó có phải là một lựa chọn có giá trị hay không như chỉ ra trong mẫu dưới đây:

Example: Mẫu:

```
(setq myEnt (entsel)) ; save the return value of the entsel function (sử dụng hàm setq để lưu lại giá trị trả về của hàm entsel như một biến myEnt)

(if myEnt          ; if the value of myEnt is not nil then do some stuff (nếu giá trị của biến myEnt khác nil thì bạn sẽ làm vài cái lằng nhằng gì đó mà bạn thích)

    (progn          ; use the progn statement here for multiple statements (sử dụng thông báo progn cho phép bạn cứ việc lằng nhằng thoải mái theo kiểu Autolisp)

        ;do some stuff          ; put your code here (bạn sẽ đặt những cái lằng nhằng mà bạn thích làm vào đây theo kiểu của Autolisp)

    )                ; close the progn statement here (kết thúc thông báo progn)

    (alert "Nothing Selected.") ; alert the user (cảnh báo bạn khi biến myEnt nhận giá trị nil)

)                    ; close the if statement (kết thúc thông báo về hàm if)
```

; Cái nàg if này còn nhiều thứ đáng yêu lắm. Ta sẽ có dịp tán tỉnh nàg ở phần sau. Giờ thì cứ để ; nàg làm duyên một chút.

ssget - This function prompts the user to select entities.

Hàm này nhắc người sử dụng lựa chọn các đối tượng trong AutoCad. Có nhiều cách sử dụng hàm này để lựa chọn các đối tượng tùy theo cú pháp mà bạn sử dụng

Syntax : (**ssget**) ;prompts the user for a selection set.

Cú pháp: (**ssget**) ; nhắc người sử dụng lựa chọn nhiều đối tượng thành bộ lựa chọn

Syntax : (**ssget mode**) ;prompts the user for a selection set and sets mode to "W", "C", "L",

and "P", corresponding to the Window, Crossing, Last, and Previous selection methods. Another optional mode value is "X", which selects the entire database. There are other modes, but this list contains the modes that will be used most often.

Cú pháp: (**ssget Model****lựa chọn**) ; nhắc người sử dụng lựa chọn nhiều đối tượng theo các mode lựa chọn “W”, “C”, “L”, “P” tương ứng với các phương pháp lựa chọn window, crossing, last, Previous, trong AutoCad. Một mode lựa chọn khác là “X” cho phép lựa chọn toàn bộ cơ sở dữ liệu làm đối tượng. Còn nhiều mode lựa chọn khác nhưng các mode trên là các mode được sử dụng thường xuyên nhất.

Syntax : (**ssget "W"**) ;prompts the user for a selection set using the Window mode.

Cú pháp: (**ssget “W”**) ;nhắc bạn sử dụng phương pháp lựa chọn Window để lựa chọn

Syntax : (**ssget "C"**) ;prompts the user for a selection set using the Crossing mode.

Cú pháp: (**ssget “C”**) ;nhắc bạn sử dụng phương pháp lựa chọn Crossing để lựa chọn

Syntax : (**ssget "L"**) ;Selects the last entity selected.

Cú pháp: (**ssget “L”**) ; Lựa chọn đối tượng được lựa chọn cuối cùng

Syntax : (**ssget "P"**) ;Selects the previously selected entities.

Cú pháp: (**ssget “P”**) ; Lựa chọn các đối tượng vừa được chọn trước đó

Syntax : (**ssget "X"**) ;Selects all entities in the drawing.

Cú pháp: (**ssget “X”**) ; Lựa chọn tất cả các đối tượng có trong bản vẽ

Syntax : (**ssget filter_List**) or (**ssget mode filter_List**)

Cú pháp: (**ssget DanhsáchLọc**) hay (**ssget Mode DanhsáchLọc**)

filter_List - an association list containing valid filters for entities corresponding to the Entity DXF Group Codes. What did I just say? Entity DXF Group Codes? What the heck is that? Click [here](#) to find out.

DanhsáchLọc – là một danh sách tương tác chứa các yếu tố lọc có giá trị đối với các đối tượng tương ứng với các mã nhóm DXF của các đối tượng. Nếu bạn muốn biết các mã nhóm DXF của đối tượng là cái gì quý giá gì, hãy xem lại phần trước

Returns a selection set of entity names.

Giá trị trả về là một bộ lựa chọn gồm các tên của các đối tượng

You can filter out entities that do not match your criteria using the filter_List. If you only wanted entities that were on layer "STR" then you could add a filter list that looks like this:

Bạn có thể lọc ra các đối tượng không phù hợp với các yếu tố lọc cụ thể được sử dụng trong Danh sách Lọc. Giả sử bạn chỉ muốn có các đối tượng thuộc lớp "STR", vậy bạn phải bổ sung một danh sách lọc như sau:

```
(setq myFilter(list (cons 8 "STR"))) ;returns (trả về giá trị) ((8 .  
"STR")) (Một danh sách tương tác)
```

; Cần chú ý tới chức năng đặc biệt tạo danh sách tương tác của hàm cons khi đối số thứ hai không phải là một danh sách. Ở đây đặt biến myFilter thành một danh sách chứa danh sách tương tác

```
(ssget "X" myFilter) ;returns a selection set containing only entities on layer  
"STR" (trả về bộ lựa chọn chỉ có các đối tượng trên lớp "STR")
```

or (hoặc)

```
(setq myFilter(list (cons 0 "LINE") (cons 8 "STR")))
```

```
;returns (trả về giá trị) ((0 . "LINE") (8 . "STR"))
```

```
(ssget "X" myFilter)
```

```
;returns a selection set containing only line entities on layer "STR" (trả về bộ lựa chọn  
chỉ có các đối tượng là đoạn thẳng trên lớp "STR")
```

End of Selecting Entities

Kết thúc các hàm lựa chọn các đối tượng.

Example Program 1:

Chương trình mẫu 1:

```
(defun C:myProg())
```

```
(if (setq myEnt(entset)) ;start the if statement (mở thông báo if)
```

; Nên tách thành hai đoạn mã như sau: (setq myEnt (entset)) rồi (if myEnt

```

(progn ;going to have multiple statements (thông báo có nhiều việc cần thực hiện)

  (setq pt1(getpoint "\n Base Point: ")) ;store base point (lưu lại pt1)

  (setq pt2(getpoint "\n Displacement Point: ")) ;store displacement
    point (lưu lại pt2)

  (command "move" (car myEnt) "" pt1 pt2) ;move entity (dùng lệnh “move”
    nội trú trong AutoCad để di chuyển đối tượng myEnt theo vector từ pt1 đến pt2)

) ;close the progn (đóng thông báo progn)

(alert "Please select an entity!") ;else alert the user of an
  error (cảnh báo người sử dụng nếu chưa chọn đối tượng)

) ;close the if statement (đóng thông báo if)

(princ) ;clean exit (supresses echo)

) ;close the program

```

Execution: Thực hiện trong AutoCad:

```

Command: (load "myProg")<enter>

Command: myProg<enter>

Command: Select Object: <pick>

Command: Base Point: <pick>

Command: Displacement Point: <pick>

Command:

```

Example Program 2:

```

(defun C:myProg2()

  (if (setq mySet(ssget "X" (list (cons 8 "STR")(cons 0 "CIRCLE"))))
    ;get set (Đặt biến mySet là bộ lựa chọn các vòng tròn trên lớp “STR”)

    (progn ;going to have multiple statements

```

```

(setq pt1(getpoint "\n Base Point: ")) ;store base point

(setq pt2(getpoint "\n Displacement Point: "));store displacement
point

(command "move" mySet "" pt1 pt2) ;move all entities (di chuyển tất cả
bộ lựa chọn theo vector từ pt1 đến pt2)

) ;close the progrn

(alert "No entites Match Criteria!") ;else alert the user of an
error

) ;close the if statement

(princ) ;clean exit (supresses echo)

) ;close the program

```

Execution:Thực hành trong AutoCad

```

Command: (load "myProg2")<enter>

Command: myProg2<enter>

Command: Base Point: <pick>

Command: Displacement Point: <pick>

Command:

```

Selection Set Functions

Các hàm xử lý bộ lựa chọn

Functions - [ssadd](#) [ssdel](#) [sslength](#) [ssname](#) [Example Programs](#)

ssadd - This function does one of three things depending on how many arguments you send with it.

Hàm này trả về một trong ba kết quả tùy theo số đối số mà bạn sử dụng

If you use the ssadd function with no arguments it will return a new selection set with no members.

Khi bạn sử dụng hàm ssadd không có đối số, hàm sẽ trả về một bộ lựa chọn mới không có đối tượng thành viên nào

Example : (Ví dụ:) `(setq myNewSet(ssadd))`

If you use ssadd with a name of an entity it will return a new selection set with only that one member. (Khi bạn sử dụng hàm ssadd với một tên đối tượng làm đối số, hàm sẽ trả về một bộ lựa chọn mới chỉ có duy nhất một đối tượng thành viên đó)

Example : (Ví dụ:) `(setq myNewSet(ssadd entityName))`

If you use ssadd with a name of an entity and name of a selection set, it will add that entity to the set. (Khi bạn sử dụng hàm ssadd với một tên đối tượng và tên của một bộ lựa chọn làm đối số, hàm sẽ thêm đối tượng đó vào bộ lựa chọn)

Example : (Ví dụ:) `(setq myOldSet(ssadd entityName selectionSetName))`

And return the selection set with the entity added to the set. (Và trả về bộ lựa chọn mới đã được thêm đối tượng đó vào)

Syntax : **(ssadd) (ssadd entityName) (ssadd entityName SelectionSetName)**

Cú pháp: **(ssadd)** hay **(ssadd TênĐối tượng)** hoặc **(ssadd TênĐối tượng TênBộlựa chọn)**

Where **entityName** is a valid entity name returned by ssnames or entsel.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi hàm ssnames hoặc hàm entsel
(;Cái thằng ssnames này lại do vỡ kế hoạch rồi, ra sớm quá làm khổ bà con, ta sẽ trị nó ở phần sau nhé)

Where **selectionSetName** is a valid name of a selection set returned by setq.

TênBộlựa chọn – là một tên có giá trị của bộ lựa chọn được trả về bởi hàm setq

Returns the selection set with the new entity added to it.

Giá trị trả về là một **bộ lựa chọn** với các đối tượng mới trong nó.

Example: (Ví dụ:) (<Selection set: 1>

Example: Let's build a function that allows me to add items to a selection set until I stop.

Ví dụ: Hãy tạo một hàm cho phép thêm các khoản mục vào một bộ lựa chọn cho đến lúc dừng lặp

```

(defun C:BldSet()                ;defin a program name

  (setq mySet(ssadd))            ;build an empty selection set (Tạo một bộ lựa chọn rỗng)

  (while (/= (setq enS(entsel)) nil) ;loop until I pick nothing or press enter.
    (lặp cho tới khi không chọn đối tượng nào nữa hoặc nhấn Enter)

    ; Hàm (/= ... ..) áp dụng cho các biến phi số học với nghĩa là khác nhau

    (setq en(car enS))           ;get the entity name of the selected item (lấy giá trị tên đối
    tượng của đối tượng đã chọn)

    (setq mySet(ssadd en mySet)) ;add the entity to the selection set (thêm tên
    đối tượng vào bộ lựa chọn mySet)

  ) ; Kết thúc hàm while

) ; Kết thúc hàm defun

```

Now at the Command prompt type: (Bây giờ nhập vào dòng nhắc lệnh của AutoCad)

Command: **move**<enter>

Command: **!mySet**<enter> ;don't forget the exclamation point in front of the variable name.(đừng quên dấu chấm than phía trước tên biến mySet)

Did everything get selected to be moved? Cool. Move individual items around and try it again. Mọi thứ đã được chọn để di chuyển rồi chứ? Tốt. Hãy di chuyển từng thẳng lòng vòng một chút và thử lại lần nữa.

Did everything get selected again? Cool. Why? Because AutoLisp finds them by their entity name, not their location. (Mọi thứ vẫn được chọn lại đây chứ? Tốt. Tại sao vậy? À, bởi rằng Autolisp tìm kiếm chúng theo tên đối tượng chứ không phải theo vị trí của chúng.

ssdel - This function does the opposite of ssadd. SSDEL removes an entity from a selection set.

Hàm này thực hiện ngược lại với hàm ssadd. Hàm ssdel xóa đi một đối tượng trong bộ lựa chọn

Syntax : (ssdel **entityName** **selectionSetName**)

Cú pháp: (ssdel **TênĐối tượng** **TênBộlựa chọn**)

Where **entityName** is a valid entity name returned by sselect or entsel.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi hàm sname hay entsel

Where **selectionSetName** is a valid name of a selection set returned by setq.

TênBộlựachọn – là một tên có giá trị của bộ lựa chọn được trả về bởi hàm setq

Returns a new selection set without the entity.

Giá trị trả về - là một bộ lựa chọn mới không có đối tượng bị điểm mặt làm đối số.

Assuming en1 is a valid entity and en2 is not. (Giả sử en1 là đối tượng có giá trị, còn en2 không phải)

Assuming ss1 is a valid selection set and ss2 is not. (ss1 là bộ lựa chọn có giá trị còn ss2 không phải)

Assuming en3 is a valid entity name but is not part of any selection set. (en3 là đối tượng có giá trị nhưng không thuộc bất kỳ bộ lựa chọn nào.)

(ssdel en1 ss1) ;returns selection set without entity en. (trả về bộ lựa chọn rỗng)

(ssdel en1 ss2) ;returns ERROR - Bad Argument Type (trả về lỗi Bad Argument)

(ssdel en2 ss1) ;returns ERROR - Bad Argument Type (trả về lỗi Bad Argument)

(ssdel en2 ss2) ;returns ERROR - Bad Argument Type (trả về lỗi Bad Argument)

(ssdel en3 ss1) ;returns nil (trả về nil)

Note: If you filter correctly using the **ssget "X"** function, you shouldn't need ssdel.

Chú ý rằng: Nếu bạn sử dụng tốt hàm **ssget "X"**, bạn sẽ không cần tới hàm **ssdel**

sslength - This function returns the number of entities inside a selection set.

Hàm này trả về **một số** là số lượng các đối tượng trong một bộ lựa chọn

Syntax : (sslength selectionSetName)

Cú pháp: (sslength TênBộlựachọn)

Where **selectionSetName** is a valid name of a selection set returned by setq.

TênBộlựachọn – là một tên có giá trị của một bộ lựa chọn được trả về bởi hàm setq

Returns a real number if the number is greater than 32,767, otherwise it returns an integer.

The number, real or integer, represents the quantity of items inside the selection set.

Giá trị trả về là một số thực nếu nó lớn hơn 32.767 hoặc **số nguyên** nếu nó nhỏ hơn 32.767, đại diện cho số lượng các đối tượng có trong bộ lựa chọn

Assuming ss1 is a valid selection set containing 15 entities. (Giả sử ss1 là bộ lựa chọn có giá trị chứa 15 đối tượng)

Assuming ss2 is a valid selection set containing 1 entity. (Giả sử ss2 là bộ lựa chọn có giá trị chứa 1 đối tượng)

Assuming ss3 is a valid selection set containing no entities. (Giả sử ss3 là bộ lựa chọn có giá trị không chứa đối tượng)

Assuming ss4 is a valid selection set containing 40,000 entities. (Giả sử ss4 là bộ lựa chọn có giá trị chứa 40.000 đối tượng)

Assuming ss5 is not a valid selection set. (Giả sử ss5 là bộ lựa chọn không có giá trị)

`(sslength ss1)` ;returns 15 as an integer (trả về giá trị số nguyên 15)

`(sslength ss2)` ;returns 1 as an integer (trả về giá trị số nguyên 1)

`(sslength ss3)` ;returns 0 as an integer (trả về giá trị số nguyên 0)

`(sslength ss4)` ;returns 40000.0 as a real number (trả về giá trị số thực 40.000,0)

`(sslength ss5)` ;returns ERROR - Bad Argument Type (trả về lỗi Bad Argument)

ssname - This function returns an entity name contained inside a selection set.

Hàm này trả về một **tên đối tượng** chứa trong một bộ lựa chọn

Syntax : (ssname selectionSetName nthItem)

Cú pháp: (ssname TênBộlựa chọn Sốthứ tự)

Where **selectionSetName** is a valid name of a selection set returned by setq.

TênBộlựa chọn – là một tên có giá trị của bộ lựa chọn được trả về bởi hàm setq

Where **nthItem** is an integer representing an index into the set.

Số thứ tự - là một số nguyên đại diện cho thứ tự của đối tượng trong bộ lựa chọn

Note: **nthItem** needs to be a real if the set contains more than 32,767 items.

Chú ý: **Số thứ tự** phải là số thực nếu số lượng đối tượng có trong bộ lựa chọn lớn hơn 32.767

Returns an entity's name on success and nil on error.

Giá trị trả về là một tên đối tượng nếu tìm được hoặc nil nếu như không tìm được đối tượng

Assuming ss1 is a valid selection set containing 15 entities. (ss1 là bộ lựa chọn có giá trị chứa 15 đối tượng)

Assuming ss2 is a valid selection set containing 40,000 entities. (ss2 là bộ lựa chọn có giá trị chứa 40.000 đối tượng)

Assuming ss3 is not a valid selection set. (ss3 là bộ lựa chọn không có giá trị)

(ssname ss1 0) ;returns the first entity in the selection set. (trả về tên đối tượng đầu tiên của bộ lựa chọn)

(ssname ss1 3) ;returns the fourth entity. (trả về tên đối tượng thứ tư của bộ lựa chọn)

(ssname ss1 24) ;returns nil (trả về nil)

(ssname ss1 -3) ;returns nil (trả về nil)

(ssname ss2 34555) ;returns ERROR - Bad Argument Type (needs real) (Trả về lỗi Bad Argument do ở đây cần một số thực chứ không phải số nguyên)

(ssname ss2 34555.0) ;returns the 34555th entity (trả về tên đối tượng thứ 3455 của bộ lựa chọn)

; Chú ý tới cách xác định số thứ tự của đối tượng khi sử dụng số thực khác với khi sử dụng số nguyên

End of Selection Set Functions

Kết thúc các hàm xử lý bộ lựa chọn

Example Program 1:

Chương trình mẫu 1:

I will write a program that prints the layer name of every entity in a selection set.

Ta sẽ viết một chương trình để in tên lớp của mỗi một đối tượng trong một bộ lựa chọn

```
(defun C:pLay() ;define a program name

  (if (setq eset(ssget) ;use the ssget function to select entities (dùng hàm ssget để chọn
    đối tượng) (dùng hàm setq để lưu lại đối tượng được chọn)

    (progn ;use progn since there will be more than 1 statement

      (setq cntr 0) ;set the cntr to the first item in the set (đặt biến cntr bằng 0)

      (while (< cntr (sslength eset)) ;while cntr is less than the length of the set
```

;note: the length is one more than the index of items since the first item is zero. In other words, to get to the first item in a selection set consisting of one item you would use (ssname eset 0) not (ssname eset 1).

; Chú ý vị trí của đối tượng trong bộ lựa chọn lớn hơn số thứ tự của đối tượng là 1. Nói cách khác để có tên đối tượng đầu tiên trong bộ lựa chọn có một đối tượng bạn phải dùng (ssname eset 0) chứ không phải (ssname eset 1)

```
    (setq en(ssname eset cntr)) ;get the entity name of the item indexed with
    cntr (đặt biến en là tên của đối tượng thứ cntr trong bộ lựa chọn eset)

    (setq enlist(entget en)) ;get the dxf group codes of the entity (đặt biến
    enlist là danh sách các mã nhóm DXF của đối tượng tên en được chọn)

    (setq layName(cdr(assoc 8 enlist))) ;get the layer name (đặt biến
    layName là tên lớp được lấy từ danh sách các mã nhóm DXF enlist)

    (princ "\n ") ;print "\n " will cause a new line to be printed (hàm princ “/n” sẽ tạo
    ra một dòng mới để hiển thị tiếp theo)

    (princ layName) ;print the layer name to the command line (hiển thị tên lớp lấy được)

    (setq cntr(+ cntr 1)) ;increment the counter (tăng biến cntr lên 1 đơn vị)
```

```
    ) ;close the while statement (đóng hàm while)

) ;close the progn on the if statement (đóng thông báo progn của hàm if)

(princ "\n Error - No entities selected.") ;print a message on the else
statement (hiển thị thông báo khi hàm if nhận giá trị else)

; note: the if statement can be a " if then " statement or a " if then else" statement

; chú ý là hàm if có thể là "if ...then..." hoặc "if ... then... else then ..."

) ;close the if statement (đóng hàm if)

) ;close the program (đóng chương trình)
```

Note: I personally do not find any of these functions useful except for the sname and sslength functions.

Chú ý: Cá nhân tôi thấy các hàm trên không đặc dụng lắm trừ hàm sname và hàm sslength

If you use your ssget "X" filters well enough there is no reason for the ssadd or ssdel statements.

Nếu bạn sử dụng thành thạo hàm ssget "X" thì chẳng cần các hàm ssadd và ssdel.

There are other selection set functions that are even less useful such as ssmemb, sssetfirst, & ssgetfirst.

Có các hàm xử lý bộ lựa chọn khác như ssmemb, sssetfirst và ssgetfirst nhưng rất ít tác dụng.

When you loop through your selection set you can check an entities DXF group codes and decide whether the entity is one to either skip or work on. So, you can avoid all of these useless functions. Don't get me wrong. I'm not saying they are completely useless and you shouldn't use them. I'm saying why bother.

Khi bạn lặp thông qua bộ lựa chọn của bạn, bạn có thể kiểm soát các mã nhóm DXF của các đối tượng và quyết định xem mã nào thì bỏ qua và mã nào bạn sẽ làm việc với nó. Vì thế bạn có thể tránh tất cả các hàm vô dụng đó. Đừng cho rằng tôi sai. Tôi không bảo rằng chúng hoàn toàn vô dụng và bạn không cần sử dụng chúng. Tôi chỉ nói vì sao không ưa chúng.

Entity Functions

Các hàm xử lý đối tượng

[entget](#) [entlast](#) [entnext](#) [entdel](#) [entmod](#) [entupd](#) [Example Program](#)

entget - This function returns the DXF Group Codes of the entity.

Hàm này trả về các mã nhóm DXF của đối tượng

Syntax : (**entget** **ename**)

Cú pháp: (**entget** **TênĐối tượng**)

ename - a valid entity name returned by entsel, sselect, entlast, or the entnext function.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi các hàm entsel, sselect, entlast hay entnext

Returns an Entity's DXF Group Codes as an association list.

Giá trị trả về - là một danh sách tương tác gồm các mã nhóm DXF của đối tượng

(setq en (car (entsel))) returns entity name as eg. (trả về tên đối tượng như ví dụ)
<Entity name: 3b40a60>

(entget en) returns: (trả về)

((-1 . <Entity name: 3b40a60>) (0 . "LINE") (5 . "FC") (100 . "AcDbEntity") (67 . 0) (8 . "DIM")
(100 . "AcDbLine") (10 252.578 68.25 0.0) (11 188.379 31.5 0.0) (210 0.0 0.0 1.0))

entlast - This function returns the name of the last non-deleted entity in the drawing.

Hàm này trả về đối tượng sau cùng chưa bị xóa trên bản vẽ

Syntax : (**entlast**)

Cú pháp: (**entlast**)

This function is mainly used to get the last entity added to the drawing. (Hàm này chủ yếu được sử dụng để đạt được đối tượng cuối cùng được thêm vào bản vẽ)

Returns an entity name.

Giá trị trả về - là một tên đối tượng

(entlast) returns (trả về giá trị) <Entity name: 3b40a60>

entnext - Let's look at AutoDesk's explanation: If entnext is called with no arguments, it returns the entity name of the first nondeleted entity in the database. If entnext is called with an entity name argument ename, it returns the entity name of the first nondeleted entity following ename in the database. If there is no next entity in the database, it returns nil. The entnext function returns both main entities and subentities.

Ta hãy xem sự giải thích của Autodesk: Nếu hàm entnext được gọi không có đối số, nó sẽ trả về tên đối tượng của đối tượng chưa bị xóa đầu tiên trong cơ sở dữ liệu (bản vẽ). Nếu hàm entnext được gọi với một đối số tên đối tượng, nó sẽ trả về tên của đối tượng đầu tiên chưa bị xóa tiếp theo đối tượng có tên bị lấy làm đối số trong bản vẽ. Nếu trong bản vẽ không có đối tượng kế tiếp nó sẽ trả về giá trị nil. Hàm entnext trả về cả hai loại đối tượng chính và đối tượng phụ.

The entities selected by sset are main entities, not attributes of blocks or vertices of polylines. You can access the internal structure of these complex entities by walking through the subentities with entnext. Once you obtain a subentity's name, you can operate on it like any other entity. If you obtain the name of a subentity with entnext, you can find the parent entity by walking forward with entnext until a seqend entity is found, then extracting the -2 group from that entity, which is the main entity's name.

Các đối tượng được chọn bởi hàm sset là các đối tượng chính, không có các thuộc tính của block hay các đỉnh của polylines. Bạn có thể đạt tới cấu trúc bên trong của các đối tượng phức tạp này bằng cách đi dọc theo các đối tượng phụ của nó nhờ hàm entnext. Mỗi lần bạn có được tên của đối tượng phụ, bạn có thể kích hoạt nó giống như bất kỳ một đối tượng nào khác. Khi bạn có được tên của đối tượng phụ nhờ hàm entnext, bạn có thể tìm được đối tượng gốc bằng cách tiếp tục dùng hàm entnext đi tới cho tới khi tìm được một đối tượng seqend, sau đó trích xuất nhóm thứ hai từ đối tượng này tên của đối tượng chính (gốc)

Syntax : (entnext) or (entnext ename)

Cú pháp: (entnext) hoặc (entnext TênĐối tượng)

ename - a valid entity name returned by entsel, sname, entlast, or the entnext function.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi các hàm entsel, sname, entlast, hay entnext

(entnext) returns the name of the first entity in the drawing. (trả về giá trị tên đối tượng của đối tượng đầu tiên trong bản vẽ)

`(entnext ename)` returns the name of the next entity following ename. (trả về tên của đối tượng tiếp theo sau đối tượng ename)

entdel - This function deletes an entity from the drawing if it exist. If it does not exist and has been previously deleted, it will undelete the entity.

Hàm này xóa một đối tượng khỏi bản vẽ nếu nó tồn tại. Nếu nó không tồn tại và bị xóa trước đó, hàm sẽ không xóa đối tượng đó. (phục hồi trả lại đối tượng đã bị xóa)

Syntax : **(entdel ename)**

Cú pháp: **(entdel TênĐối tượng)**

ename - a valid entity name returned by entsel, sname, entlast, or the entnext function.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi các hàm entsel, sname, entlast hay entnext

Returns an entity name.

Giá trị trả về - là một tên đối tượng

`(setq en (car (entsel)))` returns the name of selected entity. (đặt biến en là tên của đối tượng được chọn bởi hàm entsel)

`(entdel en)` deletes the entity and returns the name of the deleted entity. (Xóa đối tượng en và trả về tên của đối tượng bị xóa en)

`(entdel en)` undeletes the entity and returns the name of the undeleted entity. (Phục hồi đối tượng en và trả về giá trị tên của đối tượng được phục hồi en)

entmod - Modifies an entity in the AutoCAD database.

Hàm này sửa đổi một đối tượng trong bản vẽ AutoCad

Syntax : **(entmod elist)**

Cú pháp: **(entmod Danh sáchMã)**

elist - a list that is returned from an entget statement. (DXF Group Codes of an entity)

Danh sách Mã – là một danh sách các mã được trả về bởi hàm entget (Các mã nhóm DXF của một đối tượng)

Returns an elist if successful otherwise it returns nil.

Giá trị trả về - là một **danh sách mã** nếu có và nếu không có hàm sẽ trả về giá trị **nil**

`(setq en(car(entget)))` returns the name of selected entity. (đặt biến en là tên của đối tượng được chọn bởi hàm entget)

`(setq elist(entget en))` returns the DXF Group Codes of the entity. (Trả về các mã nhóm DXF của đối tượng được chọn)

`(setq elist(subst (cons 8 "DIM") (assoc 8 elist) elist))`

The line above Replaces existing layer name in the elist with new layer name "DIM"

Dòng mã lệnh này để thay thế tên lớp hiện tại của đối tượng bằng tên lớp mới “DIM”

; Ở đây sử dụng hàm subst mà tôi chưa nói được. Ai biết chỉ giùm chỗ nói với. Tạm thời tôi cứ hiểu đại khái như cụ Jeff giải thích ở trên vậy.

`(entmod elist)` updates the AutoCAD database and replaces the old layer name with the new one. (Cập nhật bản vẽ với sự thay thế tên lớp cũ bằng tên lớp mới)

entupd - Updates the screen image of an entity.

Hàm này cập nhật hình ảnh trên màn hình của một đối tượng

Syntax : **(entupd ename)**

Cú pháp: **(entupd TênĐối tượng)**

ename - a valid entity name returned by entget, ssnames, entlast, or the entnext function.

TênĐối tượng – là một tên đối tượng có giá trị được trả về bởi các hàm entget, ssnames, entlast, hay entnext

Returns an entity name.

Giá trị trả về - là một **tên đối tượng**

`(setq en(car (entget)))` returns an entity name. (trả về giá trị một tên đối tượng)

(entupd ename) refreshes the entity on the graphics screen. (Vẽ lại đối tượng trên bản vẽ)

Note: If you modify the DXF Group Codes of an entity, such as changing the layer (8 . "LAYERName"), you will not notice the change until you regenerate the entire drawing or use entupd to regenerate the entity.

Chú ý: Khi bạn sửa đổi các mã nhóm DXF của một đối tượng, chẳng hạn như thay đổi lớp (8 . "TênLớp"), bạn sẽ không nhận thấy sự thay đổi trên bản vẽ cho tới khi bạn tái tạo lại toàn bộ bản vẽ hoặc dùng hàm entupd để tái tạo lại đối tượng đó.

End of Entity Functions

Kết thúc các hàm xử lý đối tượng

Example Program 1:

Chương trình mẫu 1:

Let's change an entity's layer to a layer named "DIM".

Ta sẽ thay đổi lớp của một đối tượng thành lớp mang tên "DIM"

```
(defun C:CLay()
  (if (setq ent (entsel))
    (progn
      (setq en (car ent))
      (setq enlist (entget en))
      (setq enlist (subst (cons 8 "Dim") (assoc 8 enlist) enlist))
      (entmod enlist)
      (entupd en)
    )
    (alert "Error - Select an entity please.")
  )
)
```

```
)  
  
(princ)  
  
)
```

Execution: Thực hành trong AutoCad:

Command: (load "CLay")<enter>

Command: CLay<enter>

Command: Select Object:<pick>

Command:

Read/Write & File Functions

Các hàm xử lý đọc/viết tập tin

Functions - open/close read-line write-line [Example Programs](#)

open/close - Open a file and close a file. That simple.

Hàm này để mở và đóng một tập tin. Đơn giản vậy thôi

Syntax : (open "filename" mode) (close fileVarName)

Cú Pháp: (open “TênTậptin” Mode) và (close TênBiếnTậptin)

Where **filename** is any valid file or resource name.

TênTậptin – là tên tập tin hay nguồn có giá trị

Where **mode** is either "r" , "w" , or "a".

Mode – là các mode hoặc “r”, hoặc “w” hoặc “a”

"r" - Read mode. Reads a file with either read-char or read-line. (Mode đọc để đọc một tập tin nhờ đọc các ký tự hay đọc các dòng)

"w" - Write mode. Writes to a file with either write-char, write-line, princ, or print.
(Mode viết dùng để viết một tập tin nhờ viết các ký tự, viết các dòng, hiển thị hay in ra)

"a" - Append mode. Appends to an existing file using write-char, write-line, princ, or print. (mode thêm vào dùng để thêm vào một tập tin có sẵn sử dụng cách viết ký tự, viết các dòng, hiển thị hay in ra)

Note: If the existing file is not found in append mode, autolisp will create a new empty file.

Chú ý: nếu tập tin có sẵn không có mode thêm vào, Autolisp sẽ tạo ra một tập tin rỗng mới)

Open Returns a file descriptor.

Hàm **open** trả về giá trị **một bản mô tả tập tin**

Note: You must use (setq varNam(open "file" mode)) to enable closing of the file.

Chú ý: bạn phải sử dụng (setq varName (open "Tên tập tin" Mode)) để có thể đóng tập tin này

Note: Writing the file does not occur until you use the **close** statement.

Chú ý: Việc viết tập tin sẽ không thể xảy ra cho tới khi bạn sử dụng thông báo **Close**

TênBiếnTập tin – là một **tên biến** kết quả trả về của hàm open.

Example: **Ví dụ:**

```
(if (setq f(open "c:/acad/myfile.txt" "r")) ; Đặt biến f là
    ;kết quả của hàm open

    (progn

      (while (setq txtLine(read-line f)) ;Trong khi biến
        ;txtLine có giá trị là kết quả của hàm read-line
        ;với đối số f

        (princ txtLine);Hiển thị biến txtLine

      )
      ; Đóng thông báo progn

      (close f) ; Sử dụng hàm close đóng tập tin mang tên f

    )

    (princ "\n Error - File was not opened."))
```

)

read-line - This function reads a line of text from an open file.

Hàm này đọc một dòng văn bản trong một tập tin mở

Syntax : (**read-line fileVar**)

Cú pháp: (**read-line TênBiếnTậpTin**)

Where **fileVar** is a valid variable name that represents an open file descriptor.

TênBiếnTậpTin – là một tên biến có giá trị, đại diện cho một mô tả tập tin mở

Returns a text string.

Giá trị trả về - là một chuỗi văn bản

Assuming (`setq f (open "text.txt" "r")`) returned successfully. (giả sử hàm (`setq f (open "text.txt" "r")`) trả về giá trị có thật)

Assuming (`setq g (open "text8.txt" "r")`) failed. (giả sử hàm (`setq g (open "text8.txt" "r")`) không trả về giá trị)

(`read-line f`) ;returns string from file (trả về giá trị một chuỗi từ tập tin text.txt)

(`read-line g`) ;returns Error - Bad Argument Type (trả về lỗi Bad Argument)

write-line - This function writes a line of text to an open file.

Hàm này viết một dòng văn bản vào một tập tin mở

Syntax : (**write-line fileVar**)

Cú pháp: (**write-line TênBiếnTậpTin**)

Where **fileVar** is a valid variable name that represents an open file descriptor.

TênBiếnTậpTin – là một tên biến có giá trị đại diện cho một mô tả tập tin mở

Returns a text string.

Giá trị trả về - là một chuỗi văn bản

Assuming `(setq f (open "text.txt" "r"))` returned successfully. (Giả sử hàm `(setq f (open "text.txt" "r"))` trả về giá trị có thật.

Assuming `(setq g (open "text8.txt" "r"))` failed. (Giả sử hàm `(setq g (open "text8.txt" "r"))` không trả về giá trị.

`(write-line "Hello World" f)` ;writes text to file and returns string "Hello World" (viết dòng text vào tập tin text.txt và trả về chuỗi "Hello World")

`(write-line "Hello World" g)` ;returns Error - Bad Argument Type (trả về lỗi Bad Argument)

End of Read/Write and File Functions

Kết thúc các hàm đọc/viết và xử lý tập tin

Final note: **Chú ý cuối cùng:**

I've stated over and over that the OPEN statement opens a file. This can also be any resource available to you. Instead of opening "C:\ACAD\MyTextFile.txt" you can just as easily open "LPT1" or "\\myserver\myprinter" and write the file to a printer exactly like you write to a file. Cool! Printing from AutoLisp!

Tôi đã nhiều lần tuyên bố rằng hàm OPEN sẽ mở một tập tin. Nó còn có thể mở bất kỳ một nguồn có thể nào khác cho bạn. Thay vì mở tập tin " C:\ACAD\MyTextFile.txt " bạn có thể mở "LPT1" hay [\\myserver\myprinter](#) và viết một tập tin cho một máy in chính xác như bạn viết một tập tin thông thường. Tuyệt cú mèo! Hãy in từ Autolisp!

One thing to keep in mind. The file does not get written to disk or sent to the printer until after the CLOSE statement. [This will help you when debugging.] If you cannot get a printer to print, write the data into a text file instead of the printer. Then open the file with notepad and see if it is what you expect. If the data looks good in notepad then you've probably got the printer name wrong, a printer problem, or a network access problem.

Một điều cần nhớ. Tập tin này chưa được viết vào đĩa hay máy in cho đến sau khi bạn sử dụng hàm CLOSE. (Điều này sẽ giúp bạn khi chạy kiểm tra từng bước một chương trình). Nếu bạn không thể đạt tới một máy in để in, hãy viết dữ liệu thành một tập tin văn bản thay cho máy in. Sau đó mở tập tin bằng NotePad và kiểm tra xem đó có đúng là điều bạn cần không. Nếu dữ liệu hoàn toàn đúng trong NotePad, bạn chắc chắn đã có một tên máy in không đúng, một trục trặc của máy in, hoặc đường truyền có sự cố.

Example Program 1: Chương trình mẫu 1:

Let's write a program that prints a text file on the command line.

Ta sẽ viết một chương trình để in một tập tin văn bản trên dòng lệnh.

```
(defun C:pTxt() ;define a program name

  (setq fname(getstring "\n Enter a valid file name: ")) ;get file name
  cheaply (cách nhập tên tập tin đơn giản )

  (if(setq f(open fname "r")) ;open the file in read mode (mở tập tin ở mode đọc)

    (while (setq txtLine(read-line f)) ;while lines of text exist in file (Trong khi
      các dòng văn bản còn tồn tại trên tập tin)

      (princ txtLine) ;print the text string to the command line (in chuỗi văn bản ra
        dòng lệnh)

    ) ;close the while statement (đóng hàm while)

    (princ "\n Error - Could not find file") ;print a message on the else
    statement (Hiển thị thông báo khi hàm if nhận giá trị else)

  ) ;close the if statement (đóng hàm if)

; note: the if statement can be a " if then " statement or a " if then else" statement

Chú ý: Hàm if có hai cú pháp là “if ... then” hay “if ... then, else ...then”

) ;close the program (đóng chương trình)
```

[AutoLisp Intermediate Tutorial Home](#)

[AutoLisp Intermediate Tutorial](#)

[AutoLisp Tutorial Home](#)

[AutoLisp Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated March 31st, 2005

Copyright 2002-2007 JefferyPSanders.com. All rights reserved.

The AutoLisp Advanced Tutorial

Autolisp cao cấp

1/20/03 - Currently working on this. Per visitors request I'll continue where I left off from the Intermediate Tutorials. What was I thinking?

Let's get started with the Advanced Tutorial. But where? Where do we start? I can't decide. I'm stuck between doing this and doing that. Back and forth. Back and forth. Hardwired in an unconditional loop. Loop? Aha!

Loops

- **repeat while**

Conditional

- **cond if**

Entities DXF Codes

- **Line Circle Text Arc PolyLine LWPolyline Spline
Ellipse Solid Trace Block Attribute**

Tired of waiting on me to get around to it?

Got something that you need to know how to do right now?

Click this-----> [I Need This Now!](#)

Loop Functions:

Các hàm lặp:

[While](#) [Repeat](#)

[Example Programs](#)

While ; Giờ là lúc chúng ta kết thân với chú While đây

Syntax : (**while** **expression** **dothis**)

Cú pháp: (**while** **Điều kiện** **Hành động**)

expression - Any valid autolisp expression that evaluates to true or non nil.

Điều kiện – là bất kỳ một mô tả điều kiện nào của Autolisp có giá trị là đúng hay sai hay nil

dothis - Any valid autolisp expression or expressions

Hành động – là bất kỳ mô tả hành động cần thực hiện nào dưới dạng các hàm trong Autolisp

Note: No PROGN necessary for multiple expressions like the IF statement.

Chú ý: Ở hàm này không cần có thông báo progn cho nhiều hành động cần thực hiện như hàm if

(while T (princ 1)) returns 1 and keeps returning 1 forever. You're locked into the loop.

(Trả về giá trị 1 và giữ kết quả này mãi mãi. Hàm của bạn bị khoá chết trong vòng lặp)

(while nil (princ 1)) Exits the loop. (Hàm này thoát khỏi vòng lặp)

```

-----
(setq cnt 1) ; setup a counter (Khởi tạo bộ đếm)

(while (< cnt 4) ; loop until cnt is not less than 4 (lặp cho tới khi cnt không nhỏ
    hơn 4)

    (princ cnt) ; print the cnt

    (setq cnt(+ cnt 1)) ; increment the counter (Tăng giá trị cnt )

) ; returns 1 then 2 then 3 and then exits the loop (trả về các giá trị 1, 2, 3, rồi thoát lặp)

```

```

-----
(setq eset(ssget)) ;returns a selection set (trả về một bộ lựa chọn)

(setq cnt 1) ;setup a counter (Khởi tạo bộ đếm)

(while (< cnt (sslenght eset)) ; loop through all entities in the selection set
    (lặp thông qua tất cả các đối tượng trong bộ lựa chọn)

    (setq en(ssname eset cnt)) ; get the entity name of the nth item (lấy
        tên đối tượng của đối tượng thứ n trong bộ lựa chọn)

    (setq enlist(entget en)) ; get the DXF group codes of the entity
        (lấy các mã nhóm DXF của đối tượng en)

    (princ (cdr(assoc 0 enlist))) ; print the entity type to the command line
        (in ra dòng lệnh loại đối tượng của đối tượng en)

    (setq cnt(+ cnt 1)) ; increment the counter (tăng bộ đếm)

) ;close the while loop (đóng vòng lặp)

```

repeat - This function does exactly what you think it would do.

Hàm này thực hiện chính xác điều bạn muốn thực hiện.

Syntax : (**repeat integer**)

Cú pháp: (**repeat Số nguyên**)

integer - Any number that is an integer.

Số nguyên – là một số nguyên có giá trị bất kỳ.

Note: No PROGN necessary for multiple expressions like the IF statement.

Chú ý: không cần có thông báo PROGN khi có nhiều hành động cần thực hiện như hàm if

```
(repeat 20
```

```
  (princ "A")
```

```
) ;prints the letter "A" to the command line twenty times (in ra chữ cái  
  "A" hai mươi lần trên dòng lệnh)
```

```
(setq a 1 b 5) ;setup some variables. (dùng hàm setq đặt giá trị cho nhiều biến)
```

```
(repeat 4 ;repeat these functions 4 times. (Thực hiện lặp 4 lần)
```

```
  (setq a (+ a 1)) ;add 1 to [a] each loop (Thêm 1 vào a mỗi lần lặp)
```

```
  (setq b (+ b 5)) ;add 5 to [b] each loop (Thêm 5 vào b mỗi lần lặp)
```

```
) ;close the loop (Kết thúc các vòng lặp)
```

```
(princ a) ;prints 5 to the command line (In 5 ra dòng lệnh)
```

```
(princ b) ;prints 25 tot he command line (In 25 ra dòng lệnh)
```

Example Program 1: Chương trình mẫu 1:

```
(defun C:DrawLines()
```

[define program]

```
(setq pt1(getpoint "\n First Point: "))
```

[get first point] (lấy điểm thứ nhất)

```
(while (/= nil (setq pt2 (getpoint pt1 "\n Next Point: "))) [get next point]  
  (lấy điểm thứ hai)
```

```

(command "line" pt1 pt2 "") [draw a line] (vẽ đoạn thẳng)

(setq pt1 pt2) [set pt1 to last point] (đặt pt1 thành điểm pt2)

) [close the loop] (đóng vòng lặp)

) [close the program] (đóng chương trình)

```

Example Program 2: Chương trình mẫu 2:

```

(defun C:LetterGuess() ; define a program

  (setq myAnswer "J") ; set up a variable to hold the answer (đặt biến giữ câu trả lời)

  (setq yourGuess nil) ; set up a variable to hold your guess (đặt biến giữ câu đoán)

  (while (/= myAnswer yourGuess) ; while you don't know the answer (trong khi câu đoán sai)

    (setq yourGuess(getstring "\n Guess what letter: ")) ; get your guess (nhập câu đoán của bạn)

  ) ; close the loop (đóng vòng lặp)

) ; close the program (đóng chương trình)

```

Example Program 3: Chương trình mẫu 3:

; Chương trình này tách một chuỗi bạn nhập vào thành các ký tự rồi và in ra kết quả

```

(defun C:PrintVert() ; define a program

  (setq str(getstring "\n Enter a string:")) ; get a string from the user

  (setq cntr 1) ; setup a counter

  (repeat (strlen str) ; loop once for each character in string

    (setq myChar(substr str cntr 1)) ; get nth character in string

    (princ (strcat "\n " myChar)) ; print a new line then the nth character
  )

```

```

    (setq cntr(+ cntr 1))          ; increment the counter
  )                                ; close the loop
)                                  ; close the program

```

End of Loop Functions

Kết thúc các hàm lặp

Conditional Statements:

Các hàm điều kiện:

if cond

Coming soon... (Nhanh thiết, đã tới tuổi cặp kè cùng với nàng **If** rồi đây)

if - This function evaluates an expression to decide which expressions to do afterwards.

Hàm này lượng giá một biểu thức Autolisp để xác định các biểu thức Autolisp cần thực hiện sau đó.

Syntax : (if **thisIsTrue** **thenDoThis**)

Cú pháp 1: (if **Biểuthứckiểm** **Hànhđộng**)

Syntax : (if **thisIsTrue** **thenDoThis** **elseDoThis**)

Cú pháp 2: (if **Biểuthứckiểm** **Hànhđộng1** **Hànhđộng2**)

Syntax : (if **thisIsTrue** (progn **thenDoAllOfThis**) (progn **elseDoAllOfThis**))

Cú pháp 3: (if **Biểuthứckiểm** (progn **Cáchhànhđộng1**) (progn **Cáchhànhđộng2**))

Syntax : (if **thisIsTrue** (progn **thenDoAllOfThis**) **elseDoThis**)

Cú pháp 4: (if **Biểuthứckiểm** (progn **Cáchhànhđộng1**) **Hànhđộng2**)

Syntax : (if **thisIsTrue** **thenDoThis** (progn **elseDoAllOfThis**))

Cú pháp 5: (if **Biểuthứckiểm** **Hànhđộng1** (progn **Cáchhànhđộng2**))

thisIsTrue - Any valid autolisp expression that evaluates to true or non nil.

Biểuthứckiểm – là bất kỳ một **biểu thức Autolisp** có giá trị **đúng** hay **sai** hay **nil**

thenDoThis - Any valid autolisp expression.

Hànhđộng1 – là một **biểu thức Autolisp** có giá trị bất kỳ, được thực hiện khi giá trị của **Biểuthứckiểm** là **Đúng**

elseDoThis - Any valid autolisp expression.

Hànhđộng2 – là một **biểu thức Autolisp** có giá trị bất kỳ, được thực hiện khi **Biểuthứckiểm** có giá trị là **sai** hay **nil**

thenDoAllOfThis - Any valid autolisp expressions.

Cáchhànhđộng1 – là các **biểu thức Autolisp** có giá trị, được thực hiện khi **Biểuthứckiểm** có giá trị **Đúng**

elseDoAllOfThis - Any valid autolisp expressions.

Cáchhànhđộng2 – là các **biểu thức Autolisp** có giá trị, được thực hiện khi **Biểuthứckiểm** có giá trị **sai** hay **nil**

progn - Simply means there will be more than one statement here.

Progn – đơn giản là **hàm thông báo** có nhiều biểu thức Autolisp cần thực hiện trong nội hàm.

(if T (princ 3)) ; returns 3 (trả về giá trị 3)

(if T (princ 3) (princ 4)) ; returns 3 because T always evaluates to True (trả về giá trị 3 vì biến T luôn có giá trị là đúng (true))

(if nil (princ 3) (princ 4)) ; returns 4 because nil always evaluates to False (trả về giá trị 4 vì biến nil luôn có giá trị là sai (false))

(if (= nil T) (princ 3) (princ 4)) ; returns 4 because nil does not equal T (trả về giá trị 4 vì biến nil không thể có giá trị là T (true))

(if (= 3 3) (princ 3) (princ 4)) ; returns 3 (trả về giá trị 3)

(if (= 3 4) (princ 3) (princ 4)) ; returns 4 (trả về giá trị 4)

```
(if (= 3 3)
  (progn
    (princ 3)           ; returns 3 (trả về giá trị 3)
    (princ 5)           ; returns 5 (trả về giá trị 5)
  )
)
```

```
(if (= 3 3)
  (progn
    (princ 3)           ; returns 3 (trả về giá trị 3)
    (princ 5)           ; returns 5 (trả về giá trị 5)
  )
  (progn
    (princ 8)           ; program never gets inside here because 3 = 3 (hàm này không bao
                        ; giờ có thể thực hiện vì 3 = 3)
    (princ 9)           ; program never gets inside here because 3 = 3 (hàm này không bao
                        ; giờ có thể thực hiện vì 3 = 3)
  )
)
```

```
(if (= 3 4)
  (progn
    (princ 3)           ; program never gets inside here because 3 does not equal 4
  )
)
```

(hàm này không bao giờ có thể thực hiện vì 3 không thể bằng 4)

```
(princ 5) ;program never gets inside here because 3 does not equal 4 (hàm
này không bao giờ có thể thực hiện vì 3 không thể bằng 4)

)

(progn

  (princ 8) ; prints 8 (trả về giá trị 8)

  (princ 9) ; prints 9 (trả về giá trị 9)

)

)
```

; Giờ nếu muốn bạn có quyền đem nàng IF này về dinh mà xài thoải mái

cond - This function test conditions until one of them is true. At that point it exits the function.

Hàm này sẽ kiểm tra lần lượt các điều kiện cho tới khi một trong số các điều kiện thỏa mãn. Khi đó hàm sẽ thoát khỏi việc kiểm tra tiếp theo.

Syntax : (**cond**

```
(ifThisIsTrue thenDoThis)

(elseifThisIsTrue thenDoThis)

(elseifThisIsTrue thenDoThis)

)
```

Cú pháp 1: (**cond**

```
(Biểuthứckiểm1 Hànhđộng1)

(Biểuthứckiểm2 Hànhđộng2)

.....
```

(**Biểu thức kiểm tra Hành động n**)

)

Syntax : (**cond**

(**if thisIsTrue (progn thenDoAllOfThis))**

(**elseIf thisIsTrue thenDoThis**)

(**elseIf thisIsTrue (progn thenDoAllOfThis))**

(**elseIf thisIsTrue thenDoThis**)

(**elseIf thisIsTrue (progn thenDoAllOfThis))**

(**elseIf thisIsTrue thenDoThis**)

)

Cú pháp 2: (**cond**

(**Biểu thức kiểm tra 1 (progn Cách hành động 1)**)

(**Biểu thức kiểm tra 2 (progn Cách hành động 2)**)

.....

(**Biểu thức kiểm tra n (progn Cách hành động n)**)

)

thisIsTrue - Any valid autolisp expression that evaluates to true or non nil.

Biểu thức kiểm tra – là bất kỳ một biểu thức Autolisp có giá trị nào, trả về giá trị là đúng hoặc sai hay nil

thenDoThis - Any valid autolisp expression.

Hành động n – là bất kỳ một biểu thức Autolisp có giá trị nào, được thực hiện khi **Biểu thức kiểm tra** nhận giá trị đúng

elseDoThis - Any valid autolisp expression.

thenDoAllOfThis - Any valid autolisp expressions.

Cách hành động n – là các biểu thức Autolisp có giá trị, được thực hiện khi **Biểu thức kiểm tra** nhận giá trị

đúng

elseDoAllOfThis - Any valid autolisp expressions.

progn - Simply means there will be more than one statement here.

Progn – là một hàm thông báo có nhiều biểu thức Autolisp sẽ được thực hiện trong nội hàm

```
(cond
  ( (= 1 1) (princ "True") ) ;prints TRUE and exits (in ra giá trị True và thoát)
  ( (= 1 2) (princ "True") ) ;doesn't make it to this point (không thực hiện tới
                              bước này do bước trước đã đạt giá trị đúng)
)
```

```
(cond
  ( (= 1 0) (princ "True") ) ;skips because 1 does not equal 0 (Bỏ qua bước này
                              vì 1 không thể bằng 0)
  ( (= 1 1) (princ "True") ) ;prints TRUE and exits (in ra giá trị True và thoát)
  ( (= 1 2) (princ "True") ) ;doesn't make it to this point (không thực hiện tới
                              bước này do bước trước đã đạt giá trị đúng)
)
```

```
(cond
  ( (= 4 3) (princ "True") ) ;skips because 4 does not equal 3 (Bỏ qua bước này
                              vì 4 không thể bằng 3)
  ( (= 4 2) (princ "True") ) ;skips because 4 does not equal 2 (Bỏ qua bước này
                              vì 4 không thể bằng 2)
  ( (= 4 1) (princ "True") ) ; skips because 4 does not equal 1 (Bỏ qua bước này
```

vì 4 không thể bằng 1)

```
) ; returns nil (trả về giá trị nil)
```

```
(cond
```

```
  ( (= 4 3) (princ "True") ) ; skips because 4 does not equal 3 (Bỏ qua bước này vì 4 không thể bằng 3)
```

```
  ( (= 4 2) (princ "True") ) ; skips because 4 does not equal 2 (Bỏ qua bước này vì 4 không thể bằng 2)
```

```
  ( (= 4 1) (princ "True") ) ; skips because 4 does not equal 1 (Bỏ qua bước này vì 4 không thể bằng 1)
```

```
  ( T (princ "Nothing") ) ; prints "Nothing" because T = True (In ra giá trị "Nothing" vì biến T luôn luôn bằng True đúng)
```

```
) ; returns "Nothing" (trả về giá trị "Nothing")
```

```
(setq a ; set a variable
```

```
(cond
```

```
  ( (= 4 3) (princ "True") ) ; skips because 4 does not equal 3
```

```
  ( (= 4 2) (princ "True") ) ; skips because 4 does not equal 2
```

```
  ( (= 4 1) (princ "True") ) ; skips because 4 does not equal 1
```

```
  ( T (princ "Nothing") ) ; prints "Nothing" because T = True
```

```
) ; returns "Nothing"
```

```
) ; sets variable [a] to "Nothing" (Đặt biến a về giá trị chuỗi "Nothing")
```

End of Conditional Statements

Kết thúc các hàm điều kiện

Entity DXF Group Code Descriptions For:

Mô tả các mã nhóm DXF của đối tượng cho một số đối tượng:

<u>Arc</u>	<u>Attribute</u>	<u>Circle</u>	<u>Ellipse</u>
<u>Image</u>	<u>Insert / Block</u>	<u>Line</u>	<u>LWPolyLine</u>
<u>MLine</u>	<u>MText</u>	<u>Point/Node</u>	<u>PolyLine</u>
<u>Solid</u>	<u>Text</u>	<u>Trace</u>	<u>XLine</u>

Note: A good working example of all of these DXF group codes can be found in the [CAD2File.lsp](#) program on the AutoLisp home page. All code is remarked to make it easier to follow along.

Chú ý: Một ví dụ hiệu quả về tất cả các mã nhóm DXF này có thể tìm được trong chương trình CAD2file.lsp trên trang chủ của Autolisp. Tất cả các mã đều được đánh dấu để dễ theo dõi hơn trong suốt chương trình

Arc (Cung tròn)

Typical DXF Group Codes for an ARC Entity: (Các mã nhóm DXF điển hình của đối tượng cung tròn)

To list an arc entity in AutoCAD you type LIST<enter> and select the arc. To do this in AutoLisp you would:

Để liệt kê một đối tượng cung tròn trong AutoCad bạn sẽ nhập LIST <Enter> và lựa chọn cung tròn đó. Để làm điều đó trong Autolisp bạn phải thực hiện các hàm:

```
(setq en(car (entsel "\n Select an Arc: ")))
```

```
(setq enlist(entget en))
```

Would return: Kết quả trả về sẽ là:

```
((-1 . <Entity name: 2b80648>) (0 . "ARC") (5 . "89")(100 . "AcDbEntity") (67 . 0) (8 .  
"STR") (100 . "AcDbCircle") (10 1.5 0.5 0.0) (40 . 1.58114) (210 0.0 0.0 1.0) (100 .  
"AcDbArc") (50 .5.96143) (51 . 3.46334))
```

What does all of this garbage mean? (Tất cả những cái loằng ngoằng này là gì vậy?)

Let's take it apart and put it in order: (Ta sẽ tách từng thành một và bắt nó vào khuôn phép nhé)

```
( ; Mở đầu một danh sách
```

(-1 . <Entity name:2b80648>)	-1 - Entity Name (AutoCAD Automatically takes care of this) Tên đối tượng (AutoCad tự động quản lý thẳng này)
(0 . "ARC")	0 - Entity Type (Loại đối tượng)
(5 . "89")	5 - Handle Name (If handles are turned on) (Tên điều khiển (nếu các chức năng điều khiển được kích hoạt))
(8 . "STR")	8 - Layer Name (Tên lớp)
(10 1.5 0.5 0.0)	10 - Center Point (Vị trí tâm)
(40 . 1.58114)	40 - Radius (Bán kính)
(50 .5.96143)	50 - Start Angle Expressed in Radians (Góc bắt đầu theo Radians)
(51 . 3.46334)	51 - End Angle Expressed in Radians (Góc kết thúc theo Radians)
(67 . 0)	67 - Don't worry about this (Khỏi cần lo cho thẳng này)
(100 . "AcDbEntity")	100 - Don't worry about this (Khỏi cần lo cho thẳng này)
(100 ."AcDbCircle")	100 - Don't worry about this (Khỏi cần lo cho thẳng này)
(100 . "AcDbArc")	100 - Don't worry about this Sub Class Marker (Khỏi lo cho thẳng Đánh dấu Phân loại Bổ sung này)
(210 0.0 0.0 1.0)	210 - Don't worry about this Extrusion Factor(Khỏi lo cho thẳng Yếu tố Vuốt dài này)
) ; Kết thúc danh sách mã nhóm	

Let's play with it: (Nào ta hãy chọc ngoáy một tí với các mã nhóm này nhé)

(cdr(assoc 10 enlist)) would return (Trả về giá trị)(1.5 0.5 0.0)

Remember, CDR returns everything after the first item in a list. (Nhớ thẳng quây cdr chưa, nó trả về mọi thứ trong danh sách ngoại trừ khoản mục đầu tiên mà)

To get the center point of the arc: (Để có tọa độ tâm của cung tròn)

(cdr (assoc 10 enlist))

To get the radius of the arc : (Để có được bán kính của cung tròn)

```
(cdr (assoc 40 enlist))
```

To get the layer name of the arc : (Để biết tên lớp mà cung tròn này ngự)

```
(cdr (assoc 8 enlist))
```

To get the start angle of the arc : (Để biết góc bắt đầu của cung tròn)

```
(cdr (assoc 50 enlist))
```

To get the end angle of the arc : (Muốn lấy góc kết thúc của cung tròn)

```
(cdr (assoc 51 enlist))
```

To see if the entity is indeed an ARC entity : (Muốn kiểm tra đối tượng có thực sự là một cung tròn)

```
(if (= "ARC" (cdr (assoc 0 enlist))))
```

Attribute (Thuộc tính)

This may be a little lengthy. Better grab a cup of joe before getting started.

Chà ngán rồi đây. Tốt hơn là hãy vò lấy một vại trước khi bắt đầu với thằng này.

First off, an attribute is located inside a block entity. This means the entities contained inside a block are called sub-entities. All entities (Lines, text, attributes, etc.) inside the block are sub-entities. You have to (I'll retract this later) start with the block entity and step your way into it to find the sub-entities. Kind of like this :

Trước hết, một thuộc tính là được đặt bên trong một đối tượng block. Điều đó có nghĩa là các đối tượng chứa bên trong một block được gọi là đối tượng phụ. Tất cả các đối tượng (các đoạn thẳng, đoạn văn bản, các thuộc tính ...) trong block đều là các đối tượng phụ. Bạn phải bắt đầu với một đối tượng block và từng bước lần mò theo cách của bạn để thâm nhập vào nó và tìm ra các đối tượng phụ này. Đó là một thứ giống như sau:

```
(block entity name (sub-entity name 1) (sub-entity name 2) (sub-entity name 3) SEQEND)
```

You start with the Block's entity name and step through until you find SEQEND using the ENTNEXT function. I'll get back to this a little later on.

Bạn bắt đầu với tên của đối tượng block và sử dụng hàm ENTNEXT lần lượt đi qua cho tới khi bạn tìm được thằng SEQEND. Lát nữa tôi sẽ trở lại với vấn đề này.

You have a couple of choices on how to get to the attribute entity inside the block. Now, after I've gone on the record saying "You have to start at the top and step your way down through the sub-entities using ENTNEXT", I'm going to prove myself wrong.

Bạn có cả lô xích xông các lựa chọn sao cho có được đối tượng thuộc tính trong block này. Bây giờ tôi sẽ tiếp tục với đoạn nói rằng: “ Bạn phải bắt đầu từ đầu và sử dụng hàm ENTNEXT từng bước đi dần tới các đối tượng phụ”, tôi sắp chứng tỏ sự sai lầm của tôi đây.

The easiest method is using NENTSEL. I have not discussed the NENTSEL function until now. NENTSEL will return the DXF group code list for a sub-entity inside a block or 3d Polyline. We will stick to blocks for this discussion. You have to actually pick the attribute sub-entity and not any sub-entity contained inside the block. You can then modify the DXF group codes and update the attribute. This is fine for single selection and single manipulation of one attribute inside the block. I'll give a quick demonstration of the NENTSEL function to satisfy those out there that are dying to try it. You know who you are.

Phương pháp dễ nhất là sử dụng hàm NENTSEL. Cho tới lúc này tôi vẫn chưa đã động tới thằng NENTSEL tí nào. Hàm NENTSEL sẽ trả về danh sách mã nhóm DXF đối với một đối tượng phụ bên trong một block hay một đường Polyline 3D. Ta sẽ tóm lấy thằng block để làm cho ra vấn đề nhé. Bạn phải tóm lấy thằng đối tượng phụ thuộc tính chứ không phải bất kỳ một thằng đối tượng phụ nào khác trong block này. Sau đó bạn có thể sửa đổi các mã nhóm DXF của nó và cập nhật thuộc tính đó. Điều đó là tuyệt vời đối với sự lựa chọn đơn và thao tác đơn cho một thuộc tính trong block. Tôi sẽ biểu diễn nhanh về một hàm NENTSEL để thỏa mãn các yêu cầu trên đây, một hàm mà đang được mong ngóng thử. Bạn biết bạn là ai mà.

```
(setq ent(nentsel "\n Select a sub-entity: "))
```

NENTSEL returns the exact same thing as ENTSEL if the entity selected is not a complex entity like a Block or 3d Polyline. If you selected a complex entity such as an attribute inside a block, NENTSEL would return something like this:

Hàm NENTSEL trả về chính xác điều mà hàm entsel trả về nếu như đối tượng được chọn không phải là một đối tượng phức hợp như là block hay đường Polyline 3D. Nếu bạn chọn một đối tượng phức hợp chẳng hạn như một thuộc tính trong một block, hàm NENTSEL sẽ trả về một kết quả giống như sau:

```
(<Entity name: 400a14a8> (5193.24 4935.03 0.0) (20.0 0.0 0.0) (0.0 20.0 0.0) (0.0 0.0 20.0) (5237.78 4923.46 0.0)) (<Entity name: 40222278>))
```

Let's break it down: (Ta hãy ngắt khúc nó nhé)

(<Entity name: 400a14a8>	; - The actual entity name of the entity selected. (Tên đối tượng thực của đối tượng được chọn)
(5193.24 4935.03 0.0)	; - The point selected. (Tọa độ điểm chọn)
(; - Start of a four item list (mở một danh sách gồm bốn khoản mục)
(20.0 0.0 0.0)	; - We will ignore the Model to World Transformation Matrix (Tờ điểu về

khái niệm Model đối với Ma trận Chuyển vị World)

(0.0 20.0 0.0)

- We will ignore the Model to World Transformation Matrix

(0.0 0.0 20.0)

- We will ignore the Model to World Transformation Matrix

(5237.78 4923.46 0.0)

- We will ignore the Model to World Transformation Matrix

)

; - End of a four item list (kết thúc danh sách chứa bốn khoản mục)

(<Entity name: 40222278>)

; - The entity name of the block that contains the selected entity (Tên của đối tượng block chứa đối tượng được chọn)

) ;Kết thúc danh sách mã nhóm

To get the DXF Group codes of the selected attribute using NENTSEL :

Để có được các mã nhóm DXF của một thuộc tính được lựa chọn sử dụng hàm NENTSEL:

```
(if (setq ent(nentsel "\n Select Attribute: ")) ; - Select the attribute (lựa chọn thuộc tính)
```

```
(progn
```

```
(setq en(car ent))
```

```
; - Get the entity name (lấy tên đối tượng)
```

```
(setq enlist(entget en))
```

```
; - Get the DXF Group codes (lấy các mã nhóm DXF)
```

```
)
```

```
(princ "\n Nothing Selected!")
```

```
; - Print message on failure (hiển thị khi không chọn đối tượng)
```

```
)
```

That's enough on NENTSEL. Let's get back to work finding attribute information. First thing we need to do is let the user select a block :

Vậy là đủ với thằng NENTSEL. Ta hãy quay lại việc tìm kiếm các thông tin thuộc tính. Điều thứ nhất ta phải làm là cho phép người sử dụng chọn một block:

```
(setq ent(entsel "\n Select a block: "))
```

Then let's get the entity name and dxf group codes of the block:

Sau đó cho phép lấy tên đối tượng và các mã nhóm DXF của block

```
(setq en(car ent))  
  
(setq enlist(entget en))
```

Command: !enlist<enter> returns: (trả về giá trị)

```
((-1 . <Entity name: 40222278>) (0 . "INSERT") (330 . <Entity name: 40073cf8>) (5 . "85F") (100 .  
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "DELAMINATION") (100 . "AcDbBlockReference") (66 . 1) (2 .  
"At117") (10 5237.78 4923.46 0.0) (41 . 20.0) (42 . 20.0) (43 . 20.0) (50 . 0.0) (70 . 0) (71 . 0) (44 . 0.0) (45 .  
0.0) (210 0.0 0.0 1.0))
```

Let's take it apart, put it in order, and get rid of the codes that are not needed:

Hãy tách thành từng phần, sắp xếp lại và loại bỏ những mã không cần thiết:

Note: If you want to see the other group codes please see the Insert / Block entity on this page. (Chú ý: Nếu bạn muốn xem các mã nhóm khác, hãy xem mục Nhập đối tượng Block trên trang này)

```
(  
  
  (-1 . <Entity name: 40222278>)      ; -1 - Entity name (Tên đối tượng)  
  
  (0 . "INSERT")                      0 - Entity type (loại đối tượng)  
  
  (2 . "At117")                       2 - Name of the block (Tên block)  
  
  (10 5237.78 4923.46 0.0)             10 - Insertion Point (Điểm nhập block)  
  
  (41 . 20.0)                         41 - X scale factor (Tỉ lệ nhập theo trục X)  
  
  (42 . 20.0)                         42 - Y scale factor (Tỉ lệ nhập theo trục Y)  
  
  (43 . 20.0)                         43 - Z scale factor (Tỉ lệ nhập theo trục Z)  
  
  (50 . 0.0)                          50 - Rotation angle (Góc quay khi nhập)  
  
  (66 . 1)                            66 - Attributes follow flag (Các thuộc tính theo flag)  
  
)
```

Notice the red code. Code number 66. This is the important one for finding attributes. If this code has a

value of zero, there are no attributes in the block. If it has a value of 1, the block contains attributes. Simple! You find the attributes by stepping through the block using ENTNEXT until you reach the SEQEND entity.

Chú ý mã màu đỏ. Số mã 66. Mã này là một điều quan trọng để tìm các thuộc tính. Nếu mã này có giá trị 0, không có các thuộc tính trong block. Nếu nó có giá trị 1, block có chứa các thuộc tính. Cực đơn giản! Bạn hãy tìm các thuộc tính đó nhờ hàm ENTNEXT lần lượt từng bước thông qua block cho tới khi đạt tới đối tượng SEQEND

So, how do we do that? Like this: (Thế đó, làm thế nào để làm được điều đó? Hãy làm như sau:)

To get the attribute's DXF group codes: (Để có các mã nhóm DXF của thuộc tính)

```
(if (setq ent(entsel "\n Select a Block: ")) ;- Let the user select a block (Chọn block)

(progn

  (setq en(car ent))                ;- Get the entity name of the block (lấy tên đối tượng của block)

  (setq enlist(entget en))          ;- Get the DXF group codes (lấy các mã nhóm DXF)

  (setq blkType(cdr(assoc 0 enlist))) ;- Save the type of entity (lưu lại loại đối tượng)

  (if (= blkType "INSERT")          ;- If the entity type is an Insert entity (nếu loại đối tượng là đối
                                     tượng Insert)

    (progn

      (if(= (cdr(assoc 66 enlist)) 1) ;- See if the attribute flag equals one (if so, attributes
                                     follow) (Kiểm tra flag thuộc tính )

        (progn

          (setq en2(entnext en))      ;- Get the next sub-entity (lấy đối tượng phụ tiếp theo)

          (setq enlist2(entget en2))   ;- Get the DXF group codes (lấy các mã nhóm DXF)

          (while (/= (cdr(assoc 0 enlist2)) "SEQEND") ;- Start the while loop and keep
                  ;- looping until SEQEND is found. (Lặp trong khi loại đối tượng không phải SEQEND)

            (princ "\n ")              ;-Print a new line (tạo dòng kết quả mới)

            (princ enlist2)            ;- Print the attribute DXF group codes (in ra các mã nhóm DXF)

            (setq en2(entnext en2))    ;- Get the next sub-entity (lấy đối tượng phụ tiếp theo)
```

```

        (setq enlist2(entget en2))    ;- Get the DXF group codes (lấy các mã nhóm DXF)

    ) ; Kết thúc hàm while

    ) ; Kết thúc hàm thông báo progn lần thứ 3

    ) ;- Close the if group code 66 = 1 statement (Đóng hàm if kiểm tra mã nhóm 66 )

    ) ; Kết thúc hàm thông báo progn lần thứ 2

    ) ;- Close the if block type = "ATTRIB" statement (Đóng hàm if kiểm tra loại đối tượng)

    ) ; Đóng thông báo progn lần thứ nhất

    ) ;- Close the if an Entity is selected statement (Đóng hàm if kiểm tra việc chọn đối tượng)

```

Finally we get to see the Attribute DXF group codes which should be printed out on your command line or text screen. They should look something like this:

Cuối cùng ta có các mã nhóm DXF của thuộc tính được in ra trên dòng lệnh của màn hình trông như sau:

```

((-1 . <Entity name: 40222290>) (0 . "ATTRIB") (330 . <Entity name: 40222278>) (5 . "862") (100 .
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "TITLETXT_S") (100 . "AcDbText") (10 5191.7 4940.62 0.0) (40
. 1.6) (1 . "CTSE890") (50 . 0.0) (41 . 1.0) (51 . 0.0) (7 . "arial80") (71 . 0) (72 . 0) (11 0.0 0.0 0.0) (210 0.0 0.0
1.0) (100 . "AcDbAttribute") (2 . "PROJ") (70 . 0) (73 . 0) (74 . 0))

```

Let's take it apart and put it in order: (hãy tách chúng thành từng phần và sắp xếp lại)

(
(-1 . <Entity name: 40222290>)	-1 - Entity name (Tên đối tượng)
(0 . "ATTRIB")	0 - Entity Type (loại đối tượng)
(1 . "CTSE890")	1 - Attribute Value (Giá trị thuộc tính)
(2 . "PROJ")	2 - Attribute Tag (Đuôi thuộc tính)
(5 . "862")	5 – Handle (Chức năng điều khiển)
(7 . "arial80")	7 - Text Style (Kiểu chữ)

(8 . "TITLETXT_S")	8 - Layer name (Tên lớp)
(10 5191.7 4940.62 0.0)	10 - Insertion Point (Điểm nhập vào)
(11 0.0 0.0 0.0)	11 - Text alignment point (optional) (Điểm căn chỉnh dòng văn bản (Định hướng))
(40 . 1.6)	40 - Text Height (Chiều cao chữ)
(41 . 1.0)	41 - X scale factor (also used for fit text) (Tỉ lệ nhập theo trục X, cũng được sử dụng đối với việc nhập theo fit)
(50 . 0.0)	50 - Text Rotation (Góc quay chữ)
(51 . 0.0)	51 - Text Oblique Angle (Góc nghiêng chữ)
(67 . 0)	67 - Absent or zero = model space / 1 = paper space (không có hay zero chỉ không gian của model, 1 chỉ không gian giấy)
(70 . 0)	70 - Attribute flag (Chỉ thị thuộc tính) 0 = No flags (Không có chỉ thị) 1 = Invisible (Không nhìn thấy được) 2 = Constant (Không thay đổi) 4 = Verification is Required (Cần kiểm tra) 8 = Preset (no prompt) (Đặt lại không cần nhắc)
(71 . 0)	71 - Text Generation Flag (See text entity) (Chỉ thị nơi khởi tạo text)
(72 . 0)	72 - Horiz. Text Justification (See text entity) (Chữ cân đều theo chiều ngang, xem đối tượng văn bản)
(73 . 0)	73 - (See text entity) (Xen đối tượng văn bản)
(74 . 0)	74 - Vertical Text Justification (See text entity) (Chữ cân đều theo chiều đứng, xem đối tượng text)
(100 . "AcDbEntity")	100 - Ignore (Mã này tởn đêch hiều)
(100 . "AcDbText")	100 - Ignore
(100 . "AcDbAttribute")	100 - Ignore
(210 0.0 0.0 1.0)	210 - Extrusion direction (ignore) (Hướng vuốt dài)

```
(330 . <Entity name: 40222278>) 330 - Ignore
```

```
(410 . "Model") 410 - Layout Tab Name (Tên Tab layout)
```

```
)
```

Let's play with it: Thử đùa một tí:

To get the value of the attribute using the program from above: (để có được giá trị của thuộc tính sử dụng trong chương trình trên:)

```
(cdr(assoc 1 enlist2))
```

To get the attribute Tag: (để có được đuôi thuộc tính:)

```
(cdr(assoc 2 enlist2))
```

To change the value of the attribute: (Để thay đổi giá trị của thuộc tính:)

Syntax: (subst new_item old_item entity_list)

Cú pháp: (subst Khoảnmựcmới Khoảnmựccũ Danh sách Đối tượng)

; Chớ có nhầm lẫn hàm subst với hàm substr

```
(setq newVal (getstring "\n New Attribute value: "))
```

```
(setq enlist2 (subst (cons 1 newVal) (assoc 1 enlist2) enlist2))
```

```
(entmod enlist2)
```

```
(entupd en2)
```

Phew....I think we got through that unscathed. If you have any questions [<Send Email>](#)

Phù Tôi nghĩ rằng ta vừa thoát nạn mà vô sự. Nếu bạn có thắc mắc gì thì hỏi qua thư điện tử nhé.

Circle (Vòng tròn)

Typical DXF Group Codes for a Circle Entity: Các mã nhóm DXF diễn hình cho đối tượng vòng tròn

To list a circle in AutoCAD you type LIST<enter> and select the circle. To do this in AutoLisp you would:

Để liệt kê một đối tượng vòng tròn trong AutoCad bạn sẽ nhập LIST <Enter> và lựa chọn vòng tròn đó. Để làm điều đó trong Autolisp bạn phải thực hiện các hàm:

```
(setq en(car(entsel "\n Select a Circle: ")))
```

```
(setq enlist(entget en))
```

Would return: (trả về giá trị)

```
((-1 . <Entity name: 37b0650>) (0 . "CIRCLE") (5 . "8A") (100 . "AcDbEntity")  
(67 . 0) (8 . "STR") (100 . "AcDbCircle") (10 17.4375 9.6875 0.0) (40 .  
1.5) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: Ta hãy tách chúng thành từng phần và sắp xếp lại:

```
(  
  (-1 . <Entity name: 37b0650>)    -1 - AutoCad Entity Name (AutoCAD does this  
                                   Automatically) (Tên đối tượng trong AutoCad do  
                                   AutoCad tự động thực hiện)  
  
  (0 . "CIRCLE")                  0 - Entity Type (Loại đối tượng)  
  
  (5 . "8A")                      5 - Handle Name (If handles are turned on) (Tên điều  
                                   khiển nếu chức năng điều khiển được kích hoạt)  
  
  (8 . "STR")                      8 - Layer Name (Tên lớp)  
  
  (10 17.4375 9.6875 0.0)          10 - Center point of circle (Tâm vòng tròn)  
  
  (40 . 1.5)                      40 - Radius of circle (Bán kính vòng tròn)  
  
  (67 . 0)                        Don't worry about this (Khỏi lo về mã này)  
  
  (100 . "AcDbCircle")            Don't worry about this (Khỏi lo về mã này)  
  
  (100 . "AcDbEntity")            Don't worry about this (Khỏi lo về mã này)  
  
  (210 0.0 0.0 1.0)              Don't worry about this (Khỏi lo về mã này)  
)
```

Let's play with it: Thử chơi với các mã này:

`(cdr(assoc 10 enlist))` would return (trả về giá trị) `(17.4375 9.6875 0.0)`

Remember, CDR returns everything after the first item in a list. (Nhớ rằng, hàm CDR trả về mọi thứ sau khoản mục thứ nhất của một danh sách)

To get the radius of the circle : Để có bán kính của vòng tròn:

```
(cdr (assoc 40 enlist))
```

To get the center of the circle : Để có tâm của vòng tròn:

```
(cdr(assoc 10 enlist))
```

To get the layer name of the circle : Để có tên lớp chứa vòng tròn:

```
(cdr(assoc 8 enlist))
```

To see if the entity is indeed a CIRCLE entity : Để kiểm tra đối tượng có thực là vòng tròn hay không?

```
(if (= "CIRCLE" (cdr(assoc 0 enlist)))
```

; Chú ý là hàm if ở đây còn đang mở, chưa hoàn thiện.

Ellipse (Ellipse)

Typical DXF Group Codes for an ELLIPSE Entity: Các mã nhóm DXF điển hình của đối tượng ellipse:

To list an ellipse entity in AutoCAD you type LIST<enter> and select the ellipse. To do this in AutoLisp you would: (Để liệt kê một đối tượng ellipse trong AutoCad bạn sẽ nhập LIST <Enter> và lựa chọn ellipse đó. Để làm điều đó trong Autolisp bạn phải thực hiện các hàm:)

```
(setq en(car (entsel "\n Select an Ellipse :")))  
(setq enlist(entget en))
```

Would return: (trả về giá trị):

```
((-1 . <Entity name: 1cb0670>) (0 . "ELLIPSE") (5 . "86") (100 . "AcDbEntity"  
) (67 . 0) (8 . "STR") (100 . "AcDbEllipse") (10 3.0 3.0 0.0) (11 21.2132 -  
21.2132 0.0) (210 0.0 0.0 1.0) (40 . 0.0471405) (41 . 0.0) (42 . 6.28319))
```

Let's take it apart and put it in order: Ta hãy tách chúng thành từng phần và sắp xếp lại:

```
(
  (-1 . <Entity name:1cb0670>) -1 - Entity Name (AutoCAD Automatically takes care of this)
                                (Tên đối tượng do AutoCad tự động thực hiện)

  (0 . "ELLIPSE")              0 - Entity Type (Loại đối tượng)

  (5 . "86")                   5 - Handle Name (If handles are turned on) (Tên điều khiển nếu
                                chức năng điều khiển được kích hoạt)

  (8 . "STR")                  8 - Layer Name (Tên lớp)

  (10 3.0 3.0 0.0)             10 - Center Point (Tọa độ tâm ellipse)

  (11 21.2132 -21.2132 0.0)    11 - end Point of Major axis (relative to center) (Điểm mút của
                                trục lớn so với tâm)

  (40 . 0.0471405)            40 - Ratio of Minor axis to Major axis (tỉ lệ giữa trục nhỏ và
                                trục lớn)

  (41 . 0.0)                   41 - Start Parameter (Equals 0.0 for closed ellipse) (Thông số
                                bắt đầu, có giá trị là không đối với ellipse đóng)

  (42 . 6.28319)              42 - End Parameter (Equals 2 * pi for closed ellipse) (Thông số
                                kết thúc, có giá trị bằng 2*pi đối với ellipse đóng)

  (67 . 0)                    67 - Absent or zero indicates the entity is in model space. If set
                                to 1, paper space. (Không có hay bằng không chỉ thị rằng
                                đối tượng ở trong không gian model, bằng 1 chỉ thị đối
                                tượng ở không gian giấy vẽ)

  (100 . "AcDbEntity")        100 - Don't worry about this (Không cần quan tâm tới mã này)

  (100 . "AcDbEllipse")       100 - Don't worry about this (Không cần quan tâm tới mã này)

  (210 0.0 0.0 1.0)          210 - Don't worry about this Extrusion Factor (Không cần quan
                                tâm tới yếu tố vuốt dài này)

)
```

Let's play with it: Hãy thực hành với các mã này:

`(cdr(assoc 10 enlist))` would return (trả về giá trị) `(3.0 3.0 0.0)`

Remember, CDR returns everything after the first item in a list.

To get the center point of the ellipse : Để có được tọa độ tâm của elip:

```
(cdr (assoc 10 enlist))
```

To get the length of the major axis : Để có được độ dài của trục lớn:

```
;;;--- Save the center point of the ellipse (lưu lại tâm elip)
(setq centerPt(cdr(assoc 10 enlist)))

;;;--- Save the endPoint of the major axis relative to center pt (lưu lại
điểm mút của trục lớn so với tâm)
(setq endPt(cdr(assoc 11 enlist)))

;;;--- Create a new point on the end of the major axis (Tạo một điểm mới ở
cuối trục lớn)

;;; by subtracting code 10 point from code 11 point (Trừ các tọa độ
điểm mã 11 đi các tọa độ tương ứng của điểm mã 10)
(setq newPt
  (list
    (- (car centerPt) (car endPt))
    (- (cadr centerPt) (cadr endPt))
  )
)

;;;--- Finally, find the major axis length (Cuối cùng tìm độ dài trục lớn)
(setq majorAxisLength(* 2.0 (distance centerPt newPt)))
```

To get the length of the minor axis : Để có được độ dài của trục nhỏ:

```
(* majorAxisLength (cdr(assoc 40 enlist)))
```

Note: See example above for majorAxisLength (Lưu ý: Xem ví dụ trên đối với độ dài trục lớn)

To see if the entity is indeed an Ellipse entity : Để kiểm tra đối tượng có phải là elip thực không:

```
(if (= "ELLIPSE" (cdr(assoc 0 enlist))) ;Chú ý: hàm này chưa hoàn thiện
```

Image

Hình ảnh

To list an image entity in AutoCAD you type LIST<enter> and select the image. To do this in AutoLisp you would: (Đề liệt kê một đối tượng hình ảnh trong AutoCad bạn sẽ nhập LIST <Enter> và lựa chọn hình ảnh đó. Để làm điều đó trong Autolisp bạn phải thực hiện các hàm:)

```
(setq en(car (entsel "\n Select an Image :")))  
(setq enlist(entget en))
```

Would return some garbage like this: (Trả về kết quả là danh sách các mã nhóm DXF như sau:)

```
((-1 . <Entity name: 40073da8>) (0 . "IMAGE") (330 . <Entity
name: 40073cf8>) (5 . "4D") (100 . "AcDbEntity") (410 . "Model") (8 . "0")
(100 . "AcDbRasterImage") (90 . 0) (10 9.00868 6.59115 0.0) (11 0.0012987
0.0 0.0) (12 7.95199e-020 0.0012987 0.0) (13 770.0 559.0 0.0) (340 .
<Entity name: 40073d98>) (70 . 7) (280 . 0) (281 . 50) (282 . 50) (283 .
0) (360 . <Entity name: 40073da0>) (71 . 1) (91 . 2) (14 -0.5 -0.5 0.0)
(14 769.5 558.5 0.0))
```

Let's take it apart and put it in order: Tách chúng thành từng phần và sắp xếp lại:

(-1 . <Entity name: 40073da8>)	-1 Entity Name (AutoCAD Automatically takes care of this) (tên đối tượng do AutoCad tự động thực hiện)
(0 . "IMAGE")	0 - Entity Type (Loại đối tượng)
(5 . "4D")	5 - Handle Name (If handles are turned on) (Tên điều khiển nếu chức năng điều khiển được kích hoạt)
(8 . "0")	8 - Layer Name (Tên lớp)
(10 9.00868 6.59115 0.0)	10 - Insertion Point (Toạ độ điểm nhập vào bản vẽ)
(11 0.0012987 0.0 0.0)	11 - U-vector of a single pixel (points along the visual bottom of the image, starting at the insertion point) (in OCS) (vector U của một pixel đơn tức các điểm dọc theo cạnh đáy quan sát của hình ảnh bắt đầu từ điểm nhập vào bản vẽ theo hệ tọa độ OCS)
(12 7.95199e-020 0.0012987 0.0)	12 - V-vector of a single pixel (points along the visual left of the image, starting at the insertion point) (in OCS) (vector V của một pixel đơn tức các điểm dọc theo cạnh trái quan sát của hình ảnh bắt đầu từ điểm nhập vào bản vẽ theo hệ tọa độ OCS)

(13 770.0 559.0 0.0)

13 - Image size in pixels (Cỡ ảnh theo pixel)

(14 -0.5 -0.5 0.0)

14 - Clip boundary vertex (in OCS) (Các đỉnh giới hạn của clip ảnh)

1 . For rectangular clip boundary type, two corners must be specified. Default is (-0.5,-0.5), (size.x-0.5, size.y-0.5). (Với loại giới hạn clip hình chữ nhật, phải xác định hai góc . Giá trị mặc định là (-0.5, -0.5), (cỡ theo trục x là -0.5, cỡ theo trục y là -0.5))

2 . For polygonal clip boundary type, three or more vertices must be specified. Polygonal vertices must be listed sequentially. (Với loại giới hạn clip hình đa giác phải có ba hoặc hơn 3 các đỉnh được xác định. Các đỉnh của đa giác phải được liệt kê liên tiếp)

(14 769.5 558.5 0.0)

14 - Other corner for rectangular. See (1) above. (Đỉnh đối diện của clip giới hạn theo hình chữ nhật. Xem giải thích 1 ở phần trên)

(70 . 7)

70 - Image display properties (Can be added together) (Các thuộc tính của hình ảnh có thể là tổng của chúng với nhau)

1. Show Image (Hiển thị ảnh)
2. Show image when not aligned with screen. (Hiển thị hình ảnh khi không được căn chỉnh theo màn hình)
4. Use clipping boundary (Sử dụng giới hạn clip)
8. Transparency is on. (Kích hoạt chức năng transparency)

With a value of 7 it is using 1, 2, & 4 (Giá trị 7 ở đây có nghĩa là sử dụng đồng thời ba thuộc tính 1, 2 và 4)

(71 . 1)

71 - Clipping boundary type. 1= Rectangular 2=Polygonal (Loại giới hạn clip. 1 là loại giới hạn hình chữ nhật, 2 là loại giới hạn hình đa giác)

(90 . 0)

90 - Class Version (Loại version ảnh)

(91 . 2)

91 - Number of clip boundary vertices that follow. (Số đỉnh giới hạn của clip kèm theo)

(330 . <Entity name: 40073cf8>)

330 – Ignore (Không cần biết)

(100 . "AcDbEntity")

100 – Ignore (Không cần biết)

(100 . "AcDbRasterImage")

100 – Ignore (Không cần biết)

(280 . 0)	280 - Clipping state 0=Off 1=On (Trạng thái của clip. 0 là tắt, 1 là mở)
(281 . 50)	281 - Brightness Value 0 to 100 (Default = 50) (Giá trị độ sáng từ 0 tới 100. Giá trị mặc định là 50)
(282 . 50)	282 - Contrast Value 0 to 100 (Default = 50) (Giá trị độ tương phản từ 0 tới 100. Giá trị mặc định là 50)
(283 . 0)	283 - Fade Value 0 to 100 (Default = 0) (Giá trị độ bóng đổ của ảnh. Giá trị mặc định là 0)
(340 . <Entity name: 40073d98>)	340 - Ignore Hard reference to imagedef object (Thông số tham khảo đối với đối tượng khuếch tán ảnh, không cần thiết)
(360 . <Entity name: 40073da0>)	360 - Ignore Hard reference to imagedef_reactor object (thông số tham khảo đối với đối tượng phản xạ khuếch tán ảnh, không cần thiết)
(410 . "Model")	410 - Layout tab name (Tên của tab chứa ảnh)
)	

Let's play with it: **Hãy thử với chúng:**

(cdr(assoc 10 enlist)) would return (trả về giá trị) (9.00868 6.59115 0.0)

Remember, CDR returns everything after the first item in a list.

To get the insertion point of the image : Để có được tọa độ điểm nhập vào của hình ảnh:

```
(cdr (assoc 10 enlist))
```

To get the image size in pixels : Để có được cỡ ảnh theo pixel:

```
(setq x(car(cdr(assoc 13 enlist))))
```

```
(setq y(cadr(cdr(assoc 13 enlist))))
```

To get the layer name of the image : Để có được tên lớp chứa ảnh:

```
(cdr(assoc 8 enlist))
```

To get the layout Tab name : Để có được tên Tab chứa ảnh:

```
(cdr(assoc 410 enlist))
```

Insert / Block (Các đối tượng nhập vào và Block)

Typical DXF Group Codes for an INSERT Entity: Các mã nhóm diễn hình của đối tượng nhập vào:

To list an insert entity in AutoCAD you type LIST<enter> and select the block. To do this in AutoLisp you would: (Để liệt kê một đối tượng nhập vào trong AutoCad bạn phải nhập LIST<Enter> và chọn đối tượng đó. Để làm điều này trong Autolisp bạn sẽ thực hiện các hàm:)

```
(setq en(car (entsel "\n Select a block :")))  
(setq enlist(entget en))
```

Would return: (Giá trị trả về sẽ là:)

```
((-1 . <Entity name: 1c90970>) (0 . "INSERT") (5 . "EB6") (100 .  
"AcDbEntity") (67 . 0) (8 . "TX") (100 . "AcDbBlockReference") (2 . "HR")  
(10 1.5 0.5 0.0) (41 . 1.0) (42 . 1.0) (43 . 1.0) (50 . 0.0) (70 . 0) (71 .  
0) (44 . 0.0) (45 . 0.0) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: Tách chúng thành từng phần và sắp xếp lại:

```
(  
  
(-1 . <Entity name:1c90970>) -1 - Entity Name (AutoCAD Automatically takes care of this) (Tên  
đối tượng do AutoCad tự động thực hiện)
```

```
(0 . "INSERT") 0 - Entity Type (Loại đối tượng)
```

```
(1 . "PATH") 1 - Xref path name (Tên đường dẫn tham khảo Xref)
```

Note: GROUP CODE 1 is optional. It is present only if the block is an xref.

Chú ý: Mã nhóm 1 là một định hướng. Nó chỉ có mặt khi block đưa vào là một đối tượng tham khảo Xref.

```
(2 . "HR") 2 - Name of Block (Tên của Block)
```

```
(5 . "EB6") 5 - Handle Name (If handles are turned on) (Tên điều khiển nếu  
chức năng điều khiển được kích hoạt)
```

```
(8 . "TX") 8 - Layer Name (Tên lớp)
```

(10 1.5 0.5 0.0)	10 - Insertion Point (Tọa độ điểm nhập vào)
(41 . 1.0)	41 - X Scale Factor. Optional, 1 = default (Tỉ lệ theo trục X. Đây là một định hướng với giá trị 1 là mặc định)
(42 . 1.0)	42 - Y Scale Factor. Optional, 1 = default (Tỉ lệ theo trục Y. Đây là một định hướng với giá trị 1 là mặc định)
(43 . 1.0)	43 - Z Scale Factor. Optional, 1 = default (Tỉ lệ theo trục Z. Đây là một định hướng với giá trị 1 là mặc định)

(44 . 0.0)	44 - Don't worry about this. Optional. (Đừng quan tâm tới tùy chọn này)
(45 . 0.0)	45 - Don't worry about this. Optional. (Đừng quan tâm tới tùy chọn này)
(50 .5.96143)	50 - Rotation Angle Expressed in Radians (Góc quay theo radians)
(66 . 0)	66 - Flag meaning attributes follow. Optional. (Tiêu lệnh định hướng các thuộc tính đi kèm)

0 = Default (0 là giá trị mặc định)

1 = Attributes-follow flag (1 là có các thuộc tính đi kèm sau Tiêu lệnh)

A series of attribute entities is expected to follow the insert, terminated by a SEQEND entity.

Một loạt các đối tượng thuộc tính được đi kèm khi nhập block và kết thúc bởi đối tượng SEQEND.

(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị đối tượng ở trong không gian model. Bằng 1 là đối tượng thuộc không gian giấy vẽ)
(70 . 0)	70 - Flag See the following: (Tiêu lệnh biểu hiện như sau ☺)

Block flags: (Các Tiêu lệnh của block)

0 = Normal (0 là Bình thường)

1 = This is an anonymous block generated by hatching, associative dimensioning, other internal operations, or an application (Đó là một block ẩn danh được tạo bởi hatching, kích thước tương tác, các hàm nội trú hay một ứng dụng khác)

2 = This block has attribute definitions (một block có thuộc tính xác định)

4 = This block is an external reference (xref) (Một block có sự tham khảo bên ngoài Xref)

8 = This block is an xref overlay (Một block có một đối tượng tham khảo bên ngoài Xref chồng lên)

16 = This block is externally dependent (Một block phụ thuộc bên ngoài)

32 = This is a resolved external reference, or dependent of an external reference (ignored on input)
(Một block kết quả của sự tham chiếu bên ngoài hoặc phụ thuộc vào một sự tham chiếu bên ngoài không lệ thuộc đầu vào)

64 = This definition is a referenced external reference (ignored on input) (việc xác định block này được tham chiếu từ tham chiếu bên ngoài không phụ thuộc đầu vào)

Note: Block flags may be combined. For instance, a 70 DXF group code of 6 would mean the block has attributes (code 2) and is an external reference (code 4). Because $2 + 4 = 6$

Chú ý: Các Tiêu lệnh có thể kết hợp với nhau. Ví dụ, mã nhóm DXF 70 là 6 sẽ có nghĩa là block có các thuộc tính mã số 2 và là một block tham chiếu bên ngoài mã số 4 vì $2 + 4 = 6$

(71 . 0) 71 - Don't worry about this. (Đừng lo về mã này)

(100 . "AcDbEntity") 100 - Don't worry about this (Đừng lo về mã này)

(100 . "AcDbBlockReference") 100 - Don't worry about this (Đừng lo về mã này)

(210 0.0 0.0 1.0) 210 - Don't worry about this Extrusion Direction (Đừng lo về mã Hướng Vuốt dài này)

)

Let's play with it: Hãy thử với các mã này:

(cdr (assoc 10 enlist)) would return (trả về giá trị) (1.5 0.5 0.0)

Remember, CDR returns everything after the first item in a list.

To get the insertion point of the block : Để có được tọa độ điểm nhập vào bản vẽ của block

(cdr (assoc 10 enlist))

To get the name of the block : Để có được tên của block:

(cdr (assoc 2 enlist))

To get the layer name of the block : Để có được tên lớp của block:

(cdr (assoc 8 enlist))

To get the group 70 block flag : Để có được Tiêu lệnh của block ở mã nhóm 70:

```
(cdr (assoc 70 enlist))
```

To get the X scale factor of the block : Để có tỉ lệ theo trục X khi nhập block vào bản vẽ:

```
(cdr (assoc 41 enlist))
```

To see if the entity is indeed a BLOCK entity : Để xem liệu đối tượng có phải là block hay không:

```
(if (= "INSERT" (cdr (assoc 0 enlist)))
```

Line (Đoạn thẳng)

Typical DXF Group Codes for a LINE Entity: Các mã nhóm DXF điển hình cho đối tượng đoạn thẳng:

To list a line in AutoCAD you type LIST<enter> and select the line. To do this in AutoLisp you would: Để liệt kê một đoạn thẳng trong AutoCad bạn phải nhập LIST <Enter> và chọn đoạn thẳng đó. Để làm điều này trong Autolisp bạn phải thực hiện các hàm:

```
(setq en(car (entsel "\n Select a Line: ")))
```

```
(setq enlist(entget en))
```

Would return: Giá trị trả về như sau:

```
((-1 . Entity name: 37b0648) (0 . "LINE") (5 . "89") (100 . "AcDbEntity")  
(67 . 0) (8 . "STR") (100 . "AcDbLine") (10 9.0 7.9375 0.0) (11 11.1213  
10.0588 0.0) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: Hãy tách chúng thành từng phần và sắp xếp lại:

(
(-1 . Entity name: 37b0648)	-1 - AutoCad Entity Name (AutoCAD does this Automatically) (Tên đối tượng trong AutoCad do Auto Cad tự động thực hiện)
(0 . "LINE")	0 - Entity Type (Loại đối tượng)
(5 . "89")	69 - Handle name (if handles are turned on) (Tên điều khiển nếu chức năng điều khiển được kích hoạt)

(8 . "STR")	8 - Name of Layer (Tên lớp chứa đối tượng)
(10 9.0 7.9375 0.0)	10 - Start Point of Line (Tọa độ điểm bắt đầu đoạn thẳng)
(11 11.1213 10.0588 0.0)	11 - End Point of Line (Tọa độ điểm kết thúc đoạn thẳng)
(67 . 0)	Don't worry about this (Đừng lo về mã này)
(100 . "AcDbLine")	Don't worry about this SubClass Marker. (Đừng lo về mã Tạo Phân lớp Bổ sung này)
(100 . "AcDbEntity")	Don't worry about this (Đừng lo về mã này)
(210 0.0 0.0 1.0)	Don't worry about this. This is default unless extruded. (Đừng lo về mã này. Đây là giá trị mặc định trừ khi đối tượng bị vuốt dài)
)	

I'll bet AutoCad doesn't look so mysterious anymore. Does it? Tôi cá rằng AutoCad chẳng còn gì bí mật nữa. Phải không?

Let's play with it: Hãy thử với các mã này:

(cdr(assoc 10 enlist)) would return (trả về giá trị) (9.0 7.9375 0.0)

Remember, CDR returns everything after the first item in a list.

To get the length of the line we need the distance from the start point to the end point :

Để có được độ dài đoạn thẳng, ta cần có khoảng cách từ điểm đầu đến điểm cuối của đoạn thẳng:

(distance (cdr (assoc 10 enlist)) (cdr(assoc 11 enlist)))

To get the angle of the line we need two points. The start point and the end point :

Để có được góc độ của đoạn thẳng, ta cần hai điểm. Điểm bắt đầu và điểm kết thúc:

(angle (cdr(assoc 10 enlist)) (cdr(assoc 11 enlist)))

; Ở đây dùng hàm angle mà tôi chưa hiểu lắm về cú pháp. Ai biết chỉ giùm.

To get the layer name of the line : **Để có được tên lớp của đoạn thẳng:**

```
(cdr (assoc 8 enlist))
```

To see if the entity is indeed a LINE entity : **Để xem đối tượng có thực là đoạn thẳng không:**

```
(if (= "LINE" (cdr (assoc 0 enlist))))
```

LWPolyLine (Đường Polyline Phức hợp)

Typical DXF Group Codes for an LWPOLYLINE Entity: **Các mã nhóm DXF điển hình của đối tượng Polyline phức hợp:**

To list an polyline entity in AutoCAD you type LIST<enter> and select the polyline. To do this in AutoLisp you would: **(Để liệt kê một đối tượng polyline trong AutoCad, bạn nhập LIST<Enter> và chọn đối tượng Polyline đó. Để làm điều này trong Autolisp, bạn phải thực hiện các hàm:**

```
(setq en(car (entsel "\n Select a Polyline :")))
```

```
(setq enlist(entget en))
```

Would return: **(Trả về giá trị:)**

```
((-1 . <Entity name: 1cb0658>) (0 . "LWPOLYLINE") (5 . "83")  
(100 . "AcDbEntity") (67 . 0) (8 . "STR") (100 . "AcDbPolyline") (90 . 5)  
(70 . 0) (43 . 0.0) (38 . 0.0) (39 . 0.0) (10 1.0 1.0) (40 . 0.0) (41 .  
0.0) (42 . 0.0) (10 2.0 1.0) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 2.0 2.0)  
(40 . 0.0) (41 . 0.0) (42 . 0.0) (10 3.0 3.0) (40 . 0.0) (41 . 0.0) (42 .  
0.0) (10 1.0 4.0) (40 . 0.0) (41 . 0.0) (42 . 0.0) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: **Ta tách chúng thành từng thành phần và sắp xếp lại:**

(
(-1 . <Entity name:1cb0658>)	-1 - Entity Name (AutoCAD Automatically takes care of this) (Tên đối tượng do AutoCad tự động thực hiện)
(0 . "LWPOLYLINE")	0 - Entity Type (Loại đối tượng)
(5 . "83")	5 - Handle Name (If handles are turned on) (Tên điều khiển nếu chức năng điều khiển được kích hoạt)
(8 . "STR")	8 - Layer Name (Tên lớp)
(38 . 0.0)	38 - Elevation (optional) Default = 0.0 (Độ dốc tùy chọn, giá

trị mặc định là 0.0)

(39 . 0.0)

39 - Thickness (optional) Default = 0.0 (Độ dày tùy chọn, giá trị mặc định là 0.0)

NOTE:

The next series of codes are Vertex points (group 10 codes) followed by group 40 codes. These codes always appear in the list by order in which they were created

Chú ý: Loạt mã tiếp theo là tọa độ các điểm đỉnh của polyline (các mã nhóm 10) kèm theo các mã nhóm 40. Các mã này luôn luôn xuất hiện trong danh sách theo trật tự mà nó được tạo ra.

10 - Vertex 2D point (tọa độ điểm 2D của đỉnh)

40 - Starting Width of the polyline. This is optional. Default = 0. (Độ rộng bắt đầu của đoạn polyline tùy chọn. Giá trị mặc định là 0.0). Code 40 is not used if group code 43 is set (Mã 40 không sử dụng khi mã nhóm 43 được đặt)

41 - Ending Width of the polyline. This is optional. Default = 0. (Độ rộng kết thúc của đoạn Polyline tùy chọn. Giá trị mặc định là 0.0). Code 41 is not used if group code 43 is set. (Mã 41 không sử dụng nếu mã nhóm 43 được đặt)

42 - Bulge factor. Used to create arcs in the polyline. (Độ cong được sử dụng khi tạo các cung polyline)

43 - Constant Width. Optional. Default = 0. (Độ rộng không thay đổi. Giá trị mặc định là 0)
Code 43 is not used if codes 40 and 41 are set. (Mã này không sử dụng khi đã đặt các mã nhóm 40, 41)

(10 1.0 1.0)	10 - First Vertex (Start Point) (Tọa độ đỉnh thứ nhất tức điểm bắt đầu)
(40 . 0.0)	40 - First Vertex Starting Width (Độ rộng bắt đầu của đoạn polyline thứ nhất)
(41 . 0.0)	41 - First Vertex Ending Width (Độ rộng kết thúc của đoạn polyline thứ nhất)
(42 . 0.0)	42 - First Vertex Bulge Factor (Độ cong của đoạn polyline thứ nhất)

(10 2.0 1.0)	10 - Second Vertex Point (Tọa độ đỉnh thứ hai của Polyline)
(40 . 0.0)	40 - Second Vertex Starting Width (Độ rộng bắt đầu của đoạn polyline thứ hai)
(41 . 0.0)	41 - Second Vertex Ending Width (Độ rộng kết thúc của đoạn polyline thứ hai)
(42 . 0.0)	42 - Second Vertex Bulge Factor (Độ cong của đoạn polyline thứ hai)

(10 2.0 2.0)	10 - Next Vertex Point (Tọa độ đỉnh tiếp theo, thứ ba, của polyline)
(40 . 0.0)	40 - Next Vertex Starting Width (Độ rộng bắt đầu của đoạn polyline tiếp theo)
(41 . 0.0)	41 - Next Vertex Ending Width (Độ rộng kết thúc của đoạn polyline tiếp theo)
(42 . 0.0)	42 - Next Vertex Bulge Factor (Độ cong của đoạn polyline tiếp theo)

(10 3.0 3.0)	10 - Next Vertex Point (Tọa độ đỉnh tiếp theo, thứ tư, của polyline)
(40 . 0.0)	40 - Next Vertex Starting Width (Độ rộng bắt đầu của đoạn polyline tiếp theo)
(41 . 0.0)	41 - Next Vertex Ending Width (Độ rộng kết thúc của đoạn polyline tiếp theo)
(42 . 0.0)	42 - Next Vertex Bulge Factor (Độ cong của đoạn polyline tiếp theo)

(10 1.0 4.0)	10 - Last Vertex Point (Tọa độ đỉnh cuối cùng của polyline)
(40 . 0.0)	40 - Last Vertex Starting Width (Độ rộng bắt đầu của đoạn polyline cuối cùng)
(41 . 0.0)	41 - Last Vertex Ending Width (Độ rộng kết thúc của đoạn polyline cuối cùng)
(42 . 0.0)	42 - Last Vertex Bulge Factor (Độ cong của đoạn polyline cuối cùng)
(43 . 0.0)	43 - Constant Width (Độ rộng không đổi của Polyline)
(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng là nằm trong không gian model, nếu bằng 1 là đối tượng nằm trong không gian giấy vẽ)
(70 . 0)	70 - Polyline Flag. See below. Tiêu lệnh của polyline. Xem phần dưới

FLAGS: (Các Tiêu lệnh)
 0 - Default (Mặc định)
 1 – Closed (Polyline khép kín)
 128 - PLINEGEN (Generation of LineType) (Nguồn gốc từ các loại line)

(90 . 5)	90 - Number of vertices (Số đoạn polyline)
(100 . "AcDbEntity")	100 - Don't worry about this (Không cần quan tâm về mã này)
(100 . "AcDbPolyline")	100 - Don't worry about this (Không cần quan tâm về mã này)
(210 0.0 0.0 1.0)	210 - Don't worry about this Extrusion Factor (Không cần quan tâm về mã đặc tính vuốt dài này)
)	

Let's play with it: Hãy thử với các mã này:

(cdr (assoc 10 enlist)) would return (trả về giá trị) (1.5 0.5 0.0)

Remember, CDR returns everything after the first item in a list.

To get the start point of the LWPolyline : Để có tọa độ điểm bắt đầu của polyline phức hợp:

(cdr (assoc 10 enlist))

Unfortunately, that is as far down as you can dig into the list. To get all of the group 10 codes, you will have to manually search for them.

Không may rằng bạn không thể đào sâu vào danh sách này theo cách đó được. Để có được toàn bộ các mã

nhóm 10, bạn phải tìm kiếm thủ công vậy

Here's an example: Đây là một mẫu cách làm đó:

```
(setq myVertexList(list)) ;create an empty list to store the vertices in. (Tạo một danh sách rỗng để lưu các đoạn polyline vào)

(foreach a enlist ;;--- step through each sub-list (lặp qua tất cả các danh sách con trong danh sách enlist)

(if(= 10 (car a)) ;;--- if the first item in the sub-list equals 10 then (khi mã nhóm là 10 thì thực hiện)

  (setq myVertexList ;;--- reset myVertexList to (đặt lại biến myVertexList)
    (append myVertexList ;;--- the old vertex list (Biến cũ của vòng lặp trước)
      (list ;;--- plus a list containing (cộng thêm biến trong vòng lặp)
        (cdr a) ;;--- the vertex point (Tọa độ của đỉnh polyline)
      ) ;;--- close the list statement (Đóng hàm list)
    ) ;;--- close the append statement (Đóng hàm append)
  ) ;;--- close thesetq statement (Đóng hàm đặt lại biếnsetq)
) ;;--- close the if statement (Đóng hàm if không có else)
) ;;--- close the foreach statement (Đóng hàm foreach)
```

This would return: Kết quả trả về:

```
( (1.0 1.0) (2.0 1.0) (2.0 2.0) (3.0 3.0) (1.0 4.0) )
```

A list of every vertex! Cool. (Một danh sách tọa độ các đỉnh của polyline phức hợp! Tuyệt cú mèo!)

To get the layer name of the LWPolyline : Để có được tên lớp của polyline phức hợp này:

```
(cdr(assoc 8 enlist))
```

To see if the entity is indeed a LWPolyline entity : Để kiểm tra xem đây có thực là một đối tượng polyline phức hợp không:

```
(if (= "LWPOLYLINE" (cdr(assoc 0 enlist)))
  (princ "\n It is a LWPolyLine")
  (princ "\n It is not a LWPolyLine")
)
```

Mline (Tổ hợp các đoạn thẳng)

Typical DXF Group Codes for a MLine Entity: Các mã nhóm DXF điển hình của một đối tượng tổ hợp các đoạn thẳng:

To list a MLine in AutoCAD you type LIST<enter> and select the MLine. To do this in AutoLisp you would:

Để liệt kê các mã của một tổ hợp các đoạn thẳng trong AutoCad bạn nhập LIST<Enter> và chọn tổ hợp đoạn thẳng đó. Để làm điều này trong Autolisp bạn phải thực hiện các hàm:

```
(setq en(car (entsel "\n Select an MLine: ")))
```

```
(setq enlist(entget en))
```

Would return something like this : Giá trị trả về như sau:

```
((-1 . <Entity name: 40222978>) (0 . "MLINE") (330 . <Entity name: 40073cf8>) (5 . "92F") (100 .  
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbMline") (2 . "STANDARD") (340 . <Entity  
name: 40073cc0>) (40 . 1.0) (70 . 0) (71 . 3) (72 . 4) (73 . 2) (10 11.4446 15.392 0.0) (210 0.0 0.0 1.0) (11  
11.4446 15.392 0.0) (12 1.0 0.0 0.0) (13 0.690073 0.72374 0.0) (74 . 2) (41 . 0.0) (41 . 0.0) (75 . 0) (74 . 2) (41  
-1.38171) (41 . 0.0) (75 . 0) (11 30.3875 15.392 0.0) (12 0.0 1.0 0.0) (13 -0.707107 0.707107 0.0) (74 . 2)  
(41 . 0.0) (41 . 0.0) (75 . 0) (74 . 2) (41 . -1.41421) (41 . 0.0) (75 . 0) (11 30.3875 31.4532 0.0) (12 -1.0 0.0 0.0)  
(13 -0.707107 -0.707107 0.0) (74 . 2) (41 . 0.0) (41 . 0.0) (75 . 0) (74 . 2) (41 . -1.41421) (41 . 0.0) (75 . 0) (11  
10.6793 31.4532 0.0) (12 0.0475993 -0.998867 0.0) (13 0.72374 -0.690073 0.0) (74 . 2) (41 . 0.0) (41 . 0.0)  
(75 . 0) (74 . 2) (41 . -1.44912) (41 . 0.0) (75 . 0))
```

Let's take it apart and put it in order: Tách chúng thành từng phần và sắp xếp lại:

(
(-1 . <Entity name: 40222978>)	-1 - Entity Name (Tên đối tượng)
(0 . "MLINE")	0 - Entity Type (Loại đối tượng)
(2 . "STANDARD")	2 - Style (Must exist in MLineStyle Dictionary) (Kiểu đoạn thẳng, phải có trong thư viện kiểu tổ hợp đoạn thẳng)
(5 . "92F")	5 - Handle (If handles are turned on) (Mã điều khiển nếu chức năng điều khiển được kích hoạt)
(8 . "0")	8 - Layer Name (Tên lớp)
(40 . 1.0)	40 - Scale Factor (Tỷ lệ)
(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hoặc bằng không chỉ

thị rằng đối tượng ở trong không gian model, bằng 1 chỉ thị rằng đối tượng ở trong không gian giấy vẽ)

(70 . 0)

70- Justification: 0 = Top, 1 = Zero, 2 = Bottom (Căn chỉnh: 0 = Trên cùng, 1 = không căn chỉnh, 2 = Dưới cùng)

(71 . 3)

71 - Flags - 1=Unlocked, 2=Closed, 4=Suppress start caps 8=Suppress end caps. (Tiêu lệnh: 1 = mở khóa, 2 = Khép kín, 4 = Chặn điểm đầu, 8 = Chặn điểm cuối)

(72 . 4)

72 - Number of vertices (Số đỉnh)

(73 . 2)

73 - Number of elements in MLineStyle Definition (Số phần tử khi xác định kiểu tổ hợp đoạn thẳng)

(10 11.4446 15.392 0.0)

10 - Start Point (Tọa độ điểm bắt đầu)

----- Start the Vertex Loop (Bắt đầu vòng lặp qua các đỉnh) -----

Note: Each vertex will have a code 11, 12, 13, 74, 41, and 75. I'm going to include the first appearance of codes 12, 13, 74, and 75 and delete the rest for clarity. I'll explain later.

Chú ý: Mỗi đỉnh sẽ có một mã 11, 12, 13, 74, 41, và 75. Tôi sẽ mô tả các mã 12, 13, 74 và 75 trong lần xuất hiện đầu tiên và xóa bỏ các phần còn lại để cho rõ ràng. Tôi sẽ giải thích điều này sau.

(11 11.4446 15.392 0.0)

11 - First Vertex point of multiple entries (Tọa độ đỉnh đầu tiên của đối tượng tổ hợp)

(12 1.0 0.0 0.0)

12 - Direction Vector for next line from this point (Vector định hướng của đoạn thẳng kế tiếp đỉnh này)

(13 0.690073 0.72374 0.0)

13 - Direction Vector of miter at this vertex (vector định hướng của chóp tại đỉnh này)

(74 . 2)

74 - Number of group 41's to follow (I'll explain later) (Số mã nhóm 41 đi kèm, điều này sẽ được giải thích sau)

(41 . 0.0)

41 - First code 41 (Mã nhóm 41 đầu tiên)

(41 . 0.0)

41 - Second code 41 (Mã nhóm 41 thứ hai)

(75 . 0)

75 - Number of group 42's to follow (fill parameters) (Số mã nhóm 42 đi kèm (các thông số điền đây))

(11 30.3875 15.392 0.0)	11 - Second vertex point (Tọa độ đỉnh thứ hai)
(12 0.0 1.0 0.0)	12 - Second Direction vector for next line (Vector định hướng thứ hai cho đoạn thẳng kế tiếp)
(13 -0.707107 0.707107 0.0)	13 - Second Direction for miter (Định hướng thứ hai cho chóp)
(11 30.3875 31.4532 0.0)	11 - Third vertex point (Tọa độ đỉnh thứ ba)
(12 -1.0 0.0 0.0)	12 - Third Direction vector for next line (Vector định hướng thứ ba cho đoạn thẳng kế tiếp)
(13 -0.707107 -0.707107 0.0)	13 - Third Direction for miter (Định hướng thứ ba cho chóp)
(11 10.6793 31.4532 0.0)	11 - Fourth vertex point (Tọa độ đỉnh thứ tư)
(12 0.0475993 -0.998867 0.0)	12 - Fourth Direction vector for next line (Vector định hướng thứ tư cho đoạn thẳng kế tiếp)
(13 0.72374 -0.690073 0.0)	13 - Fourth Direction for miter (Định hướng thứ tư cho chóp)
(100 . "AcDbEntity")	100 – Ignore (Không cần quan tâm)
(100 . "AcDbMline")	100 - Ignore this subclass marker (Không cần quan tâm mã tạo phân lớp phụ này)
(210 0.0 0.0 1.0)	210 - Ignore extrusion direction marker (Không cần quan tâm mã tạo hướng vuốt này)
(330 . <Entity name: 40073cf8>)	330 – Ignore (Không cần quan tâm)
(340 . <Entity name: 40073cc0>)	340 - Ignore (needed to modify MLineStyle. See note below. (Không cần quan tâm)(chỉ cần để sửa đổi kiểu tổ hợp đoạn thẳng, xem ghi chú dưới đây)
(410 . "Model")	410 - Layout Tab Name (Tên Tab chứa bản vẽ)
)	

Notes: From the book...

Chú ý: Theo tài liệu

The group code 41 may contain zero or more items. The first group code 41 value is the distance from the segment vertex along the miter vector to the point where the line element's path intersects the miter vector. The next group code 41 value is the distance along the line element's path from the point defined by the first group 41 to the actual start of the line element. The next is the distance from the start of the line element to the first break (or cut) in the line element. The successive group code 41 values continue to list the start and stop points of the line element in this segment of the mline. Linetypes do not affect group 41 lists.

Mã nhóm 41 có thể chứa không hoặc nhiều khoản mục. Mã nhóm 41 đầu tiên có giá trị là khoảng cách từ đỉnh phân đoạn dọc theo vector chớp tới điểm nơi mà đường đi của phân đoạn thẳng cắt vector chớp. Giá trị mã nhóm 41 kế tiếp là khoảng cách dọc theo đường đi của phân đoạn thẳng từ điểm được xác định bởi mã nhóm 41 đầu tiên tới điểm bắt đầu thực tế của phân đoạn thẳng. Tiếp theo là khoảng cách từ điểm bắt đầu của phân đoạn thẳng tới điểm ngắt hay điểm cắt đầu tiên trên phân đoạn thẳng này. Các giá trị mã nhóm 41 tiếp tục liệt kê các điểm bắt đầu và kết thúc của phân đoạn thẳng trong phân đoạn này của tổ hợp các đoạn thẳng. Các loại đoạn thẳng không có ảnh hưởng tới các danh sách mã nhóm 41.

The 2 group codes in mline entities and mlinestyle objects are redundant fields. These groups should not be modified under any circumstances, although it is safe to read them and use their values. The correct fields to modify are as follows:

Các mã nhóm 2 trong các đối tượng tổ hợp các đoạn thẳng và các đối tượng kiểu tổ hợp đoạn thẳng là các trường dư. Các nhóm này không được sửa đổi dưới bất kỳ chu trình nào, mặc dầu nó rất an toàn đối với việc đọc và sử dụng các giá trị của nó. Các trường chính sửa để sửa đổi là như sau:

Mline - The 340 group in the same object, which indicates the proper MLINESTYLE object.

Với đối tượng tổ hợp các đoạn thẳng Mline – Nhóm mã 340 trong cùng đối tượng sẽ chỉ thị đối tượng kiểu tổ hợp đoạn thẳng thích hợp

Mlinestyle -The 3 group value in the MLINESTYLE dictionary which precedes the 350 group that has the handle or entity name of the current mlinestyle.

Với đối tượng kiểu tổ hợp đoạn thẳng MlineStyle – Giá trị nhóm mã 3 trong thư viện kiểu tổ hợp đoạn thẳng đứng trước nhóm mã 350 là mã có tên điều khiển hoặc tên đối tượng của kiểu tổ hợp đoạn thẳng hiện tại.

Here's an example of how to get all of the vertices in a list: Đây là một ví dụ về cách có được toàn bộ các tọa độ đỉnh trong một danh sách:

```
(setq en(car(entsel "\n Select a MLine: ")));;--- Get the entity name (Lấy tên đối tượng)

(setq enlist(entget en));;--- Get the dxf group codes (Lấy các mã nhóm DXF)

(setq myVertexList(list));;--- create an empty list to store the vertices in. (tạo một
```

danh sách rỗng để lưu các tọa độ đỉnh)

```
(setq v1(cdr(assoc 10 enlist)))      ;;;--- Get the starting point of the mline (lấy tọa độ điểm  
                                     bắt đầu của tổ hợp các đoạn thẳng)  
  
(setq myVertexList(append myVertexList (list v1)))      ;;;--- Add the point to the list  
                                                         (Thêm tọa độ đỉnh vào danh sách)  
  
(foreach a enlist      ;;;--- step through each sub-list (Lặp qua các danh sách phụ)  
  
(if(= 11 (car a))      ;;;--- if the first item in the sub-list equals 11 then (Khi mã nhóm  
                        là 11 thì thực hiện)  
  
    (setq myVertexList      ;;;--- reset myVertexList to (Đặt lại giá trị biến myVertexList)  
      (append myVertexList      ;;;--- the old vertex list (Danh sách tọa độ cũ)  
        (list      ;;;--- plus a list containing (cộng thêm một danh sách)  
          (cdr a)      ;;;--- the vertex point (tọa độ đỉnh)  
        )      ;;;--- close the list statement (Đóng hàm list)  
      )      ;;;--- close the append statement (Đóng hàm append)  
    )      ;;;--- close the setq statement (đóng hàm đặt lại giá trị setq)  
  )      ;;;--- close the if statement (Đóng hàm if không có else)  
)      ;;;--- close the foreach statement (Đóng hàm lặp foreach)
```

This would return: (Kết quả trả về)

```
( (11.4446 15.392 0.0) (11.4446 15.392 0.0) (30.3875 15.392 0.0) (30.3875  
31.4532 0.0) (10.6793 31.4532 0.0) )
```

A list of every vertex! Cool.(Một danh sách tọa độ của mọi đỉnh! Tuyệt vời)

Extra...Let's redraw the MLine with Lines: (Mở rộng ...Hãy vẽ lại tổ hợp đoạn thẳng với các lệnh Line)

```
(command "line" (car myVertexList))      ;;;--- Start the line command with the first vertex (Bắt  
                                           đầu lệnh Line với đỉnh đầu tiên)  
  
(foreach a (cdr myVertexList)      ;;;--- Cycle through the rest of the vertex list (Lặp qua  
                                     các đỉnh còn lại trong danh sách các đỉnh)  
  
    (command a)      ;;;--- Send the vertex to the line command (Đưa đỉnh vào lệnh Line)  
  
) ;Kết thúc hàm Foreach  
  
(command)      ;;;--- End the line command (Kết thúc lệnh Line)
```

Mtext (Tổ hợp văn bản)

Typical DXF Group Codes for an MText Entity: Các mã nhóm diễn hình của đối tượng tổ hợp văn bản :

To list MText in AutoCAD you type LIST<enter> and select the MText. To do this in AutoLisp you would: Để liệt kê các mã nhóm DXF của một tổ hợp văn bản trong AutoCad, bạn nhập LIST<Enter> và chọn tổ hợp văn bản đó. Để thực hiện điều này trong Autolisp bạn phải thực hiện các hàm sau:

```
(setq en(car (entsel "\n Select MText: ")))  
(setq enlist(entget en))
```

Would return: Kết quả trả về:

```
((-1 . <Entity name: 40222a10>) (0 . "MTEXT") (330 . <Entity name: 40073cf8>) (5 . "942") (100 .  
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbMText") (10 37.2758 63.7667 0.0) (40 . 0.2) (41  
. 54.9151) (71 . 1) (72 . 5) (1 . "This is an Mtext string. ") (7 . "Standard") (210 0.0 0.0 1.0) (11 1.0 0.0 0.0) (42  
. 4.2) (43 . 0.266667) (50 . 0.0) (73 . 1) (44 . 1.0))
```

Let's take it apart and put it in order: Tách chúng thành từng phần và sắp xếp lại:

(
(-1 . <Entity name: 40222a10>)	-1 - Entity Name (Tên đối tượng)
(0 . "MTEXT")	0 - Entity Type (Loại đối tượng)
(1 . "This is an Mtext string. ")	1 - Text Value (See Note 1 and 2 Below!!!) (Giá trị của văn bản, xem các chú ý 1 và 2 bên dưới)
(5 . "942")	5 - Handle (If handles are turned on) (Mã điều khiển nếu chức năng điều khiển được kích hoạt)
(7 . "Standard")	7 - Text Style (Kiểu văn bản)
(8 . "0")	8 - Layer Name (Tên lớp)
(10 37.2758 63.7667 0.0)	10 - Insertion Point (Tọa độ điểm nhập văn bản)
(11 1.0 0.0 0.0)	11 - X Axis Direction Vector (Vector định hướng theo trục X)
(40 . 0.2)	40 - Initial Text Height (Độ cao của văn bản)
(41 . 54.9151)	41 - Reference Rectangle Width (Độ rộng tham khảo của ô văn bản)

(42 . 4.2)	42 - Horizontal width of the characters (Ignored) (Độ rộng của các ký tự. Không cần lưu ý)
(43 . 0.266667)	43 - Vertical Height of the MText entity (Ignored) (Độ cao của đối tượng tổ hợp văn bản)
(44 . 1.0)	44 - Line Spacing factor (Optional) (Khe hở giữa các dòng văn bản)
(50 . 0.0)	50 - Rotation angle (Always in radians) (Góc quay theo radians)
(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng thuộc không gian model, bằng 1 là thuộc không gian giấy vẽ)
(71 . 1)	71 - Attachment point 1=Top 2=Top Center 3= Top Right 4=Middle Left 5=Middle Center 6=Middle Right 7=Bottom Left 8= Bottom Center 9=Bottom Right Điểm căn chỉnh : 1 là căn trên, 2 là căn tâm bên trên, 3 là căn phải bên trên, 4 là căn trái giữa, 5 là căn giữa tâm, 6 là căn phải giữa, 7 là căn trái đáy, 8 là căn đáy tâm, 9 là căn phải đáy
(72 . 5)	72 - Drawing Direction 1 = Left to Right 3 = Top to Bottom 5 = Use Text Style Default Hướng vẽ : 1 là từ trái qua phải, 3 là từ trên xuống dưới, 5 là sử dụng giá trị mặc định của kiểu văn bản
(73 . 1)	73 - Line Spacing Style (Kiểu khe hở giữa các dòng) 1 = At Least (Taller characters will override) (Tối thiểu) 2 = Exact (Taller characters will not override) (Chính xác)
(100 . "AcDbEntity")	100 - Ignore (không cần quan tâm)
(100 . "AcDbMText")	100 - Ignore (không cần quan tâm)
(210 0.0 0.0 1.0)	210 - Extrusion direction (Ignore) (Hướng vuốt không cần chú ý)
(330 . <Entity name: 40073cf8>)	330 - Ignore (Không cần quan tâm)
(410 . "Model")	410 - Tab Layout Name (Tên tab chứa tổ hợp văn bản)
)	

NOTE 1 : Formatting! CHÚ Ý 1: Định dạng văn bản

MText can contain formatting characters. For instance, if I change the MText entity we used in the example above: (Tổ hợp văn bản có chứa các ký tự định dạng. Ví dụ, khi tôi thay đổi đối tượng tổ hợp văn bản mà tôi đã sử dụng trong ví dụ trên:)

```
(1 . "This is an Mtext string. ")
```

to use the Arial font, underlined, and bold, the MText value would look something like this: (Để sử dụng font Arial, gạch dưới, chữ đậm, giá trị của tổ hợp văn bản sẽ trông giống như sau:

```
(1 . "{\\fArial|b1|i0|c0|p34;\\LThis is an Mtext string. \\Ftxt.shx; }")
```

Not very pretty is it? (Không dễ thương lắm hả?)

Mtext formatting codes: Các mã nhóm định dạng tổ hợp văn bản

<u>Format code</u>	<u>Purpose</u>
<code>\O... \o</code>	Turns overline on and off (Bật hoặc tắt chức năng chèn dòng)
<code>\L... \l</code>	Turns underline on and off (Bật hoặc tắt chức năng gạch dưới)
<code>\~</code>	Inserts a nonbreaking space (Đưa vào một không gian liên tục)
<code>\\</code>	Inserts a backslash (Đưa một ký tự “\” vào văn bản)
<code>\{... \}</code>	Inserts an opening and closing brace (Đưa một cặp ngoặc móc đóng và mở vào văn bản)
<code>\File name;</code>	Changes to the specified font file (Thay đổi file font được mô tả)
<code>\Hvalue;</code>	Changes to the text height specified in drawing units (Thay đổi độ cao văn bản theo đơn vị bản vẽ)
<code>\Hvaluex;</code>	Changes the text height to a multiple of the current text height (Thay đổi độ cao văn bản theo bội số của độ cao văn bản hiện tại)
<code>\S... ^... ;</code>	Stacks the subsequent text at the \, #, or ^ symbol (Chồng văn bản hệ quả vào chỗ các ký hiệu \, #, hay ^)
<code>\Tvalue;</code>	Adjusts the space between characters, from .75 to 4 times (Điều chỉnh không gian giữa các ký tự từ 0.75 đến 4 lần)
<code>\Qangle;</code>	Changes obliquing angle (Thay đổi góc nghiêng của ký tự)
<code>\Wvalue;</code>	Changes width factor to produce wide text (Thay đổi độ rộng ký tự để tạo văn bản rộng)
<code>\A</code>	Sets the alignment value; valid values: 0, 1, 2 (bottom, center, top) (Đặt giá trị căn dòng, các giá trị cho phép là 0,1,2 tương ứng căn đáy, căn giữa và căn trên)

Note: Use curly braces ({ }) to apply a format change only to the text within the braces. You can nest braces up to eight levels deep.

Chú ý: Sử dụng ngoặc móc ({ }) để áp dụng việc thay đổi định dạng cho đoạn văn bản có trong ngoặc. Bạn có thể tổ hợp các ngoặc móc tới 8 cấp về độ sâu cho việc thay đổi định dạng.

NOTE 2 : String Length! CHÚ Ý 2: Độ dài chuỗi!

If the number of characters in an MText entity is less than 250, all characters will be in group code number

1. If you exceed 249 characters, the characters are broken up in 250 character chunks and placed under group code number 3. The remainder of the characters will be in group code 1. For instance, if I change the MText entity we used in the example above:

Khi số ký tự trong một đối tượng tổ hợp văn bản nhỏ hơn 250, tất cả các ký tự sẽ được nhóm trong mã nhóm số 1. Khi vượt quá 249 ký tự, các ký tự sẽ bị ngắt thành từng xâu 250 ký tự và đặt trong mã nhóm số 3. Phần các ký tự còn lại sẽ được đặt trong mã nhóm 1. Ví dụ: khi tôi thay đổi đối tượng tổ hợp văn bản đã được sử dụng trong ví dụ trên:

```
(1 . "This is an Mtext string. ")
```

into a string that is 600 characters long, the MText value would look something like this: (thành một chuỗi dài 600 ký tự, giá trị tổ hợp văn bản sẽ trông giống như sau:

```
(1 .  
"1234567899123456789012345678911234567892123456789312345678941234567895123  
45678961234567897123456789812345678991234567890 ")
```

Where's the rest of the string? It is stored in the group 3 codes. There are two of them that are 250 characters each. (Phần còn lại của chuỗi ở đâu? Nó được lưu trong các mã nhóm 3. Có hai nhóm như vậy do mỗi nhóm là 250 ký tự)

```
(3 .  
"1234567890123456789112345678921234567893123456789412345678951234567896123456789  
71234567898123456789912345678901234567891123456789212345678931234567894123456789  
51234567896123456789712345678981234567899123456789012345678911234567892123456789  
31234567894")
```

```
(3 .  
"1234567895123456789612345678971234567898123456789912345678901234567890123456789  
11234567892123456789312345678941234567895123456789612345678971234567898123456789  
91234567890123456789112345678921234567893123456789412345678951234567896123456789  
71234567898")
```

To put the string back together, you would have to use the STRCAT function on the group 3 codes first: (Để đặt các chuỗi trở lại với nhau, bạn phải sử dụng hàm strcat trên các mã nhóm 3 trước)

```
(setq myStr "") ;Đặt biến myStr là một chuỗi trống
```

```
(setq en(car(entsel "\n Select Mtext: "))) ;Đặt biến en là tên đối tượng tổ hợp văn bản
```

```
(setq enlist(entget en)) ;Đặt biến enlist là danh sách các mã nhóm DXF của đối tượng en
```

```
(foreach a enlist ;Lặp qua tất cả các danh sách con a trong danh sách enlist
```



```
(if (= 3 (car a)) ; Nếu mã nhóm DXF của danh sách con a bằng 3
```

```
(setq myStr (strcat myStr (cdr a))) ; Đặt lại biến myStr là chuỗi tổ hợp của biến myStr  
; cũ và chuỗi mới được xác định bởi hàm (cdr a) trong vòng lặp hiện tại.
```

```
) ; Kết thúc hàm if
```

```
) ; Kết thúc hàm lặp Foreach
```

Then you would have to tack the string stored in group code 1 to the end. Like this: (Sau đó bạn phải nối tiếp vào đuôi chuỗi được lưu trong mã nhóm 1 như sau:)

```
(setq myStr(strcat myStr (cdr(assoc 1 enlist))))
```

Let's play with it: Hãy thử với các mã này:

```
(cdr(assoc 10 enlist)) would return (trả về giá trị) (37.2758 63.7667 0.0)
```

Remember, CDR returns everything after the first item in a list. (Nhớ là hàm CDR trả về giá trị là toàn bộ danh sách trừ khoản mục đầu tiên của danh sách)

To get the initial height of the MText : Để có độ cao ban đầu của đối tượng tổ hợp văn bản

```
(cdr (assoc 40 enlist))
```

To get the insertion : Để có tọa độ điểm nhập đối tượng vào:

```
(cdr(assoc 10 enlist))
```

To get the layer name of the MText : Để có được tên lớp của đối tượng tổ hợp văn bản:

```
(cdr(assoc 8 enlist))
```

To get the text value of the MText : Để có được giá trị văn bản của đối tượng tổ hợp văn bản

```
(cdr(assoc 1 enlist)) ;if it less than 250 characters long (See Note 1 Above)
```

To get the rotation angle of the MText : Để có được góc quay của đối tượng tổ hợp văn bản:

```
(cdr(assoc 50 enlist))
```

To see if the entity is indeed a MText entity : Để kiểm tra đối tượng có thực là đối tượng tổ hợp văn bản

```
(if (= "MTEXT" (cdr (assoc 0 enlist))) ; Hàm này chưa hoàn thiện
```

Point / Node (Điểm và tọa độ điểm)

Typical DXF Group Codes for an Point or Node Entity: Các mã nhóm DXF điển hình cho một đối tượng điểm và tọa độ điểm:

To list a point entity in AutoCAD you type LIST<enter> and select the point. To do this in AutoLisp you would: (Để liệt kê các mã nhóm DXF của một đối tượng điểm trong AutoCad bạn nhập LIST<Enter> và chọn điểm đó. Trong Autolisp bạn phải thực hiện các hàm sau:

```
(setq en(car (entsel "\n Select a Point: ")))  
(setq enlist(entget en))
```

Would return: (Kết quả trả về:)

```
((-1 . <Entity name: 4009ff78>) (0 . "POINT") (330 . <Entity name:  
40073cf8>) (5 . "267") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 .  
"0") (100 . "AcDbPoint") (10 4.27913 8.26876 0.0) (210 0.0 0.0 1.0) (50 .  
0.0))
```

Let's take it apart and put it in order: Tách rời từng phần và sắp xếp lại:

```
(  
  
  (-1 . <Entity name: 4009ff78>)    -1 - Entity Name (AutoCAD Automatically takes care of  
                                     this) (Tên đối tượng do AutoCad tự động thực hiện)  
  
  (0 . "POINT")                    0 - Entity Type (Loại đối tượng)  
  
  (5 . "267")                      5 - Handle Name (If handles are turned on) (Tên điều khiển  
                                     nếu chức năng điều khiển được kích hoạt)  
  
  (8 . "0")                        8 - Layer Name (Tên lớp)  
  
  (10 4.27913 8.26876 0.0)          10 - Insertion Point (Tọa độ điểm nhập vào)  
  
  (50 . 0.0)                      50 - Angle of the X axis for the UCS in effect when the  
                                     point was drawn if PdMode does not = 0 (default = 0)  
                                     (Góc với trục X trong hệ tọa độ UCS khi điểm được vẽ  
                                     với Pmode khác 0, giá trị mặc định là 0)  
  
  (67 . 0)                        67 - Absent or zero indicates the entity is in model space.
```

If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng thuộc không gian model, bằng 1 là đối tượng thuộc không gian giấy vẽ)

```
(100 . "AcDbEntity")          100 - Ignore
(100 . "AcDbPoint")           100 - Ignore
(210 0.0 0.0 1.0)              210 - Ignore extrusion direction
(330 . <Entity name: 40073cf8>) 330 - Ignore
(410 . "Model")                410 - Layout tab name (tên tab chứa bản vẽ)
)
```

Let's play with it: Thử nghiệm với mấy chú này nhé:

`(cdr (assoc 10 enlist))` would return (trả về giá trị) `(4.27913 8.26876 0.0)`

Remember, CDR returns everything after the first item in a list.

To get the insertion point of the Point : Để có tọa độ điểm nhập vào của một điểm:

`(cdr (assoc 10 enlist))` would return (trả về giá trị): `(4.27913 8.26876 0.0)`

There's not a lot of data associated with a point! Có hàng lô xích xông các dữ liệu tương tác với tọa độ điểm.

Polyline (Các Polyline)

Typical DXF Group Codes for a Polyline Entity: Các mã nhóm diễn hình của đối tượng polyline:

To list a polyline entity in AutoCAD you type LIST<enter> and select the polyline. To do this in AutoLisp you would: (Để liệt kê các mã nhóm DXF của một đối tượng Polyline trong AutoCad bạn nhập List<Enter> và chọn đối tượng polyline đó. Trong Autolisp bạn sẽ thực hiện điều đó như sau:)

```
(setq en(car (entsel "\n Select a PolyLine: ")))
(setq enlist(entget en))
```

Would return: (Kết quả trả về:)

```
((-1 . <Entity name: 40222a38>) (0 . "POLYLINE") (330 . <Entity name:
40073cf8>) (5 . "947") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 .
"0") (100 . "AcDb2dPolyline") (66 . 1) (10 0.0 0.0 0.0) (70 . 4) (40 . 0.0)
(41 . 0.0) (210 0.0 0.0 1.0) (71 . 0) (72 . 0) (73 . 0) (74 . 0) (75 . 6))
```

Let's take it apart and put it in order: **Tách rời chúng và sắp xếp lại:**

(
(-1 . <Entity name: 40222a38>)	-1 - Entity Name (Tên đối tượng)
(0 . "POLYLINE")	0 - Entity Type (Loại đối tượng)
(5 . "947")	5 - Handle (If handles are turned on.) (Tên điều khiển nếu chức năng điều khiển được kích hoạt)
(8 . "0")	8 - Layer Name (Tên lớp)
(10 0.0 0.0 0.0)	10 - Always Zero (luôn luôn là không)
(40 . 0.0)	40 - Default Start Width (Default = 0) (Độ rộng bắt đầu, giá trị mặc định là 0)
(41 . 0.0)	41 - Default End Width (Default = 0) (Độ rộng kết thúc, giá trị mặc định là 0)
(66 . 1)	66 - Vertices follow flag (Always 1 for Polylines) (Tiêu lệnh đi kèm các đỉnh, với polyline giá trị này luôn là 1)
(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng thuộc không gian model, bằng 1 là đối tượng thuộc không gian giấy vẽ)
(70 . 4)	70 - PolyLine flag 0 = Default (No flags) (Tiêu lệnh Polyline mặc định là 0) 1 = This is a closed polyline or mesh closed in M direction. (Đây là một polyline hay một lưới Khép kín) 2 = Curve fit Vertices have been added (Đường cong phù hợp với các đỉnh đã thêm vào) 4 = Spline fit Vertices have been added (Đường Spline phù hợp với các đỉnh đã vẽ) 8 = This is a 3D Polyline (Đây là một polyline 3D) 16 = This is a 3D polygon Mesh (Đây là một lưới đa giác 3D) 32 = This polygon Mesh is Closed in the N Direction (Đây là một lưới đa giác khép kín theo hướng N) 64 = This polygon is a polygon Mesh (Đây là một lưới đa giác) 128 = Generate Line Type Continuously around Vertices (Khởi

tạo các đoạn thẳng xung quanh các đỉnh)

(71 . 0)	71 - Polygon mesh M vertex count (Optional) (Tùy chọn đếm các đỉnh M của lưới đa giác)
(72 . 0)	72 - Polygon mesh N vertex count (Optional) (Tùy chọn đếm các đỉnh N của lưới đa giác)
(73 . 0)	73 - Smooth Surface M Density (Optional) (Tùy chọn mật độ M của bề mặt lưới)
(74 . 0)	74 - Smooth Surface N Density (Optional) (Tùy chọn mật độ N của bề mặt lưới)
(75 . 6)	75 - Curves and Smooth Surface type (Optional) (tùy chọn loại mặt lưới và cong) 0 = No Smooth Surface Fitted (Không có mặt lưới phù hợp) 5 = Quadratic B-Spline Surface (Bề mặt B-Spline Quadratic) 6 = Cubic B-Spline Surface (Bề mặt B-Spline lập phương) 8 = Bezier Surface (Bề mặt Bezier)
(100 . "AcDbEntity")	100 - Ignore
(100 . "AcDb2dPolyline")	100 - Ignore
(210 0.0 0.0 1.0)	210 - Extrusion Direction (Ignore)
(330 . <Entity name: 40073cf8>)	330 - Ignore
(410 . "Model")	410 - Tab Layout Name (Tên tab chứa bản vẽ)
)	

Notice there are no vertices shown. You have to step through this entity looking at the sub-entities for the vertices until you come across the SEQEND entity. Sound familiar? It should. This is exactly like the Attribute Entity. Pretty much anyway. Please read the [Attribute](#) section above for instructions on the ENTNEXT function used to step into an entity looking for sub-entities and the SEQEND entity definition. When you get through, come back to here.

Chú ý không có các đỉnh được chỉ ra. Bạn phải bước qua từng đối tượng này để tìm kiếm các đối tượng phụ đối với các đỉnh cho đến khi bạn đạt được đối tượng SEQEND. Nghe quen quá hả? Đúng vậy. Nó chính xác giống như đối tượng thuộc tính. Dù sao cũng quá dễ thương. Bạn hãy đọc lại đoạn các đối tượng thuộc tính ở trên về các chỉ dẫn dùng hàm ENTNEXT từng bước thâm nhập đối tượng để tìm kiếm các đối tượng phụ và xác định đối tượng SEQEND. Khi bạn đã nắm được vấn đề, hãy quay trở lại đây.

So, we now know how to step into an entity. Let's do that for this polyline and save a list of all of the vertices.

Vậy là ta đã biết cách xâm nhập vào đối tượng. Hãy thực hiện điều đó với Polyline này và lưu lại một danh sách tất cả các đỉnh.

```
(defun C:PTEST()

  ;;;--- Get the entity's name (lấy tên đối tượng)
  (setq en(car(entsel "\n Select a PolyLine: ")))

  ;;;--- Get the DXF group codes of the entity (lấy các mã nhóm DXF của đối tượng)
  (setq enlist(entget en))

  ;;;--- Create an empty list to hold the points (tạo một danh sách rỗng để lưu các
  tọa độ điểm)
  (setq ptList(list))

  ;;;--- Get the sub-entities name (lấy tên đối tượng phụ)
  (setq en2(entnext en))

  ;;;--- Get the dxf group codes of the sub-entity (lấy các mã nhóm DXF của đối
  tượng phụ)
  (setq enlist2(entget en2))

  ;;;--- While the polyline has a next vertice (Trong khi polyline có đỉnh tiếp theo)
  (while (not (equal (cdr(assoc 0 (entget(entnext en2)))) "SEQEND"))

    ;;;--- Get the next sub-entity (lấy đối tượng phụ tiếp theo)
    (setq en2(entnext en2))

    ;;;--- Get its dxf group codes (Lấy mã nhóm DXF của nó)
    (setq enlist2(entget en2))

    ;;;--- Check to make sure it is not a spline reference point (Kiểm soát
    để chắc chắn đó không phải là điểm Spline tham chiếu)
    (if(/= 16 (cdr(assoc 70 enlist2)))

      ;;;--- It is a vertex, save the point in a list [ptlist] (lưu tọa độ
      điểm nếu nó là một đỉnh)
      (setq ptList(append ptList (list (cdr(assoc 10 enlist2))))))

  )

)

(princ)

)
```

If you run this program and then type this in the command line: (Nếu bạn chạy chương trình và sau đó nhập

vào dòng nhắc lệnh:)

```
Command: ptList!<enter>
```

It will return a list of all of the vertices and will look something like this: (Nó sẽ trả về một danh sách tất cả các đỉnh và trông giống như sau:)

```
((48.4773 56.3423 0.0) (47.9891 57.1226 0.0) (47.7463 57.7535 0.0) (47.7072 58.2456 0.0) (47.8304 58.6095 0.0) (48.0741 58.8559 0.0) (48.3968 58.9954 0.0) (48.7568 59.0385 0.0) (49.1125 58.9959 0.0) (49.4264 58.8788 0.0) (49.6767 58.7011 0.0) (49.8457 58.4773 0.0) (49.9157 58.2219 0.0) (49.869 57.9495 0.0) (49.688 57.6745 0.0) (49.355 57.4114 0.0) (48.8522 57.1748 0.0))
```

Let's play with it: Thử nghịch với chúng:

Let's get the start point of the polyline from the point list: Để lấy tọa độ điểm bắt đầu của polyline từ danh sách các điểm:

```
(setq startPt(car ptList))
```

Let's get the end point of the polyline from the point list: Để lấy tọa độ điểm kết thúc của polyline từ danh sách các điểm:

```
(setq endPt(car(reverse ptList)))
```

Let's get the area and perimeter of the polyline: Để có được diện tích và chu vi của polyline:

```
(command "area" "Object" en)
(setq plineArea(getvar "area"))
(setq plinePeri(getvar "perimeter"))
```

Let's get the Line Type of the polyline: Để có được loại đoạn thẳng của Polyline:

```
(if(cdr(assoc 6 enlist))
  (setq plineLtyp(cdr(assoc 6 enlist)))
  (setq plineLtyp "BYLAYER"))
)
```

Let's get the Color of the polyline: Để có được màu sắc của polyline:

```
(if(cdr(assoc 62 enlist))
  (setq plineClr(cdr(assoc 62 enlist)))
  (setq plineClr "BYLAYER"))
)
```

Solid (Các khối đặc)

Typical DXF Group Codes for an SOLID Entity: Các mã nhóm diễn hình của đối tượng khối đặc:

To list a solid entity in AutoCAD you type LIST<enter> and select the solid. To do this in AutoLisp you would: (Để liệt kê các mã nhóm DXF của một khối đặc trong AutoCad, bạn phải nhập LIST<Enter> rồi chọn đối tượng khối đặc đó. Trong Autolisp bạn phải thực hiện điều đó như sau:

```
(setq en(car (entsel "\n Select a Solid: ")))  
(setq enlist(entget en))
```

Would return: (Kết quả trả về:)

```
((-1 . <Entity name: 1cb06d0>) (0 . "SOLID") (5 . "92") (100 . "AcDbEntity")  
(67 . 0) (8 . "STR") (100 . "AcDbTrace") (10 1.0 1.0 0.0) (11 2.0 1.0 0.0)  
(12 3.0 3.0 0.0) (13 4.0 4.0 0.0) (39 . 0.0) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: Tách các mã thành từng phần và sắp xếp lại:

```
(  
  (-1 . <Entity name:2b80648>) -1 - Entity Name (AutoCAD Automatically takes care of this) (Tên  
                                đối tượng do AutoCad tự động thực hiện)  
  
  (0 . "SOLID")                0 - Entity Type (loại đối tượng)  
  
  (5 . "92")                   5 - Handle Name (If handles are turned on) (Tên điều khiển nếu  
                                chức năng điều khiển được kích hoạt)  
  
  (8 . "STR")                  8 - Layer Name (tên lớp)  
  
  (10 1.0 1.0 0.0)             10 - First Corner Point (Tọa độ đỉnh thứ nhất)  
  
  (11 2.0 1.0 0.0)             11 - Second Corner Point (Tọa độ đỉnh thứ hai)  
  
  (12 3.0 3.0 0.0)             12 - Third Corner Point (Tọa độ đỉnh thứ ba)  
  
  (13 4.0 4.0 0.0)             13 - Fourth Corner Point (Tọa độ đỉnh thứ tư)  
  
                                Note: If only three corners are used to define the solid, corner 4 will equal corner 3. (Nếu chỉ sử dụng  
                                ba đỉnh để xác định một khối đặc thì đỉnh thứ tư sẽ là đỉnh thứ ba)  
  
  (39 . 0.0)                   39 - Thickness. (optional) Default = 0.0 (Tùy chọn độ dày, giá
```


trị mặc định là 0.0)

(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng thuộc không gian model, bằng 1 là đối tượng thuộc không gian giấy vẽ)
(100 . "AcDbEntity")	100 - Don't worry about this
(100 . "AcDbTrace")	100 - Don't worry about this
(210 0.0 0.0 1.0)	210 - Don't worry about this Extrusion Factor
)	

Let's play with it: Thử với các mã này:

(cdr(assoc 10 enlist)) would return (trả về giá trị) (1.0 1.0 0.0)

Remember, CDR returns everything after the first item in a list. (Nhớ là hàm CDr trả về toàn bộ danh sách trừ phần mục đầu tiên của danh sách)

To get the first corner of the solid : Để có được tọa độ đỉnh thứ nhất của đối tượng:

```
(cdr (assoc 10 enlist))
```

To get the second corner of the solid : Để có được tọa độ đỉnh thứ hai của khối đặc:

```
(cdr(assoc 11 enlist))
```

To get the layer name of the solid : Để có được tên lớp của khối đặc:

```
(cdr(assoc 8 enlist))
```

To see if the entity is indeed a SOLID entity : Để kiểm tra đối tượng có thực là một khối đặc không:

```
(if (= "SOLID" (cdr(assoc 0 enlist)))
```

Text (Văn bản)

Typical DXF Group Codes for a Text Entity: Các mã nhóm diễn hình của một đối tượng văn bản:

To list a text entity in AutoCAD you type LIST<enter> and select the text. To do this in AutoLisp you would:
(Để liệt kê các mã DXf của một đối tượng văn bản trong AutoCad bạn phải nhập LIST<Enter> và chọn đối tượng văn bản đó. Trong Autolisp bạn phải thực hiện điều đó như sau:)

```
(setq en(car (entsel "\n Select a Text Entity: ")))  
  
(setq enlist(entget en))
```

Would return: (Kết quả trả về)

```
((-1 . <Entity name: 37b0658>) (0 . "TEXT") (5 . "8B") (100 . "AcDbEntity")  
(67 . 0) (8 . "STR") (100 . "AcDbText") (10 23.4375 9.1875 0.0) (40 . 0.125)  
(1 . "This is some Text!") (50 . 0.0) (41 . 1.0) (51 . 0.0) (7 . "STANDARD")  
(71 . 0) (72 . 0) (11 0.0 0.0 0.0) (210 0.0 0.0 1.0) (100 . "AcDbText") (73 . 0))
```

Let's take it apart and put it in order: Tách các mã này thành từng phần và sắp xếp lại:

```
(  
  
  (-1 . <Entity name: 37b0658>)    -1 - Entity Name (AutoCAD Automatically takes care of  
                                     this) (Tên đối tượng do AutoCad tự động thực hiện)  
  
  (0 . "TEXT")                      0 - Entity Type (Loại đối tượng)  
  
  (1 . "This is some Text!")        1 - Text Value (Giá trị văn bản)  
  
  (5 . "8B")                        5 - Handle Name (If handles are turned on) (Tên điều  
                                     khiển nếu chức năng điều khiển được kích hoạt)  
  
  (7 . "STANDARD")                  7 - Text Style Name (Tên kiểu văn bản)  
  
  (8 . "STR")                       8 - Layer Name (Tên lớp)  
  
  (10 23.4375 9.1875 0.0)           10 - Start Point for the text entity (Toạ độ điểm bắt đầu của  
                                     văn bản)  
  
  (11 0.0 0.0 0.0)                  Don't worry about this (Không quan tâm tới mã này)  
  
  (40 . 0.125)                      40 - Text Height (Chiều cao văn bản)  
  
  (41 . 1.0)      Don't worry about this width factor. (Không quan tâm tới mã độ rộng này)  
  
  (50 . 0.0)                        50 - Rotation Angle expressed in radians (Góc quay theo Radians)
```

(51 . 0.0) Don't worry about this oblique angle. (Không quan tâm tới mã góc nghiêng này)

(67 . 0) Don't worry about this (Không quan tâm tới mã này)

(71 . 0) 0=Normal 2=Backward 4=Upside Down (0 là Bình thường, 2 là từ phải qua trái, 4 là viết lộn ngược) ; Hì hì, cái này tớ cũng mu tít

(72 . 0) Don't worry about this. 72 and 73 work together for alignment. (Không quan tâm tới mã này. Mã 72 và mã 73 cùng được kết hợp để căn chỉnh văn bản)

(73 . 0) Don't worry about this. 73 and 72 work together for alignment. (Không quan tâm tới mã này. Mã 72 và mã 73 cùng được kết hợp để căn chỉnh văn bản)

(100 . "AcDbText") Don't worry about this SubClass Marker (Không quan tâm tới mã tạo phân lớp phụ này)

(100 . "AcDbEntity") Don't worry about this (Không quan tâm tới mã này)

(210 0.0 0.0 1.0) Don't worry about this Extrusion Factor. (Không quan tâm tới mã vuốt dài này)

)

Let's play with it: Thử với các mã này:

(cdr (assoc 10 enlist)) would return (trả về kết quả) (23.4375 9.1875 0.0)

Remember, CDR returns everything after the first item in a list. (Nhớ rằng hàm CDR trả về toàn bộ danh sách trừ khoản mục đầu tiên)

To get the height of the text : Để có được chiều cao của văn bản:

```
(cdr (assoc 40 enlist))
```

To get the start point of the text : Để có tọa độ điểm bắt đầu của văn bản:

```
(cdr (assoc 10 enlist))
```

To get the layer name of the text : Để có tên lớp của văn bản:

```
(cdr (assoc 8 enlist))
```

To get the text value of the text : Để có giá trị của văn bản:

```
(cdr (assoc 1 enlist))
```

To get the rotation angle of the text : Để có góc quay của văn bản:

```
(cdr (assoc 50 enlist))
```

To see if the entity is indeed a TEXT entity : Để kiểm tra đối tượng có thực là văn bản không:

```
(if (= "TEXT" (cdr (assoc 0 enlist)))
```

Trace (Các đường vết)

Typical DXF Group Codes for a TRACE Entity: Các mã nhóm DXF điển hình của đối tượng đường vết:

To list a trace entity in AutoCAD you type LIST<enter> and select the trace. To do this in AutoLisp you would: (Để liệt kê các mã nhóm của đối tượng đường vết trong AutoCad, bạn phải nhập LIST<Enter> và chọn đường vết đó. Trong Autolisp bạn sẽ thực hiện điều đó như sau:)

```
(setq en(car (entsel "\n Select a TRACE:")))  
(setq enlist(entget en))
```

Would return: (Kết quả trả về)

```
((-1 . <Entity name: 1cb06f8>) (0 . "TRACE") (5 . "97") (100 . "AcDbEntity") (67 . 0) (8 . "STR") (100 .  
"AcDbTrace") (10 2.0 2.5 0.0) (11 2.0 1.5 0.0) (12 4.0 2.5 0.0) (13 4.0 1.5 0.0) (39 . 0.0) (210 0.0 0.0 1.0))
```

Let's take it apart and put it in order: Tách các mã này thành từng phần và sắp xếp lại:

```
(  
  
(-1 . <Entity name:1cb06f8>) -1 - Entity Name (AutoCAD Automatically takes care of this) (Tên  
đối tượng do AutoCad tự động thực hiện)  
  
(0 . "TRACE") 0 - Entity Type (Loại đối tượng)  
  
(5 . "97") 5 - Handle Name (If handles are turned on) (Tên điều khiển  
nếu chức năng điều khiển được kích hoạt)  
  
(8 . "STR") 8 - Layer Name (Tên lớp)  
  
(10 2.0 2.5 0.0) 10 - First Corner Point (Tọa độ đỉnh thứ nhất)  
  
(11 2.0 1.5 0.0) 11 - Second Corner Point (Tọa độ đỉnh thứ hai)
```

(12 4.0 2.5 0.0)	12 - Third Corner Point (Tọa độ đỉnh thứ ba)
(13 4.0 1.5 0.0)	13 - Fourth Corner Point (Tọa độ đỉnh thứ tư)
(39 . 0.0)	39 - Thickness. (optional) Default = 0.0 (Độ dày tùy chọn, giá trị mặc định là 0.0)
(67 . 0)	67 - Absent or zero indicates the entity is in model space. If set to 1, paper space. (Không có hay bằng không chỉ thị rằng đối tượng thuộc không gian model, bằng 1 là đối tượng thuộc không gian giấy vẽ)
(100 . "AcDbEntity")	100 - Don't worry about this (Không cần lưu ý)
(100 . "AcDbTrace")	100 - Don't worry about this (Không cần lưu ý)
(210 0.0 0.0 1.0)	210 - Don't worry about this Extrusion Factor (Không cần lưu ý về yếu tố vuốt này)
)	

Let's play with it: Hãy thử với các mã này:

(cdr(assoc 10 enlist)) would return (giá trị trả về) (1.0 1.0 0.0)

Remember, CDR returns everything after the first item in a list.

To get the first corner of the trace : Để có được tọa độ đỉnh đầu tiên của đường vết:

```
(cdr (assoc 10 enlist))
```

To get the second corner of the trace : Để có được tọa độ đỉnh thứ hai của đường vết:

```
(cdr(assoc 11 enlist))
```

To get the layer name of the trace : Để có được tên lớp của đường vết:

```
(cdr(assoc 8 enlist))
```

To see if the entity is indeed a TRACE entity : Để kiểm tra đối tượng có thực là đường vết không:

```
(if (= "TRACE" (cdr(assoc 0 enlist)))
```

Note: Notice the resemblance to the SOLID entity. The only difference being, the TRACE requires FOUR corners to define itself.

Lưu ý: Đường vết có sự tương đồng với một đối tượng khối đặc. Điều duy nhất khác biệt là đường vết yêu cầu phải có bốn đỉnh để xác định nó

XLine

Typical DXF Group Codes for an XLine Entity: Các mã nhóm diễn hình của một đối tượng Xline:

To list an XLine entity in AutoCAD you type LIST<enter> and select the XLine. To do this in AutoLisp you would: (Để liệt kê các mã nhóm DXF của một đối tượng Xline trong AutoCad, bạn phải nhập LIST<Enter> và chọn đối tượng Xline đó. Trong Autolisp bạn sẽ thực hiện điều đó như sau:)

```
(setq en(car (entsel "\n Select an XLINE:")))  
(setq enlist(entget en))
```

Would return: (Kết quả trả về)

```
((-1 . <Entity name: 40222960>) (0 . "XLINE") (330 . <Entity name: 40073cf8>) (5 . "92C") (100 .  
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbXline") (10 5181.72 4894.2 0.0) (11 0.983723  
0.179691 0.0))
```

Let's take it apart and put it in order: Tách các mã thành từng phần và sắp xếp lại:

```
(  
  (-1 . <Entity name: 40222960>)    -1 - Entity Name (AutoCAD Automatically takes care of  
                                     this) (Tên đối tượng do AutoCad tự động thực hiện)  
  
  (0 . "XLINE")                     0 - Entity Type (Loại đối tượng)  
  
  (5 . "92C")                        5 - Handle Name (If handles are turned on) (Tên điều khiển)  
  
  (8 . "0")                          8 - Layer Name (Tên lớp)  
  
  (10 5181.72 4894.2 0.0)            10 - First Point (In WCS) (Toạ độ điểm đầu tiên trong hệ tọa độ WCS)  
  
  (11 0.983723 0.179691 0.0)        11 - Unit Direction Vector (In WCS) (Vector định hướng đơn vị  
                                     trong hệ tọa độ WCS)  
  
  (67 . 0)                          67 - Absent or zero indicates the entity is in model space. If set
```

to 1, paper space.

(100 . "AcDbEntity") 100 – Ignore

(100 . "AcDbXline") 100 - Don't worry about this Subclass marker

(330 . <Entity name: 40073cf8>) 330 - Ignore

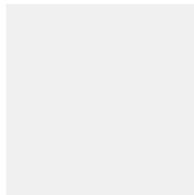
)

Note: There's not much you can do with an XLine so, let's let it go at that. Get on away from here Miss Daisy!

Chú ý: Không có gì nhiều để làm với thẳng Xline này, thôi kệ nó ở đó. Kể từ đây nghỉ chơi Ms.Daisy

End of Entity DXF Group Code Descriptions (Kết thúc phần mô tả mã nhóm DXF của đối tượng)

Support this site! Make a Donation.



[AutoLisp Advanced Tutorial](#)

[AutoLisp Tutorial Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated February 26th, 2005

Copyright 2002-2005 JefferyPSanders.com. All rights reserved.

The AutoLisp Tutorial – Extreme

Autolisp Siêu cấp

Well you made it. You've read the Beginners, Intermediate, and Expert's tutorials and now you want to apply your knowledge in real world situations. I believe the best way to approach this is to read through commented code and follow what is going on. It would be even better if you had the programmers thoughts before starting the program. So, we will do that. But what code? This I will leave up to you. If you have read all of the tutorials and still can't get the code to do what you want, click [<here>](#) to suggest a program for the Extreme section. I'll start it with a couple of problems I've encountered in the last year or so. Placing vertices along a polyline bulge and figuring the angle of an arc. I know, seems simple. But what happens when your start angle is around 350 degrees and your end angle is around 30 degrees. You can't subtract the end angle from the start angle can you?

Ồ, vậy là bạn đã làm được. Bạn đã đọc hết phần tự học Autolisp qua các phần từ sơ cấp tới cao cấp và bây giờ bạn muốn áp dụng các kiến thức của mình vào những tình huống thực tế. Tôi tin rằng cách tốt nhất để đạt được điều đó là đọc kỹ các đoạn mã Autolisp đã được chú giải và bám theo nó thực hiện. Thậm chí tốt hơn nếu bạn có những suy nghĩ của một nhà lập trình trước khi bắt đầu chương trình đó. Nào, ta hãy bắt đầu nhé. Nhưng đoạn mã Autolisp nào chứ? Tôi mặc kệ bạn đó. Nếu bạn đã đọc hết tất cả các phần tự học mà vẫn không kiếm được đoạn mã nào để làm cái điều bạn muốn thì hãy gửi ý định của bạn cho tôi (cụ Jeff) theo địa chỉ email dưới đây.

Tôi sẽ bắt đầu bằng hai vấn đề mà tôi đã đối đầu trong năm ngoái. Đặt các đỉnh dọc theo một polyline cong và dựng góc của một cung. Tôi biết nó dường như đơn giản. Nhưng điều gì sẽ xảy ra khi góc bắt đầu của cung khoảng 350 độ và góc kết thúc là khoảng 30 độ. Bạn không thể trừ góc bắt đầu cho góc kết thúc được, đúng không?

Placing vertices along a PolyLine Bulge	Figuring the Angle of an Arc		

PolyLine Bulge (Polyline cong)

Have you ever tried to recreate a polyline by drawing lines from vertex to vertex only to find out it doesn't work if you have a polyline bulge? (ARC) Aggravating isn't it. Well, I've found a method to insert vertex points along the polyline bulge area to recreate a bulge effect and get you closer to what you need. *I'm looking for this code....*

Bạn đã từng cố gắng để vẽ lại một polyline bằng cách vẽ các đoạn thẳng từ đỉnh này tới đỉnh khác chỉ để tìm ra rằng không thể được khi polyline của bạn cong? (ARC) Tức không cơ chứ.Ồ, tôi đã tìm ra phương pháp để đưa vào các tọa độ đỉnh dọc theo vùng polyline cong để tái tạo lại một hiệu quả cong và cho bạn tới gần hơn với điều bạn muốn. *Tôi đang tìm kiếm mã Autolisp này*

Arc Angle (Góc của cung tròn)

Have you ever tried to figure the angle of an arc only to be frustrated by the fact the angle cannot be simply subtracted? It's true because an arc's angle is always measured counter clock-wise so you could have a starting angle of 350 degrees and an ending angle of 30 degrees. Can't subtract the end angle from the start angle can you? When it crosses zero, there are problems. (Angle could also be set to clock-wise instead of counter clock-wise)

Bạn đã từng cố gắng để dựng góc của một cung tròn để chỉ được ... bởi sự thật là các góc không thể được trừ một cách đơn giản? Đó là sự thật vì góc của một cung tròn luôn được đo theo ngược chiều kim đồng hồ. Vì thế bạn có góc bắt đầu khoảng 350 độ và góc kết thúc là 30 độ. Không thể trừ bớt góc kết thúc khỏi góc bắt đầu, đúng không? Khi nó cắt ngang qua 0 độ, đó là vấn đề. (Góc cũng có thể được đặt theo chiều kim đồng hồ thay vì theo ngược chiều kim đồng hồ)

The first method I used was to begin at the starting angle and increment the angle by one until I reached the end angle or I crossed the zero angle. This works but it is a little goofy.

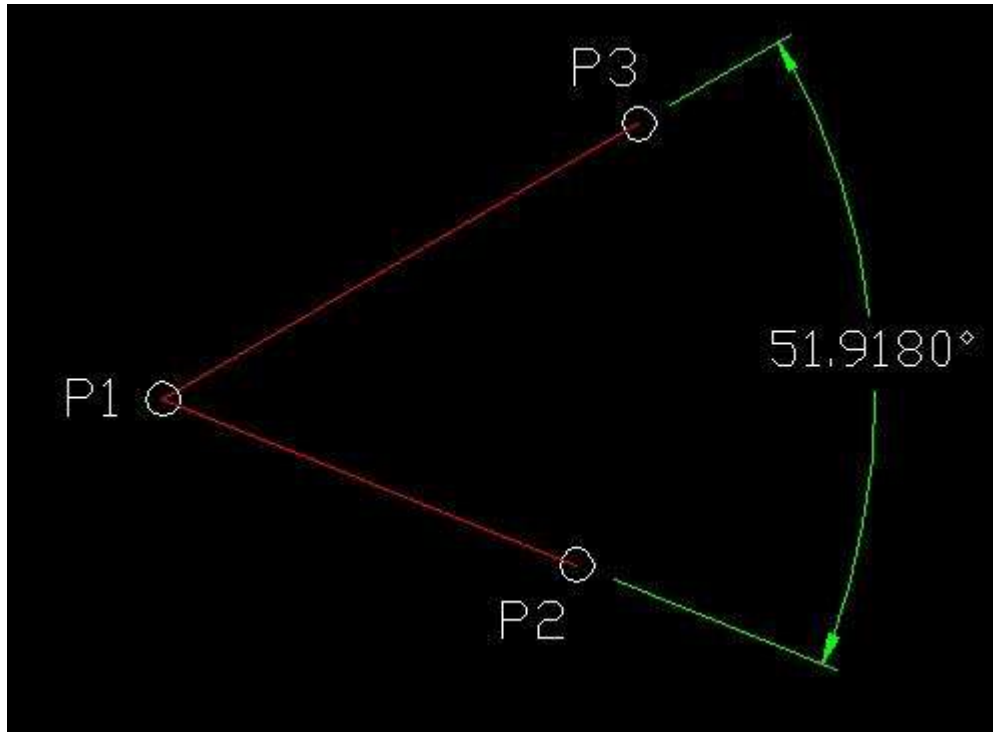
Phương pháp thứ nhất tôi sử dụng là bắt đầu từ góc bắt đầu và tăng dần góc cho tới khi đạt tới góc kết thúc hoặc cắt ngang góc 0 độ. Cách này được nhưng hơi cù lần.

The second method I've seen (sent to me by a visitor) was to reset the UCS to the center point with an angle that matched the starting angle. Then simply get the ending angle. Not bad. Not bad at all.

Phương pháp thứ hai tôi nhận được từ một khách thăm là đặt lại hệ trục tọa độ UCS về tâm với một góc trùng với góc bắt đầu. Sau đó đơn giản là lấy góc kết thúc. Không tồi chứ, không tồi chút nào.

The shortest method I've found was to use the built in geometry calculator:

Phương pháp ngắn nhất tôi đã tìm được là sử dụng hàm tính toán hình học nội trú.



The calculator function we will use is called ANG. The syntax is ANG(apex, endPt1, endPt2). In the illustration above the apex is P1, the first point is P2, and the third point is P3. You can also do this on the command line. Type CAL and press enter. Then type ang(cur,cur,cur) and press enter. Now pick the three points.

Hàm tính toán ta sẽ sử dụng là hàm ANG. Cú pháp là ANG(apex, endPt1, endPt2). Trong minh họa ở trên apex là P1, Điểm đầu tiên là P2, và Điểm thứ 3 là P3. Bạn cũng có thể làm điều đó trên dòng lệnh. Nhập CAL và nhấn Enter. Sau đó nhập Ang(Cur,Cur,Cur) và nhấn Enter. Bây giờ chọn ba điểm đó.

To do this in autolisp.....you first need to make sure the calculator is loaded:

Để thực hiện điều này trong Autolisp, đầu tiên bạn phải đảm bảo hàm tính toán được tải về:

```
(if (not c:cal) (arxload "geomcal"))
```

Then send your points to the calculator: (Sau đó đưa các điểm của bạn vào trong hàm tính toán)

```
(setq myAng (cal "ang(p1,p2,p3)"))
```

This should set variable myAng to 51.9180 (Điều này sẽ đặt biến myAngle bằng 51.9180)

Anyone else have a shorter answer? (Bất cứ ai có cách ngắn hơn, xin mời)<[click](#)>

[AutoLisp Tutorial Home](#)

[AutoLisp Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated February 26th, 2005

Copyright 2002-2007 JefferyPSanders.com. All rights reserved.

The AutoLisp Tutorial - DCL

Dialog Control Language

Ngôn ngữ điều khiển hộp thoại

Contents

- [Getting Started](#)
- [Rows and Columns](#)
- [Controls](#)
- [Image](#)
- [Action](#)
- [Set_Tile and Mode_Tile](#)
- [List and how to handle them.](#)
- [Saving data from the dialog box](#)
- [Part 1 - Buttons](#)
- [Part 2 - Edit_Box](#)
- [Part 3 - List_Box](#)
- [Part 4 - PopUp_List](#)
- [Part 5 - Radio_Buttons](#)
- [Part 6 - Text and Toggle](#)
- [Part 7 - Putting it all together](#)

Getting Started (Mở đầu)

I'm not going to explain everything there is to know about Dialog Control Language. There are plenty of books on the subject. I will attempt to give you a good understanding of how DCL interacts with AutoLisp. You should be able to write your first DCL driven Autolisp program in no time. Let's get to it.

Tôi không giải thích mọi điều hiểu biết về ngôn ngữ điều khiển hộp thoại. Có rất nhiều tài liệu về môn học này. Tôi sẽ cố gắng cung cấp cho bạn một kiến thức tốt về cách mà DCL tương tác với Autolisp. Bạn sẽ nhanh chóng có thể viết chương trình Autolisp điều khiển DCL đầu tiên của bạn . Nào ta hãy bắt đầu nhé.

Here is the basic order for things to happen inside the AutoLisp file: (Đây là trật tự cơ bản cho mọi thứ xảy ra trong tập tin Autolisp:)

<code>(defun C:MyProgram()</code>	<code>; Define the main program (Định nghĩa chương trình chính)</code>
<code> (defun myFunction1()</code>	<code>; Define the functions you need like saveVars (Định nghĩa chương trình mà bạn cần chẳng hạn như saveVar để lưu biến)</code>
<code> ;do stuff he</code>	<code>; Thực hiện một số nhiệm vụ nào đó</code>
<code>)</code>	<code>; Kết thúc myFunction1</code>
<code> (defun myFunction2()</code>	<code>; (Định nghĩa chương trình mà bạn cần)</code>
<code> ;do stuff here</code>	<code>; Thực hiện một số nhiệm vụ nào đó</code>
<code>)</code>	<code>; Kết thúc myFunction2</code>
<code> (setq list1(list "a" "b" "c" "d"))</code>	<code>; Build the list if any are required (Tạo một danh sách nếu được yêu cầu)</code>
<code> (setq list2(list "1" "2" "3" "4"))</code>	<code>; (Tạo một danh sách nếu được yêu cầu)</code>
<code> (setq list3(list "x" "y" "z"))</code>	<code>; (Tạo một danh sách nếu được yêu cầu)</code>
<code>load_dialog</code>	<code>;Load the DCL file (Tải tập tin DCL)</code>

[illegible]

```
        ;do stuff here if okay was pressed (Thực hiện một số nhiệm vụ nào đó nếu người
        sử dụng nhấn nút OK)

    )

; close the program (Đóng chương trình)
```

We will discuss this in detail later on. (Chúng ta sẽ thảo luận chi tiết vấn đề này ở phần sau)

I've found the hardest thing to get a grip on is laying out the dialog box. Please take a look at the rows and columns section if you need help with laying out a dialog box. For now let's look at the basics. *I'm going to throw a lot at you, but don't panic. I'll cover them a little closer later on.* We will cover these items first: button, row, column, boxed_row, and boxed_column.

Tôi phát hiện điều khó nhất để nắm vững là bố trí hộp thoại. Hãy xem phần các hàng và các cột khi bạn cần giúp đỡ việc bố trí một hộp thoại. Bây giờ ta sẽ xem xét phần cơ bản. *Tôi sắp quăng cho bạn một lô xích xông các vấn đề, nhưng đừng có run nhé. Tôi sẽ giải quyết chúng từng chút một ở phần sau.* Chúng ta hãy bắt đầu với các mục Nút, hàng, cột, hàng hộp và cột hộp

Okay and Cancel Buttons - The DCL code for a these buttons look like this: (Mã DCL cho các nút này là:)

```
: button {                                \\ Define the button (Định nghĩa nút)

    key = "accept";                        \\ Define the action key (This is my name assigned to this button)
                                           (Định nghĩa khóa hành động. Đây là tên của tôi gán cho nút này)

    label = " Okay ";                      \\ This is the text displayed on the button. (Đây là chữ hiển thị trên nút)

    is_default = true;                    \\ Allows the button to activate when the user presses the enter key.
                                           (Cho phép kích hoạt khi người sử dụng nhấn Enter)

}                                           \\ Close the button definition (Đóng định nghĩa nút)


: button {                                \\ Define another button (Định nghĩa một nút khác)

    key = "cancel";                        \\ Define the action key. (I chose this name for the button.) (Định nghĩa
                                           khóa hành động. Tôi chọn tên này cho nút mới)

    label = " Cancel ";                    \\ Text displayed on the button. (Chữ hiển thị trên nút)

    is_default = false;                    \\ Does not allow this key to activate when the enter key is pressed. (
                                           Không cho phép khóa này kích hoạt khi nhấn Enter)
```

```
is_cancel = true;      \\ Activate this key when the close button [ X ] is selected. (Kích hoạt  
                        khóa này khi nút Close [X] được chọn)  
  
}                      \\ Close the button definition (Đóng định nghĩa nút này)
```

Rows and Columns designate areas to lay your controls on. [Controls are list boxes, edit boxes, buttons, etc.]. (là các vùng được thiết kế để đặt các nút điều khiển của bạn.[Nút điều khiển là các hộp danh sách, hộp chỉnh sửa, các nút, vv...vv])

Here is the DCL code for a column: (Đây là mã DCL cho cột)

```
: column {  
  
}
```

Here is the DCL code for a row: (Đây là mã DCL cho hàng)

```
: row {  
  
}
```

Simple right? Thật đơn giản, phải không?

Here is the code for a column with two buttons: Đây là mã cho một cột với hai nút:

```
: column {  
  
  : button {  
    key = "accept";  
    label = " Okay ";  
    is_default = true;  
  }  
  
  : button {  
    key = "cancel";  
    label = " Cancel ";  
    is_default = false;  
    is_cancel = true;  
  }  
  
}
```



Notice the buttons are stacked on top of each other.

(Chú ý là các nút chồng lên nhau)

Here is the code for a row with two buttons: **Đây là mã cho một hàng với hai nút:**

```
: row {  
  
  : button {  
    key = "accept";  
    label = " Okay ";  
    is_default = true;  
  }  
  
  : button {  
    key = "cancel";  
    label = " Cancel ";  
    is_default = false;  
    is_cancel = true;  
  }  
  
}
```



Notice the buttons are side by side.

Chú ý là các nút nằm cạnh nhau

Let's turn the above into Boxed Rows and Boxed Columns. Here is the code for a boxed_column with two buttons: **Quay trở lại với các hàng hộp và cột hộp đã nói ở trên. Đây là mã cho một cột hộp với hai nút:**

```
: boxed_column {  
  
  : button {  
    key = "accept";  
    label = " Okay ";  
    is_default = true;  
  }  
  
  : button {  
    key = "cancel";  
    label = " Cancel ";  
    is_default = false;  
    is_cancel = true;  
  }  
  
}
```



Notice the box around the buttons.

Chú ý tới hộp xung quanh các nút

Here is the code for a boxed_row with two buttons: **Đây là mã cho một hàng hộp với hai nút:**


```

: boxed_row {

: button {
  key = "accept";
  label = " Okay ";
  is_default = true;
}

: button {
  key = "cancel";
  label = " Cancel ";
  is_default = false;
  is_cancel = true;
}

}

```



Notice the box around the buttons.

Chú ý tới hộp xung quanh các nút

Now we know the difference between a row and a boxed_row. We also know that controls inside a column get stacked on top of each other [vertical] and controls inside a row are next to each other [horizontal].

Giờ ta đã biết sự khác nhau giữa hàng và hàng hộp. Ta cũng biết rằng các nút điều khiển trong một cột thì xếp chồng lên nhau [thẳng đứng], còn trong một hàng thì xếp nằm cạnh nhau.[nằm ngang]

Important: You should never ever attempt to build a dialog box without a cancel button. Trust me. *Or you can do what I did and find out for yourself.*

Chú ý quan trọng: Đừng bao giờ cố gắng tạo một hộp thoại không có nút Cancel. Hãy tin tôi đi. *Hoặc là bạn có thể cứ làm điều tôi đã làm và tự tìm ra kết luận của bạn.*

Overview (Tổng quan)

Let's take a look at the entire code for the dialog box above, including the LSP file: (Ta hãy xem xét mã hoàn chỉnh cho hộp thoại trên bao gồm cả tập tin Autolisp)

DCL FILE NAMED: DCL_TEST.DCL

AUTOLISP PROGRAM NAMED: DCL_TEST.lsp

TÊN TẬP TIN DCL: DCL_TEST.DCL

TÊN CHƯƠNG TRÌNH AUTOLISP: DCL_TEST.lsp

```

DCL_TEST : dialog {

: boxed_row {

```

```

(defun C:DCL_TEST()
  (setq dcl_id (load_dialog "DCL_TEST.dcl"))
  (if (not (new_dialog "DCL_TEST" dcl_id) )
    (exit))

```

<pre> : button { key = "accept"; label = " Okay "; is_default = true; } : button { key = "cancel"; label = " Cancel "; is_default = false; is_cancel = true; } } </pre>	<pre> (action_tile "cancel" "(setq ddiag1) (done_dialog)") (action_tile "accept" "(setq ddiag2) (done_dialog)") (start_dialog) (unload_dialog dcl_id) (if (= ddiag 1) (princ "\n \n ...DCL_TEST Cancelled. \n ")) (if (= ddiag 2) (alert "You pressed the OKAY button!"))) </pre>
---	--

AutoLISP

Let's take a look at the AutoLisp part of this simple program. (Ta hãy xem xét phần Autolisp của chương trình đơn giản này)

First we defined our program: (Trước hết ta xác định chương trình của mình)

```
(defun C:DCL_TEST()
```

The second thing was to load the DCL file from disk. Insert a path if necessary, but I always liked to use the autocad search path to find the file. Notice the dcl_id variable used. We will need this later on. This identifies the file opened.

Thứ hai là tải tập tin DCL từ ổ đĩa. Đưa vào đường dẫn nếu cần, mà tôi luôn thích sử dụng đường tìm kiếm của AutoCad để tìm tập tin đó. Chú ý rằng ta sử dụng biến dcl_id mà ta sẽ cần sau này. Điều này cũng như tập tin được mở.

```
;;;--- Put up the dialog box (Đặt hộp thoại)
(setq dcl_id (load_dialog "DCL_TEST.dcl"))
```

Since a DCL file can contain multiple dialog box definitions, [I'll show you an example later on.] we need to specify which dialog box we want to load. This is taken care of in the next statement. Notice the name matches the definition inside the DCL file.

Do một tập tin DCL có thể chứa nhiều định nghĩa hộp thoại [tôi sẽ chỉ cho bạn một ví dụ ở phần sau], chúng ta cần mô tả hộp thoại nào mà ta muốn tải. Điều này được bảo đảm trong thông báo tiếp theo. Chú ý tới cái tên trùng với định nghĩa trong tập tin DCL

```
;;;--- See if it is already loaded (Kiểm tra xem nó đã được tải chưa)
```

```
(if (not (new_dialog "DCL_TEST" dcl_id) ) (exit))
```

Next, we define our action keys. These tell the program what to do whenever a user presses a button, selects an item in a list, etc. Notice the two action keys. "cancel" and "accept" . These match the keys in the DCL file. I made these names up. You can name them whatever you want as long as it is not a name used by AutoCAD and it only appears once in each DCL definition. You can't have two controls with the same key, the program would get confused. So would I!!!! Whenever the user presses the cancel key, the program sets variable ddiag to 1 and closes the dialog box. Upon pressing the Okay button, [key = accept], the program sets variable ddiag to 2 and closes the dialog box. That's it. Nothing else to add except, you could have just about anything in here, including instructions to execute a function [I'll demonstrate that later.]

Tiếp theo, ta xác định các khóa hành động của chúng ta. Điều này sẽ bảo chương trình điều phải làm bất cứ khi nào người sử dụng nhấn một nút, chọn một khoản mục trong một danh sách, vvv..... Chú ý tới hai khóa hành động “Cancel” và “Accept”. Chúng trùng với các khóa trong tập tin DCL. Tôi đã tạo các tên này. Bạn có thể đặt bất cứ tên gì bạn thích, dài thoải mái mà không phải các tên đã được AutoCad sử dụng và nó chỉ xuất hiện một lần trong mỗi định nghĩa của DCL. Bạn không thể có hai khả năng điều khiển cho cùng một khóa, chương trình sẽ bị rối. Tôi đã bị như vậy!!!!. Bất cứ khi nào người sử dụng nhấn khóa Cancel, chương trình sẽ đặt biến ddiag về 1 và đóng hộp thoại. Áp dụng việc nhấn nút Okay [khóa là Accept], chương trình sẽ đặt biến ddiag thành 2 và đóng hộp thoại. Vậy đó. Chẳng có gì khác được thêm vào trừ phi bạn phải có cái gì đó ở đây, bao gồm các chỉ dẫn để thực hiện một hàm [Tôi sẽ biểu diễn điều đó sau]

```
;;;--- If an action event occurs, do this function (Nếu một sự hành động xảy ra, thực hiện hàm sau)
```

```
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")  
(action_tile "accept" "(setq ddiag 2) (done_dialog)")
```

Finally, the big step. Everything is defined, let's display the dialog box. (Cuối cùng, một bước vĩ đại. Mọi thứ đã kết thúc, ta hãy hiển thị hộp thoại)

```
;;;--- Display the dialog box (Hiển thị hộp thoại)  
(start_dialog)
```

The program halts here until the dialog box is issued a "done_dialog" call. In the mean time the user is interacting with the action keys. When the user presses a button the program kills the dialog box with the unload_dialog call. Notice the dcl_id passed as a parameter. You could have more than one dialog file open at one time. So, the unload_dialog function needs to know which one to unload.

Chương trình sẽ treo ở đây cho tới khi hộp thoại có một thao tác gọi “Hộpthoại_hànhđộng”. Lúc đó người sử dụng phải có tương tác với các khóa hành động. Khi người sử dụng nhấn một nút, chương trình sẽ hủy hộp thoại với việc gọi lệnh hộp thoại thoát tải (unload_dialog). Chú ý rằng biến dcl_id thoát ra như một tham số. Bạn có thể có hơn một tập tin hộp thoại mở trong một lần. Vì thế hàm lệnh unload_dialog cần phải biết thoát khỏi hộp thoại nào.

```
;;;--- Display the dialog box (Hiển thị hộp thoại)  
(unload_dialog dcl_id)
```

Finally the dialog box is gone. What next? Find out which button the user pressed? Exactly! Remember setting the ddiag variable to either 1 or 2. We will use that now to determine which button was pressed. *There is a better way to do this, I'll show you later.*

Hộp thoại đã kết thúc xong. Điều gì kế tiếp? Hãy tìm xem nút nào đã được người sử dụng nhấn? Chính xác? Hãy nhớ tới việc đặt biến ddiag về giá trị 1 hoặc 2. Ta sẽ sử dụng chúng để xác định xem nút nào được nhấn. *Có một cách tốt hơn để làm điều này, tôi sẽ chỉ cho bạn sau.*

```
;;;--- If the cancel button was pressed - display message (Nếu nút Cancel bị nhấn, hiển thị)
(if (= ddiag 1)
  (princ "\n \n ...DCL_LSP Cancelled. \n ")
)
```

```
;;;--- If the "Create" button was pressed (Nếu nút Create bị nhấn)
(if (= ddiag 2)
  (alert "You pressed the OKAY button!")
)
```

```
And the last step, close the autolisp program. (và bước cuối cùng, đóng chương trình Autolisp)
)
```

The basic DCL model (Một mẫu tập tin DCL cơ bản)

I will now give you the model I use for 98% of all dialog based programs I write. I start with this basic program and build from there. It is basically the same thing we've gone over before. I put up a dialog box. I allow the user to press buttons and manipulate data. The only difference is, when the user presses the OKAY button I save the settings in the dialog file before it closes. I do this inside the action key call with a function called saveVars. Without further ado, here's the model:

Bây giờ tôi sẽ cung cấp cho bạn một mẫu mà tôi sử dụng cho 98% các chương trình chứa hộp thoại mà tôi viết. Tôi bắt đầu với chương trình cơ bản này và tạo dựng tiếp từ đó. Điều đó cũng như điều tương tự chúng ta đã từng làm trước đây. Tôi đặt một hộp thoại. cho phép người sử dụng nhấn các nút và thao tác với các dữ liệu. Điều khác duy nhất là khi người sử dụng nhấn nút Okay tôi sẽ lưu lại những điều đặt trong tập tin hộp thoại trước khi đóng nó. Tôi thực hiện điều này trong khóa hành động kèm theo hàm được gọi là hàm saveVar. Ngoài ra chẳng còn gì khác nữa, và đây là mẫu đó.

The DCL File: Tập tin DCL

```
EXAMPLE : dialog {
  label = "EXAMPLE.lsp";    \\ Puts a label on the dialog box (Đặt nhãn cho hộp thoại)
  initial_focus = "textval"; \\ Sets the initial focus (Đặt mục tiêu ban đầu)
  : column {
    : row {
      : boxed_column {
```

```

: edit_box {      \\ The edit_box control - Something new! (Cái này mới nè:
                    Nút điều khiển hộp thoại hiệu chỉnh)

    key = "textval";
    label = "Text Value to Find";
    edit_width = 30;
    value = "";
}
}
}
: row {
: boxed_row {
: button {
    key = "accept";
    label = " Okay ";
    is_default = true;
}
: button {
    key = "cancel";
    label = " Cancel ";
    is_default = false;
    is_cancel = true;
}
}
}
}
}
}

```

The AutoLisp File: (Tập tin Autolisp)

```

;;;--- EXAMPLE.lsp - Text Find (Tìm dòng văn bản)

(defun saveVars()
  ;;;;--- Save the input from the dialog box (Lưu lại đầu vào của hộp thoại)
  (setq textVal (get_tile "textval"))
)

(defun C:EXAMPLE()

  ;;;;--- Load the dcl file (Tải tập tin DCL)
  (setq dcl_id (load_dialog "EXAMPLE.dcl"))

  ;;;;--- Load the dialog definition if it is not already loaded (Tải định nghĩa
  hộp thoại nếu nó chưa được tải)
  (if (not (new_dialog "EXAMPLE" dcl_id) ) (exit))

```

```

;;;--- If an action event occurs, do this function (Nếu một sự hành động xảy ra,
thực hiện hàm này)
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;;--- Display the dialog box (Hiển thị hộp thoại)
(start_dialog)

;;;--- If the cancel button was pressed - display message (Hiển thị nếu nút
Cancel được nhấn)
(if (= ddiag 1)
  (princ "\n \n ...EXAMPLE Cancelled. \n ")
)

;;;--- If the "Okay" button was pressed (Nếu nút Okay được nhấn)
(if (= ddiag 2)
  (princ "\n \n ...Example Complete!")
)

;;;--- Suppress the last echo for a clean exit (Triệt tiêu việc lặp lại thao tác cuối
và thoát êm)
(princ)

)

```

Screen Shot: Hiển thị trên màn hình:



If you could not follow some of the autolisp functions, you can ignore them for now or read the [Autolisp Tutorial](#). (Nếu bạn không hiểu chỗ nào trong chương trình Autolisp trên, hãy đọc lại phần tự học Autolisp)

This is the model. Very little will change from this setup. You may add some controls to the DCL file which in turn will add Action calls and influence the number of lines inside the saveVars routine. The only other thing that will change is inside the OKAY button function. [When ddiag = 2]. You will have to tell the program what to do when all of the information is gathered. We will cover these later.

Đây là một mẫu . Có rất ít thay đổi so với ban đầu. Bạn có thể thêm vài nút điều khiển vào tập tin DCL mà nó sẽ thêm vào các việc gọi lệnh hành động và ảnh hưởng đến số dòng bên trong của việc lưu biến saveVar.

Điều khác duy nhất là có sự thay đổi bên trong chức năng của nút Okay.[Khi ddiag = 2]. Bạn phải bảo chương trình điều nó cần làm khi tắt cả các thông tin được nhóm lại. Ta sẽ học những điều này sau.

[Back](#)

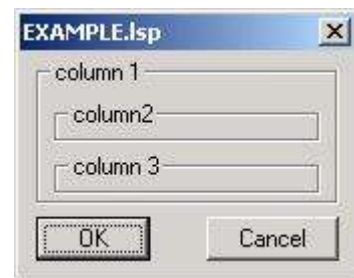
Rows and Columns (Các hàng và các cột)

Let's get a feel for laying out a dialog box using nested rows and columns. I'll do this by showing you the DCL code next to a picture of the dialog box. I can't think of any easier way to do it. *Ignore all of the Labels and TEXT controls. I will use these to visually show the location of things.*

Ta hãy hình dung sự bố trí một hộp thoại sử dụng các tổ hợp hàng và cột. Tôi sẽ thực hiện điều này bằng cách trình bày cho bạn mã DCL bên cạnh hình thể hiện của hộp thoại. Tôi không thể nghĩ ra cách nào dễ dàng hơn cách này. Hãy quên đi tất cả các nhãn và các text điều khiển. Tôi sử dụng chúng chỉ để thể hiện cái chỗ đặt chúng mà thôi.

Example 1:

```
EXAMPLE : dialog {
  label = "EXAMPLE.lsp";
  : column {
    : boxed_column {
      label = "column 1";
      : boxed_column {
        label = "column2";
      }
      : boxed_column {
        label = "column 3";
      }
    }
  }
  ok_cancel;
}
```



Column (Một cột chính bao gồm)
Column (Một cột hộp 1 chứa)
Column<\p> (Các cột hộp bên trong 2,3)

Tổ hợp hàng gồm hai nút OK và Cancel

```

EXAMPLE : dialog {
    label = "EXAMPLE.lsp";
    : column {
        : boxed_row {
            label = "row 1";
            : boxed_column {
                label = "column2";
            }
            : boxed_column {
                label = "column 3";
            }
        }
        ok_cancel;
    }
}

```



Notice I changed the first boxed column into a boxed row. (Chú ý sự thay đổi cột hộp thứ nhất thành hàng hộp)

Cột chính bao gồm:

Row (Hàng hộp 1 chứa)

Column (Cột hộp 2 bên trong)

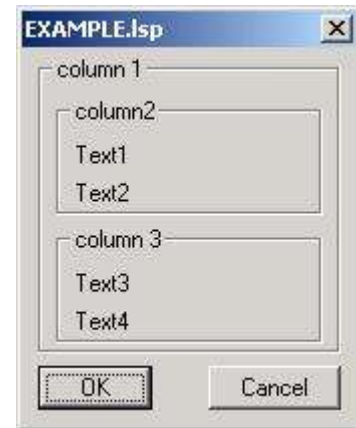
Column (Cột hộp 3 bên trong)

Hàng chứa hai nút OK và Cancel


```

EXAMPLE : dialog {
    label = "EXAMPLE.lsp";
    : column {
        : boxed_column {
            label = "column 1";
            : boxed_column {
                label = "column2";
: text {
            key = "t1";
            value = "Text1";
        }
        : text {
            key = "t2";
            value = "Text2";
        }
    }
    : boxed_column {
        label = "column 3";
        : text {
            key = "t3";
            value = "Text3";
        }
        : text {
            key = "t4";
            value = "Text4";
        }
    }
}
    ok_cancel;
}

```



Column (Cột chính bao gồm)

Column (Cột hộp 1 chứa)

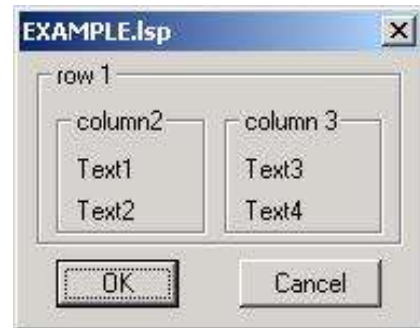
Column<\p> (Các cột hộp bên trong 2,3 chứa các khoá t1, t2 và các khóa t3,t4)

Hàng hai nút OK và Cancel

```

EXAMPLE : dialog {
  label = "EXAMPLE.lsp";
  : column {
    : boxed_row {
      label = "row 1";
      : boxed_column {
        label = "column2";
      : text {
        key = "t1";
        value = "Text1";
      }
      : text {
        key = "t2";
        value = "Text2";
      }
    }
    : boxed_column {
      label = "column 3";
      : text {
        key = "t3";
        value = "Text3";
      }
      : text {
        key = "t4";
        value = "Text4";
      }
    }
  }
  ok_cancel;
}

```



Cột chính bao gồm

Row (Hàng hộp 1 chứa hai cột hộp 2,3)
 Column (Cột hộp 2 chứa hai khoá t1, t2)
 Column (Cột hộp 3 chứa hai khoá t3,t4)

Hàng chứa hai nút OK và Cancel

```

EXAMPLE : dialog {
  label = "EXAMPLE.lsp";
  : column {
    : boxed_row {
      label = "row 1";
      : boxed_row {
        label = "row 2";
        : text {
          key = "t1";
          value = "Text1";
        }
        : text {
          key = "t2";
          value = "Text2";
        }
      }
    }
    : boxed_row {
      label = "row 3";
      : text {
        key = "t3";
        value = "Text3";
      }
      : text {
        key = "t4";
        value = "Text4";
      }
    }
  }
  ok_cancel;
}

```



Cột chính bao gồm:

Row (Hàng hộp 1 chứa các hàng hộp 2,3)

Row (Hàng hộp 2 chứa các khóa t1,t2)

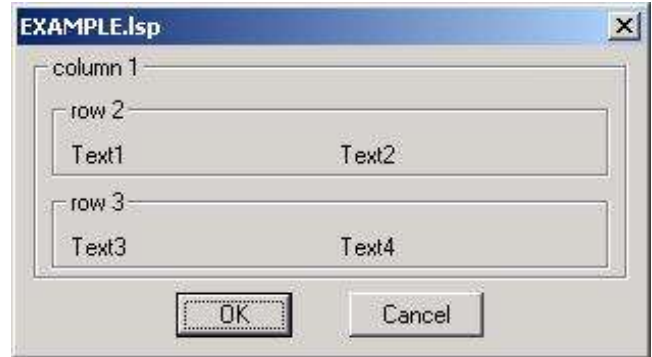
Row (Hàng hộp 3 chứa các khóa t3,t4)

Hàng chứa hai nút OK và Cancel

```

EXAMPLE : dialog {
  label = "EXAMPLE.lsp";
  : column {
    : boxed_column {
      label = "row 1";
      : boxed_row {
        label = "row 2";
        : text {
          key = "t1";
          value = "Text1";
        }
        : text {
          key = "t2";
          value = "Text2";
        }
      }
    }
    : boxed_row {
      label = "row 3";
      : text {
        key = "t3";
        value = "Text3";
      }
      : text {
        key = "t4";
        value = "Text4";
      }
    }
  }
  ok_cancel;
}

```



Cột chính bao gồm:

Column (Cột hộp 1 chứa các hàng hộp 2,3)

Row (Hàng hộp 2 chứa các khóa t1, t2)

Row (Hàng hộp 3 chứa các khóa t3, t4)

Hàng chứa hai nút OK và Cancel

I hope this puts an end to the rows and columns issue. If not, you know where to find me. (Tôi hy vọng đến đây có thể kết thúc cái món hàng và cột này. Nếu không chắc bạn biết tìm tôi ở đâu rồi.)

[Back](#)

Dialog Control Language–Controls (Các nút điều khiển trong DCL)

Let's take a look at the different controls you can put on the dialog box. You've seen a few of them in the

overview portion but let's go over the majority of them now. I will only show the most used properties for each control.

Ta hãy đi kiểm các nút điều khiển khác nhau mà bạn có thể đặt vào hộp thoại. Bạn đã từng ngó thấy chúng trong phần tổng quan nhưng bây giờ ta sẽ tập trung ngâm cứu chúng. Tôi sẽ chỉ trình bày những thuộc tính sử dụng chính của mỗi nút .

LAYOUT CONTROLS:

```
: column {  
    label = "My Column";  
}
```

Column (Cột)

You can omit the label. Bạn có thể bỏ qua việc gắn nhãn

```
: boxed_column {  
    label = "My Column";  
}
```

Boxed Column (Cột hộp)

You can omit the label. Bạn có thể bỏ qua việc gắn nhãn

```
: row {  
    label = "My Row";  
}
```

Row (Hàng)

You can omit the label. Bạn có thể bỏ qua việc gắn nhãn

```
: boxed_row {  
    label = "My Row";  
}
```

Boxed Row (Hàng hộp)

You can omit the label. Bạn có thể bỏ qua việc gắn nhãn

```
: spacer {  
    width = 10;  
    height = 15;  
}
```

A normal spacer to help align other controls. You can omit the width and height properties. (Một khoảng không gian bình thường để giúp cho việc căn chỉnh vị trí của nút điều khiển khác. Bạn có thể bỏ qua thuộc tính độ rộng và chiều cao)

```
: spacer_0;
```

A spacer that usually has no width. It simply means, if you have to stretch something in this row to make things fit, stretch this spacer instead of the other items.. (Một không gian có độ rộng luôn bằng 0. Nó đơn giản có nghĩa là nếu bạn phải duỗi dài nút nào đó trên hàng này để tạo một sự hài hoà thì hãy dẫn không gian này thay cho các không gian khác)

```
: spacer_1;
```

The smallest spacer that's visually noticable. (Không gian nhỏ nhất có thể thấy được)

BUTTON (NÚT)

```
: button {  
    key = "button1;  
    label = "Okay";  
    is_tab_stop = true;  
    is_cancel = false;  
    is_default = true;  
    width = 15;  
    height = 10;  
    mnemonic = "B";  
}
```

key = The name you assign to the button. (Tên bạn gán cho nút)

label = The text displayed on the button. Dòng chữ thể hiện trên nút)

is_cancel = Determines if this is the cancel button. One control must be assigned to be the cancel action. (Xác định đây có phải là nút xóa không. Một quá trình điều khiển được gán cho hành động xóa này)

is_default = Determines if this button activates when the user presses the enter key. (Xác định nút này có bị kích hoạt khi người sử dụng nhấn khóa Enter hay không)



mnemonic = Determines if you can press ALT+B to move to this action key. You assign the letter and DCL will underscore the letter on the dialog box to let the user know it is ALT-able. *If that is a word!* (Xác định bạn có thể sử dụng tổ hợp phím Alt +B để chuyển tới khoá hành động này hay không. Bạn gán chữ cái và DCL sẽ gạch dưới chữ cái đó trong hộp thoại để báo cho người sử dụng biết nó có khả năng kết hợp với phím Alt. Nếu đó là một từ)

BOXED RADIO COLUMN, RADIO COLUMN, & RADIO BUTTON

CỘT LỰA CHỌN HỘP, CỘT LỰA CHỌN, VÀ NÚT LỰA CHỌN

Cái tên này hơi tự bịa ra vì có biết nội dung nó là gì, thôi thì cứ cho nó một cái tên Tí Tèo như vậy để mà đọc tiếp. Ai có cái tên hay hơn, đúng hơn thì chỉ giùm tớ, xin cảm ơn trước nhé.

```

// Use boxed_radio_column if a box is
// required. (Sử dụng cột lựa chọn hộp nếu một
// hộp được yêu cầu)
: radio_column {
    label = "Choices"; // Label for the column or boxed_column
    key = "choices";   // (Gắn nhãn cho cột hay cột hộp)
                        // Action key for the radio column (Khóa
                        // hành động cho cột lựa chọn)

    : radio_button { // First radio button (Nút lựa chọn thứ nhất)
        label = "Choice 1";
        key = "choice1";
    }

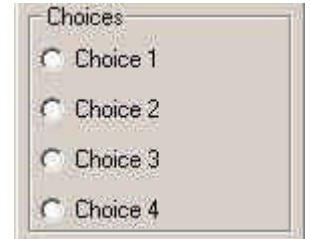
    : radio_button { // Second radio button (Nút lựa chọn thứ hai)
        label = "Choice 2";
        key = "choice2";
    }

    : radio_button { // Third radio button (Nút lựa chọn thứ ba)
        label = "Choice 3";
        key = "choice3";
    }

    : radio_button { // Fourth radio button (Nút lựa chọn thứ tư)
        label = "Choice 4";
        key = "choice4";
    }
}

// Close the radio_column (Đóng cột lựa
// chọn)

```



BOXED RADIO ROW, RADIO ROW, & RADIO BUTTON

HÀNG LỰA CHỌN HỘP, HÀNG LỰA CHỌN, VÀ NÚT LỰA CHỌN

```

// Use boxed_radio_row for box. (Sử dụng
// hàng lựa chọn hộp cho một hộp)
: radio_row {
    label = "Choices"; // Label for the row or boxed_row (Gắn
    key = "choices";   // nhãn cho hàng hay hàng hộp)
                        // Action key for the radio row (Khóa hành
                        // động cho hàng lựa chọn)

    : radio_button { // First radio button (Nút lựa chọn thứ
        label = "1"; // nhất)
        key = "choice1";
    }
}

```



```

: radio_button {           // Second radio button (Nút lựa chọn thứ
    label = "2";           hai)
    key = "choice2";
}
: radio_button {           // Third radio button (Nút lựa chọn thứ ba)
    label = "3";
    key = "choice3";
}
: radio_button {           // Fourth radio button (Nút lựa chọn thứ tư)
    label = "4";
    key = "choice4";
}
}                           // Close the radio_row (Đóng hàng lựa
                           chọn)

```

EDIT BOX (HỘP HIỆU CHỈNH)

```

: edit_box {               // Action key (Khóa hành động)
    key = "myval";         // Label for the edit box (Nhãn gắn
    label = "Value: ";     cho hộp hiệu chỉnh)
    edit_width = 10;       // Character width (Độ rộng ký tự)
    value = "";            // Initial value (Giá trị ban đầu)
}

```

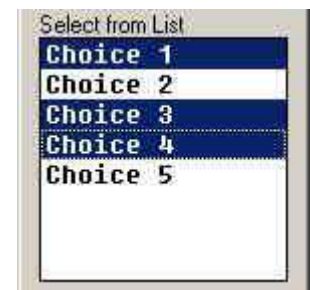


LIST BOX (HỘP DANH SÁCH)

```

: list_box {               // Label for the list box (Nhãn cho hộp
    label = "Choose Items"; danh sách)
    key = "mylist";        // Action key (Khóa hành động)
    height = 15;           // Height of list box (Độ cao hộp)
    width = 25;            // Width of list box (Độ rộng hộp)
    multiple_select = true; // Multiple Select = TRUE (nhiều lựa
    fixed_width_font = true; chọn)
    value = "0";           // Use fixed with font = TRUE (Sử
                           dụng font chữ đậm cố định)
}                           // Initial selection = 1st item (Lựa
                           chọn ban đầu là khoản mục đầu tiên)

```



Fixed Width Font = TRUE

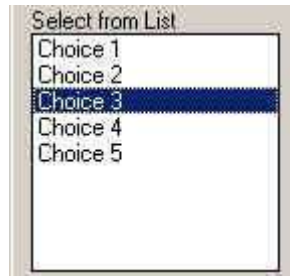
Multiple Select = TRUE

View of list box with... (Hình hiển thị
hộp danh sách với:)

View of list box with... (Hình hiển thị
hộp danh sách với:)



Fixed Width Font = FALSE. (Font chữ
mảnh)



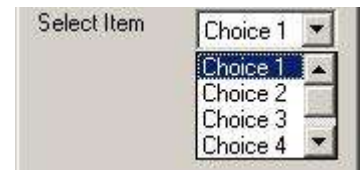
Fixed Width Font = FALSE. (Font
chữ mảnh)

Multiple Select = TRUE. (Cho chọn
nhiều khoản mục)

Multiple Select = FALSE. (Không
cho chọn nhiều khoản mục)

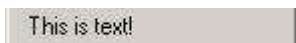
POPUP LIST (DANH SÁCH THẢ XUỐNG)

```
: popup_list {
  key = "mylist";           // Action key (Khóa hành động)
  label = "Select Item";    // Label (nhãn gắn)
  fixed_width_font =       // Use fixed with font = false (Không
false;                      sử dụng font chữ đậm)
  width = 15;              // Width of list box (Độ rộng hộp
  value = 0;               danh sách)
}                          // Initial selection = 1st item (Lựa
                          chọn ban đầu là khoản mục thứ nhất)
```



TEXT (VĂN BẢN)

```
: text {
  key = "mytext";           // Action key (Khóa hành động)
  value = "This is text!";  // Value (Giá trị văn bản)
}
```



Okay. That's it for controls. There are others but, these are the most used controls. Let's move on. (OK. Đó là các nút điều khiển. Còn có các nút khác nhưng đây là những nút sử dụng chủ yếu. Ta sang mục khác nhé.

[Back](#)

Dialog Control Language – Image (Hình ảnh trong DCL)

Image (Các hình ảnh)

Image (Hình ảnh)

An Image is used to display a vector graphic picture inside a rectangle. To create the image you can use three different methods. (Một hình ảnh được sử dụng để hiển thị một bức hình hoạ đồ vector trong một khung chữ nhật. Để tạo một hình ảnh bạn có thể sử dụng 3 phương pháp khác nhau)

- Vector_image - Draws a single straight line in the current image. (Ảnh Vector - Vẽ một đoạn thẳng trong ảnh hiện tại)
- Fill_image - Draws a filled rectangle in the current image. (Ảnh đặc- Vẽ một khung chữ nhật đặc trong ảnh hiện tại)
- Slide_image - Draws an AutoCAD slide in the image. (Ảnh Silse- Vẽ một ảnh AutoCad Slide trong ảnh hiện tại)

For this tutorial we will concentrate on the Slide_Image function. Everyone knows how to create a slide right? The MSLIDE command. Okay. We will assume our image is ready.

Để học điều này ta sẽ tập trung vào hàm ảnh Slide. Mọi người đã biết cách tạo một ảnh Slide rồi, đúng không? Lệnh Mslide nữa. Rồi Ok. Ta giả sử rằng ta đã có ảnh sẵn sàng rồi.

Let's look at the DCL Image definition and the AutoLisp functions required to display the image. (Hãy xem định nghĩa hình ảnh trong DCL và các hàm Autolisp cần có để hiển thị hình ảnh này.)

DCL - You must supply either the width and height or one of these plus an aspect_ratio. (Bạn phải áp dụng hoặc là độ cao và độ rộng hoặc là một trong chúng và một tỉ lệ ảnh)

```
: image {  
  key = "sld";  
  height = 30;  
  width = 30;  
  color = 0;  
  is_enabled = false;  
  is_tab_stop = false;  
}
```

AutoLisp

```
;;;--- First we need a slide name (Trước hết bạn cần đặt tên ảnh silde)
```

```

(setq mySlideName "c:/acad/myslide.sld")

;;;--- Second we need the key to the image control (Thứ hai bạn cần đặt khoá điều khiển hình ảnh)
(setq myKey "sld")

;;;--- Next we send the slide name and the key to the update function (Tiếp theo ta gửi tên
và khoá này vào hàm update hình ảnh)
(upDateImage mySlideName myKey)

; NOTE: Notice mySlideName becomes sldName and myKey becomes key when passed as
; parameters to the upDateImage function. (Chú ý: Biến mySlideName trở thành sldName và biến
myKey trở thành key khi đi qua hàm update hình ảnh như các tham số)

;;;--- Function to update the slide (Hàm update hình ảnh Slide)

(defun upDateImage(sldName key)

  ;;;;--- Get the width of the slide (Lấy độ rộng của hình ảnh)
  (setq width (dimx_tile key))

  ;;;;--- Get the height of the slide (Lấy độ cao của hình ảnh)
  (setq height (dimy_tile key))

  ;;;;--- Start the slide definition (Bắt đầu định nghĩa ảnh slide)
  (start_image key)

  ;;;;--- Wipe out the background (Xóa sạch nền)
  (fill_image 0 0 width height 0)

  ;;;;--- Put the slide in the image area (Đặt ảnh Slide vào vùng ảnh)
  (slide_image 0 0 width height sldName)

  ;;;;--- Finish the slide definition (Kết thúc định nghĩa ảnh slide)
  (end_image)

)

```

This function can be used over and over to display any slide image in an image control. Use the upDateImage function after the new_dialog call and before the action_tile statements. You can also use this function while the dialog box is displayed in one of your action calls. We will use this later on in the tutorial so you can see it in action.

Hàm này có thể sử dụng lần lượt để hiển thị bất kỳ ảnh slide nào trong một khóa điều khiển. Việc sử dụng hàm update hình ảnh sau khi gọi hộp thoại mới và trước các thông báo của phần tử hành động. Bạn cũng có thể sử dụng hàm này trong khi hộp thoại hiển thị trong một lần gọi hành động của bạn. Ta sẽ sử dụng điều này trong phần sau và bạn có thể quan sát nó làm việc.

[Back](#)

Dialog Control Language – Action (Hành động trong DCL)

Action (Các hành động)

In order for the dialog box to respond to user interaction, you must set up an action event. A trigger, to fire when the user selects an item from a list or presses a button. This works off of the KEY defined for each control inside the DCL file. If the key for a button is named "mybutton" then you must have an action named "mybutton". Otherwise the user will press the button and nothing will happen. Let's take a look at the action_tile.

Để hộp thoại tương tác với người sử dụng, bạn phải tạo ra một sự kiện hành động. Một cái lẫy để khai hỏa khi người sử dụng chọn một khoản mục trong danh sách hoặc nhấn một nút. Điều này sẽ ngắt một khoá được xác định cho mỗi thao tác điều khiển trong tập tin DCL. Nếu khóa cho một nút được đặt tên là “mybutton” thì bạn phải có một hành động được đặt tên là “mybutton”. Nếu không khi người sử dụng nhấn nút đó sẽ chẳng có gì xảy ra. Ta hãy xem xét phần tử hành động này.

```
(action_tile "key" "action")
```

The action_tile has two parameters. "Key" and "action". (Phần tử hành động có hai tham số là key và action)

"Key" - The name of the key you defined with the control inside the DCL file. (Tên của khoá mà bạn đã xác định với khoá điều khiển trong tập tin DCL)

"Action" - The action you want taken when the action event is fired. You can set a variable or run a function. (Hành động mà bạn muốn thực hiện khi sự kiện hành động được khai hỏa. Bạn có thể tạo một biến hoặc thực thi một hành động)

Examples: Các ví dụ:

Let's take for example, you have a button named "test" and you want to set the variable named myTEST to 1 if the user presses the button: (Lấy ví dụ, bạn có một nút đặt tên là “test” và bạn muốn đặt một biến tên là “myTEST” về 1 khi người sử dụng nhấn nút này:)

```
(action_tile "test" "(setq myTest 1)")
```

Notice the key is in quotes and the setq function is in quotes. This is standard procedure for the action_tile statement. Next let's say you have the same thing except you want to run a function named "saveVars" when the user presses the test button. (Chú ý là tên khoá và hàm setq phải được đặt trong ngoặc kép. Đó là thủ tục tiêu chuẩn cho các hàm thông báo phần tử hành động. Tiếp theo bạn có một hàm thông báo phần tử hành động tương tự như trên trừ việc bạn muốn chạy hàm saveVar khi người sử dụng nhấn nút test)

```
(action_tile "test" "(saveVars)")
```

Notice is is the same as the above. The (saveVars) is inside quotes. (Chú ý là cũng như ở trên hàm saveVar phải đặt trong ngoặc kép)

What if you wanted to do both, run the function and set a variable? (Vậ khi bạn muốn thực hiện cả hai điều trên chạy một chương trình và đặt một biến thì sao?)

```
(action_tile "test" "(setq myTest 1) (saveVars)")
```

Simply put both inside the quotes. (Đơn giản là đặt cả hai hàm setq và saveVar vào trong ngoặc kép.)

One more thing....What if the button is set to be the cancel or accept button for the dialog box? No problem...

Một điều nữa, nếu nút này được đặt là nút xóa hay chấp nhận đối với hộp thoại thì sao? Không có sao cả...

```
(action_tile "test" "(setq myTest 1) (saveVars) (done_dialog)")
```

Add the done_dialog statement to the end. [Inside the quotes](Thêm thông báo done_dialog vào cuối trước khi đóng ngoặc kép.)

That is about all there is to an action_tile. Remember, anything with a key can have an action call.

Đó là tất cả về một phần tử hành động. Nhớ rằng, bất kỳ khóa nào cũng có thể có một lệnh hành động.

[Back](#)

Dialog Control Language - Set and Mode Tile (Đặt và tạo chức năng cho phần tử trong DCL)

Set_Tile and Mode_Tile (Đặt phần tử và tạo chức năng phần tử)

Set_Tile - is used to set the value of a control. (Sử dụng để đặt giá trị cho một nút điều khiển)

`(set_tile "key" "value")`

Set_Tile has two parameters. "Key" and "Value". (Hàm Set_tile có hai tham số là "key" và "value")

"Key" - The name of the key you defined with the control inside the DCL file. (là tên của khóa mà bạn xác định với nút điều khiển trong tập tin DCL)

"Value" - The new value for the control. (Giá trị mới của nút điều khiển)

- Edit_box (Hộp hiệu chỉnh)
 - `(set_tile "mybox" "Jeff")` Displays Jeff in the edit box. (Hiển thị Jeff trong hộp hiệu chỉnh)
 - `(set_tile "mybox" "4'-3 1/2")` Displays 4'-3 1/2 in the edit box. (Hiển thị 4'-3 1/2 trong hộp hiệu chỉnh)
- List_box (Hộp danh sách)
 - `(set_tile "mylist" "0")` Selects the first item in the list. (Chọn khoản mục thứ nhất trong danh sách)
 - `(set_tile "mylist" "5")` Selects the sixth item in the list. (Chọn khoản mục thứ sáu trong danh sách)
 - `(set_tile "mylist" "")` No Items are selected. (Không chọn khoản mục nào)
- PopUp_List (Danh sách thả xuống)
 - `(set_tile "mylist" "0")` Selects the first item in the list. (Chọn khoản mục thứ nhất trong danh sách)
 - `(set_tile "mylist" "5")` Selects the sixth item in the list. (Chọn khoản mục thứ sáu trong danh sách)
 - `(set_tile "mylist" "")` No Items are selected. (Không chọn khoản mục nào)
- Toggle (Bật hoặc tắt kiểm soát hộp điều khiển)
 - `(set_tile "mytog" "0")` Removes the check from the box. (Xóa kiểm soát)
 - `(set_tile "mytog" "1")` Checks the box. (Kiểm soát)
- Radio_Button (Nút lựa chọn)
 - `(set_tile "myradio1" "1")` Moves the selection to the first radio button. (Chuyển việc lựa chọn về nút lựa chọn 1)
 - `(set_tile "myradio2" "1")` Moves the selection to the 2nd radio button. (Chuyển việc lựa chọn về nút lựa chọn 2)

Use Set_Tile after the new_dialog and before the action_tiles. Set_Tile can also be used as a function while the dialog box is displayed. (Sử dụng hàm Set_Tile sau thông báo new_dialog và trước thông báo action_tile.

Hàm Set_tile cũng có thể sử dụng như một hàm trong lúc hộp thoại được hiển thị)

Mode_Tile - is used to enable or disable a control. (Hàm được sử dụng để bật hoặc tắt một nút điều khiển)

```
(mode_tile "key" value)
```

Mode_Tile has two parameters. "Key" and Value. (Hàm có hai tham số là "Key" và Value)

"Key" - The name of the key you defined with the control inside the DCL file. (là tên của khóa mà bạn đã xác định với nút điều khiển trong tập tin DCL)

Value- The new value for the control. 0 = Enabled 1 = Disabled (Giá trị mới của nút điều khiển 0 là bật 1 là tắt)

```
(mode_tile "mylist" 0) Enables the list box (Kích hoạt hộp danh sách)
```

```
(mode_tile "mylist" 1) Disables the list box (Tắt hộp danh sách)
```

Use Mode_Tile after the new_dialog call and before the action_tiles. Mode_Tile can also be used as a function while the dialog box is displayed. (Hàm Mode_tile sử dụng sau hàm lệnh new_dialog và trước hàm action_tile. Hàm Mode_tile cũng có thể được sử dụng trong lúc hộp thoại được hiển thị)

That is all there is to set_tile and mode_tile. We will use them later on in the tutorial so you can see them in action. (Đó là tất cả mọi điều về hàm Set_tile và hàm Mode_tile. Ta sẽ sử dụng chúng trong phần sau và bạn sẽ thấy chúng hoạt động ra sao.)

[Back](#)

Dialog Control Language – List (Danh sách trong DCL)

List and how to handle them. (Danh sách và cách điều khiển chúng)

List_box and popup_list handle a list of items in basically the same manner. You have to load your list into the list box and at some point you have to decide which item or items were selected. In the "[Saving data from a dialog box](#)" I cover how to find the selected items from a list_box and a popup_list. How do we get the list inside the list_box? That is what we need to cover in this section. So let's get started.

Hộp danh sách và danh sách thả xuống về cơ bản điều khiển một danh sách theo cùng một phương cách. Bạn đã tải danh sách của bạn thành một hộp danh sách tại một điểm nào đó và bạn phải quyết định khoản mục nào hay những khoản mục nào sẽ được lựa chọn. Trong phần “Lưu giữ liệu từ một hộp thoại” tôi sẽ trình bày cách để tìm những khoản mục được lựa chọn từ một hộp danh sách hay một danh sách thả xuống. Làm sao để lấy được danh sách đó bên trong một hộp danh sách? Đó là điều bạn sẽ học trong phần này. Bắt đầu thôi.

If you looked at the previous sections you know the AutoLisp and DCL basic model. Let's get a copy of that in this section so we can look at it. I will replace the edit_box with a list_box control and add the code to handle the list. The list handling code is shown in the "[Saving data from a dialog box](#)" section of this tutorial.

Ở các phần trước, bạn đã biết về các mẫu cơ bản về AutoLisp và DCL. Ở phần này ta sẽ copy chúng lại để xem xét. Ta sẽ thay thế hộp hiệu chỉnh bằng một hộp danh sách và thêm vào các mã điều khiển danh sách đó. Mã điều khiển danh sách này được trình bày trong phần “Lưu dữ liệu từ một hộp thoại” của tài liệu này

I will show the **revised and new code in red** below. All of the **black code is unchanged** from the Basic Model and the instructions from the "list_box multiple selection" area of the "[Saving data from a Dialog box](#)" page.

Tôi sẽ trình bày các phần mã chỉnh sửa và mã mới bằng màu đỏ. Tất cả các mã màu đen là không thay đổi so với mẫu cơ bản và so với các chỉ dẫn ở mục “hộp danh sách đa lựa chọn” của phần “Lưu dữ liệu từ một hộp thoại”

The Basic Revised Model (Mẫu cơ bản được chỉnh sửa)

The DCL File: (Tập tin DCL)

```
EXAMPLE : dialog {
  label = "EXAMPLE.lsp";           \\ Puts a label on the dialog box (Gắn nhãn hộp thoại)
  : column {
    : row {
      : boxed_column {
        : list_box {               \\ I replaced the edit_box with a list_box. (Thay
thế hộp hiệu chỉnh bằng hộp danh sách)
          label = "Choose Items";
          key = "mylist";
          height = 15;
          width = 25;
          multiple_select = true;  \\ Multiple selection is on (Bật đa lựa chọn)
          fixed_width_font = true; \\ Fixed Width Font is on (Bật font đậm)
```



```

        value = "0";
    }
}
: row {
    : boxed_row {
        : button {
            key = "accept";
            label = " Okay ";
            is_default = true;
        }
        : button {
            key = "cancel";
            label = " Cancel ";
            is_default = false;
            is_cancel = true;
        }
    }
}
}
}
}

```

The AutoLisp File: Tập tin Autolisp:

```
;;;--- EXAMPLE.lsp
```

```

;;;--- I replaced the saveVars routine with the one from the "Save data from a list" section of this tutorial.
;;;      I also changed the things marked in red. (Tôi thay thế vòng lặp lưu biến saveVar bằng một trong các
;;;      cách được trình bày trong phần ” Lưu dữ liệu từ một hộp thoại” của tài liệu này. Tôi cũng thay đổi
;;;      những điều được đánh dấu màu đỏ.)

```

```
(defun saveVars(/ readlist count item)
```

```

;;;--- Notice the "mylist" is the action key associated with the DCL file and
;;;      the myList is the variable for the list built below.(Chú ý rằng “mylist” là khoá hành
;;;      động tương ứng với tập tin DCL và myList là biến cho danh sách được tạo thành dưới đây)

```

```

;;;--- Setup a list to hold the selected items (Tạo một danh sách để giữ các khoản mục chọn)
(setq retList(list))

```

```

;;;--- Save the list setting (Lưu việc đặt danh sách)
(setq readlist(get_tile "mylist"))

```

```

;;;--- Setup a variable to run through the list (Đặt biến để chạy qua toàn bộ danh sách)
(setq count 1)

```

```
;;;--- cycle through the list getting all of the selected items (Lặp qua toàn bộ danh sách
```

để lấy các khoản mục được lựa chọn)

```
(while (setq item (read readlist))
  (setq retlist(append retList (list (nth item myList))))
  (while
    (and
      (/= " " (substr readlist count 1))
      (/= "" (substr readlist count 1))
    )
    (setq count (1+ count))
  )
  (setq readlist (substr readlist count))
)
```

```
(defun C:EXAMPLE()
```

```
;;;--- I need to build a list of data (Tôi cần tạo một danh sách các dữ liệu)
(setq myList(list "1" "2" "3" "4" "5" "6" "7" "8"))

;;;--- Load the dcl file (Tải tập tin DCL)
(setq dcl_id (load_dialog "EXAMPLE.dcl"))

;;;--- Load the dialog definition if it is not already loaded (Tải định nghĩa hộp thoại nếu
nó chưa được tải)
(if (not (new_dialog "EXAMPLE" dcl_id) ) (exit))

;;;-- Here, I add the data to the list_box control (Ở đây tôi thêm dữ liệu vào hộp danh sách)
;;; I do this after the new_dialog call and before the action_tiles. (Điều này thực
;;; hiện sau hàm new_dialog và trước hàm action_tile.)
(start_list "mylist" 3)
(mapcar 'add_list myList)
(end_list)

;;;--- Notice the "mylist" is the action key associated with the DCL file and
;;; the myList is the variable for the list built above. (Chú ý rằng “mylist” là khoá
;;; hành động tương ứng với tập tin DCL và myList là biến cho danh sách được tạo thành dưới đây)

;; If an action event occurs, do this function (Nếu một thao tác xảy ra, thực hiện hàm này)
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;; If an action event occurs, do this function (Nếu một thao tác xảy ra, thực hiện hàm này)
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;;--- Display the dialog box (Hiển thị hộp thoại)
(start_dialog)

;;;- If the cancel button was pressed - display message (Hiển thị nếu nút Cancel được nhấn)
```

```

(if (= ddiag 1)
  (princ "\n \n ...EXAMPLE Cancelled. \n ")
)

;;;--- If the "Okay" button was pressed (Nếu nút “Okay” được nhấn)
(if (= ddiag 2)

  ;;;;--- Here, I decide what to do with my data (Ở đây tôi xác định điều cần thực hiện với biến)
  ;;;; I'm going to print each selected item on the command line. (In từng khoản mục
được chọn ra dòng lệnh)
  (foreach a retList
    (princ "\n") (princ a)
  )
)

;;;--- Suppress the last echo for a clean exit (Triệt tiêu sự lặp lại lệnh cuối và thoát êm)
(princ)
)

```

So, in order to display a list in a dialog box, you must first build the list. Then use the start_list, add_list, and end_list functions just after the new_dialog call and before the action_tiles. This is the same for both a list_box and a popup_list. That wasn't so bad. Was it?

Vậy đó, để hiển thị một danh sách trong một hộp thoại, trước hết bạn phải tạo danh sách đó. Sau đó sử dụng các hàm start_list, add_list, và end_list ngay sau hàm lệnh new_dialog và trước hàm action_tile. Điều này là như nhau đối với hộp danh sách và danh sách thả xuống. Không tồi chứ, phải không?

[Back](#)

Dialog Control Language–SaveVars(Lưu biến trong DCL-SaveVars)

In this section we are going to take a look at the SaveVars routine and dissect it for each type of control. The types of controls that we sometimes need to save are :

Trong phần này, ta sẽ xem xét vòng lưu biến SaveVars và dissect nó với từng loại nút điều khiển. Các loại nút điều khiển mà đôi khi chúng ta cần lưu lại là:

Edit_box - List_box - Popup_List - Radio_Buttons - Radio_Column - Radio_Row - Toggle

The SaveVars routine is executed just before the dialog box is issued a done_dialog call. You cannot get the data from the dialog box once it has been shut down. If you remember our action calls in the AutoLisp Program, they looked something like this:

Vòng lưu biến SaveVars được thực hiện ngay trước khi hộp thoại thực hiện lệnh gọi done_dialog. Bạn không thể lấy được dữ liệu từ hộp thoại khi nó đã đóng lại. Trừ phi bạn nhớ gọi lệnh hành động trong Autolisp. Chương trình sẽ giống như sau:

```
;;;-- If an action event occurs, do this function (Nếu một thao tác xảy ra, thực hiện hàm này)
(action_tile "accept" "(saveVars) (done_dialog)")
```

Notice the (saveVars) is just before the (done_dialog). This is the proper order. The settings from the dialog box will be saved just before it is shut down. (Chú ý rằng hàm (saveVars) được đặt ngay trước hàm (done_dialog). Đó là trật tự đúng. Điều này sẽ khiến hộp thoại phải lưu lại biến trước khi đóng lại.)

The SaveVars routine always has the same definition: (Vòng lưu biến saveVars luôn có cùng một định nghĩa là:)

```
(defun saveVars()

  ;stuff here (Thực hiện ba cái lằng nhằng ở đây)

)
```

That's the shell. We need to work on the "stuffing". Let's get to it. (Đó mới chỉ là cái vỏ. Ta cần phải làm việc với “ba cái lằng nhằng” nữa. Hãy tiếp tục đi nữa nào.)

Stuffing (Các hành động thực hiện trong vòng lưu biến SaveVars)

There are three parts to a control with an action call back statement. (Một nút điều khiển với một hành động gọi lại hàm thông báo có ba phần gồm:)

1. The DCL definition (Định nghĩa DCL)
2. The action call in the AutoLisp file. (Lệnh gọi hành động trong tập tin Autolisp)
3. The code inside the SaveVars routine. (Mã bên trong vòng lưu biến SaveVars)

In each of the following controls I will show you all three parts so we know how to put it all together later. Although, all of the parts are important, try to focus on the SaveVars routine.

Trong mỗi nút điều khiển dưới đây, tôi sẽ trình bày cả ba phần để bạn có thể biết cách đặt chúng kết hợp với nhau. Mặc dầu cả ba phần đều quan trọng nhưng hãy cố gắng tập trung vào vòng lưu biến SaveVars

[Edit_box](#) - [List_Box Single Selection](#) - [List_Box Multiple Selection](#) - [Popup List](#)
[Radio Column](#) - [Radio Row](#) - [Toggle](#) - [Ok Cancel](#)

Edit_Box (Hộp hiệu chỉnh)

1	<pre>: edit_box { key = "edbox"; label = "Bolt Circle Diameter:"; edit_width = 8; value = "10"; }</pre>	
2	<p>No Action call required. We will get the data the user typed when we use the SaveVars routine. But, if your deadset about doing something when the user types in the edit box, here you go:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy dữ liệu mà người sử dụng nhập vào khi ta sử dụng vòng lưu biến Savevars. Nhưng nếu bạn muốn cố định thực hiện một điều gì đó khi người sử dụng nhập vào hộp hiệu chỉnh, bạn sẽ thực hiện:</p> <pre>(action_tile "edbox" "(doThisFunction)")</pre>	
3	<p>To get the data as a real number: (Để có dữ liệu là một số thực:)</p> <pre>(defun saveVars() (setq edBox(distof(get_tile "edbox"))))</pre>	<p>To get the data as a string: (Để có dữ liệu là một chuỗi:)</p> <pre>(defun saveVars() (setq edBox(get_tile "edbox")))</pre>

NOTE: Notice I used the key name as the variable to store the data in. You will find this method makes it easier to keep things straight. DCL is case sensitive so, I never capitalize key names, only autolisp variable names if they are made up of more than one word. Like: thisIsMyVariable.

CHÚ Ý: Tôi sử dụng tên khoá như một biến để lưu dữ liệu vào. Bạn sẽ tìm thấy phương pháp này là dễ hơn để giữ trực tiếp mọi thứ của khoá. DCL là rất nhạy với kiểu chữ in hoa hay in thường, tôi không bao giờ sử dụng chữ in hoa cho các tên khoá, mà chỉ sử dụng trong tên biến của Autolisp khi tên đó dài hơn một từ. Ví dụ: thisIsMyVariable.

List_Box Single Choice Only (Hộp danh sách với sự lựa chọn đơn duy nhất)

1	<pre> : list_box { key = "mylist"; label = "Available Choices"; multiple_select = "FALSE"; // Sets single selection (Đặt sự lựa chọn đơn) width = 40; height = 20; fixed_width_font = true; // Equally spaced characters (Các ký tự cách đều nhau) value = ""; // Start with no item selected (Không chọn trước) } </pre>
2	<p>No Action call required. We will get the selected item when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại nếu người sử dụng thực hiện sự lựa chọn, vậy thì:</p> <p>(action_tile "edbox" "(doThisFunction)") ;;; Chỗ này chắc cụ Jeff muốn kiểm tra mấy thằng học mót đây, phải thay “edbox” bằng “mylist” mới phải chứ???</p>
3	<pre> (defun saveVars () ;;;--Get the selected item from the list (Lấy các khoản mục được chọn từ danh sách) (setq sStr(get_tile "mylist")) ;;;--- If the index of the selected item is not "" then something was selected (if (/= sStr "") (progn ;;;--- Something is selected, so convert from string to integer (Có đối tượng được chọn, chuyển đổi đối tượng từ chuỗi thành số nguyên) (setq sIndex(atoi sStr)) ;;;--- And get the selected item from the list (Và lấy đối tượng được chọn từ danh sách) (setq sName(nth sIndex myList))) ;;;--- Else, nothing is selected (ngược lại, không có đối tượng nào được chọn) (progn ;;;--- Set the index number to -1 (Đặt số thứ tự về -1) (setq sIndex -1) ;;;--- And set the name of the selected item to nil (Và đặt tên đối tượng được chọn về nil) (setq sName nil)))) </pre>

)
	<p>Notes: Chú ý:</p> <ol style="list-style-type: none"> 1. This should only be used when single selection is required. This will not work on multiple selection. (Điều này chỉ được sử dụng với việc lựa chọn đơn. Nó không làm việc nếu là đa lựa chọn) 2. This is assuming there is a list of items called myList. (Giả sử đã có danh sách tên là myList)

List_Box Multiple Choice (Hộp danh sách đa lựa chọn)

1	<pre>: list_box { key = "mylist"; label = "Available Choices"; multiple_select = "TRUE"; // Sets multiple selection (Đặt đa lựa chọn) width = 40; height = 20; fixed_width_font = false; // Use default font [no alignment] (Sử dụng font // mặc định, Không căn hàng) value = "4"; // Start with item 5 selected. (Bắt đầu với khoản mục thứ 5 được chọn) }</pre>
2	<p>No Action call required. We will get the selected items when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so: Không yêu cầu lệnh gọi hành động. Ta sẽ lấy các khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại khi người sử dụng tạo một lựa chọn, Thế thì:</p> <pre>(action_tile "mylist" "(doThisFunction)")</pre>
3	<pre>(defun saveVars(/ readlist count item) ;;--- Setup a list to hold the selected items (Đặt một danh sách để giữ các khoản // mục được chọn) (setq retList(list)) ;;--- Save the list setting (Lưu lại danh sách chọn) (setq readlist(get_tile "files")) ;;-- Setup a variable to run through the list (Đặt biến để lặp qua toàn bộ danh sách) (setq count 1) ;;--- cycle through the list getting all of the selected items (Lặp qua toàn bộ // danh sách các khoản mục được chọn) (while (setq item (read readlist))</pre>

	<pre> (setq retlist(append retList (list (nth item fileNames)))) (while (and (/= " " (substr readlist count 1)) (/= "" (substr readlist count 1))) (setq count (1+ count))) (setq readlist (substr readlist count)))) </pre> <p>Note: This method can be used for a single or multiple selection list_box. (Chú ý rằng phương pháp này chỉ sử dụng cho hộp danh sách đa lựa chọn)</p>
--	--

PopUp_List (Danh sách thả xuống)

1	<pre> : popup_list { key = "mylist"; // action key (Khóa hành động) fixed_width_font = false; // fixed width font off (Tắt font rộng đều) width = 20; // width of popup list (Độ rộng danh sách thả) height = 20; // height of popup list (Độ cao danh sách thả) } </pre>
2	<p>No Action call required. We will get the selected item when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy các khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại khi người sử dụng tạo một lựa chọn, Thế thì:</p> <pre> (action_tile "mylist" "(doThisFunction)") </pre>
3	<pre> (defun saveVars() ;;--- Get the selected item from the list (Lấy khoản mục lựa chọn từ danh sách) (setq sStr(get_tile "mylist")) (if(= sStr "") (alert "No Item was Selected!"))) </pre>

Radio_Column and Radio_Buttons (Cột lựa chọn và các nút lựa chọn)

1	<pre> : radio_column { // Use boxed_radio_column if a box is required. (Sử dụng cột hộp lựa chọn nếu yêu cầu một hộp) label = "Choices"; // Label for the column or boxed_column (Gắn nhãn cho cột hay </pre>
---	---

	<p>cột hộp)</p> <pre> key = "choices"; // Action key for the radio column (Khoá hành động của cột lựa chọn) : radio_button { // First radio button (Nút lựa chọn thứ nhất) label = "Choice 1"; key = "choice1"; } : radio_button { // Second radio button (Nút lựa chọn thứ hai) label = "Choice 2"; key = "choice2"; } : radio_button { // Third radio button (Nút lựa chọn thứ ba) label = "Choice 3"; key = "choice3"; } : radio_button { // Fourth radio button (Nút lựa chọn thứ tư) label = "Choice 4"; key = "choice4"; } } // Close the radio_column (Đóng cột lựa chọn) </pre>
2	<p>No Action call required. We will get the selected item when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy các khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại khi người sử dụng tạo một lựa chọn, Thế thì:</p> <pre> (action_tile "choices" "(doThisFunction)") // If any choice is made (Nếu một khoản mục bất kỳ được lựa chọn) (action_tile "choice1" "(doThatFunction)") // If choice1 is made (Nếu khoản mục thứ nhất được chọn) (action_tile "choice2" "(doDisFunction)") // If choice2 is made (Nếu khoản mục thứ hai được chọn) etc. </pre>
3	<pre> (defun saveVars() ;;;--- Get the key of the choice made (Lấy khóa của sự lựa chọn đã thực hiện) </pre>

```

;;; [ returns (Trả về giá trị) "choice1" "choice2" "choice3" or "choice4" ]

(setq choicesVal (get_tile "choices"))

)

OR (Hoặc)

(defun saveVars ()

;;;--- Get the value of each item (Lấy giá trị của mỗi một khoản mục)
(setq choice1 (atoi (get_tile "choice1"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn)
(setq choice2 (atoi (get_tile "choice2"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn)
(setq choice3 (atoi (get_tile "choice3"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn)
(setq choice4 (atoi (get_tile "choice4"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn)

)

```

Radio_Row And Radio_Buttons (Hàng lựa chọn và các nút lựa chọn)

```

1 : radio_row { // Use boxed_radio_row if a box is required. (Sử dụng hàng
    label = "Choices"; // Label for the row or boxed_row (Gắn nhãn cho hàng hay
    key = "choices"; // Action key for the radio row (Khoá hành động cho hàng lựa chọn)

    : radio_button { // First radio button (Nút lựa chọn thứ nhất)
        label = "Choice 1";
        key = "choice1";
    }

    : radio_button { // Second radio button (Nút lựa chọn thứ hai)
        label = "Choice 2";
        key = "choice2";
    }

    : radio_button { // Third radio button (Nút lựa chọn thứ ba)
        label = "Choice 3";
        key = "choice3";
    }

    : radio_button { // Fourth radio button (Nút lựa chọn thứ tư)
        label = "Choice 4";
        key = "choice4";
    }

```

	<pre> } } // Close the radio_row (Đóng hàng lựa chọn) </pre>
2	<p>No Action call required. We will get the selected item when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy các khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại khi người sử dụng tạo một lựa chọn, Thế thì:</p> <pre> (action_tile "choices" "(doThisFunction)") // If any choice is made (Nếu bất kỳ khoản mục nào được chọn) (action_tile "choice1" "(doThatFunction)") // If choice1 is made (Nếu choice1 được chọn) (action_tile "choice2" "(doDisFunction)") // If choice2 is made (Nếu choice2 được chọn) etc. </pre>
3	<pre> (defun saveVars() ;;--- Get the key of the choice made (Lấy khoá của lựa chọn được thực hiện) ;; [returns (Trả về giá trị) "choice1" "choice2" "choice3" or "choice4"] (setq choicesVal (get_tile "choices"))) OR (Hoặc) (defun saveVars() ;;--- Get the value of each item (Lấy giá trị của mỗi khoản mục) (setq choice1 (atoi (get_tile "choice1"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn) (setq choice2 (atoi (get_tile "choice2"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn) (setq choice3 (atoi (get_tile "choice3"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn) (setq choice4 (atoi (get_tile "choice4"))) // 0 = not chosen 1 = chosen (0 là không chọn, 1 là chọn)) </pre>

Toggle (Bật và tắt kiểm soát nút điều khiển)

1	<pre> : toggle { key = "tog1"; // Action key (Khoá hành động) </pre>
---	--

	<pre>label = "Name"; // Label (Nhãn) }</pre>
2	<p>No Action call required. We will get the value of the toggle when we use the SaveVars routine. But, sometimes you need to change things in the dialog box if the user makes a selection, so:</p> <p>Không yêu cầu lệnh gọi hành động. Ta sẽ lấy các khoản mục được chọn khi sử dụng vòng lưu biến SaveVars. Nhưng đôi khi bạn cần thay đổi những điều trong hộp thoại khi người sử dụng tạo một lựa chọn, Thế thì:</p> <pre>(action_tile "tog1" "(doThisFunction)")</pre>
3	<pre>(defun saveVars() ;;--- Get the selected item from the radio column (Lấy khoản mục lựa chọn từ cột lựa chọn) (setq tog1Val(atoi(get_tile "tog1"))) // 0 = unchecked 1 = checked (0 là không chọn, 1 là chọn))</pre>

Ok_Cancel

1	<pre>ok_cancel;</pre> <p>Note: I usually define my own okay and cancel buttons using two standard buttons, but this works fine. (Chú ý là tôi thường xác định hai nút Okay và Cancel của mình bằng cách sử dụng hai nút tiêu chuẩn mà chúng làm việc rất tốt)</p>
2	<p>You will need two action calls for this button...both close the dialog box but the accept or "Okay" key will save the dialog box settings before shutting the dialog box down.</p> <p>Bạn cần có hai lệnh gọi hành động cho nút này, cả hai đều đóng hộp thoại nhưng khóa Accept hay Okay sẽ lưu việc đặt hộp thoại lại trước khi đóng hộp thoại.</p> <pre>(action_tile "cancel" "(setq ddiag 1) (done_dialog)") (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")</pre> <p>Ahh....finally the SaveVars routine shows up. (A ha, vậy là xong với vòng lưu biến SaveVars rồi)</p>
3	<p>Nothing to do in the saveVars routine for these buttons. (Chả còn gì làm thêm với các nút này trong vòng lưu biến SaveVars)</p>

[Back](#)

Dialog Control Language - Part 1 (Phần thực hành 1 về DCL)

Part 1 - Buttons (Phần 1- Các nút)

Let's build a working DCL file showing us exactly how to handle buttons. (Ta sẽ xây dựng tập tin DCL hoạt động để thấy được chính xác cách điều khiển các nút)

We will build a DCL file containing three buttons plus a Close [Cancel] button. Each of the three buttons will display a message when pressed. (Ta sẽ tạo một tập tin DCL gồm ba nút và thêm một nút Close [Cancel]. Mỗi một trong ba nút sẽ hiển thị một thông báo khi bị nhấn.)

Layout thoughts: I will place the buttons in a column, (stacked on top of each other). Then I'll put the Close button at the bottom on a row of it's own. So...I'll need something like this:

Thiết kế hộp thoại: Ta sẽ đặt các nút trong một cột (Các nút chồng lên nhau). Sau đó ta sẽ đặt nút Close ở hàng cuối cùng của hộp thoại. Do đó, ta sẽ cần một thứ đại loại như thế này:

```
: column {
  : boxed_column {
    : button {
      // Put code for button 1 here (Đặt mã cho nút 1 ở đây)
    }
    : button {  ]
      // Put code for button 2 here (Đặt mã cho nút 2 ở đây)
    }
    : button {  ]
      // Put code for button 3 here (Đặt mã cho nút 3 ở đây)
    }
  }
  : boxed_row {
    : button {
      // Put code for the Close button here (Đặt mã cho nút Close ở đây)
    }
  }
}
```

Let's copy in the code for the header and all of the controls above. I'll show them in red. Notice the key names and labels had to be changed. (Ta sẽ copy các mã phần tiêu đề và tất cả các mã điều khiển ở trên. Tôi

sẽ trình bày chúng màu đỏ. Chú ý các tên khoá và các nhãn có sự thay đổi.)

```
SAMPLE1 : dialog {
  label = "Sample Dialog Box Routine - Part 1";
  : column {
    : boxed_column {
      : button {
        key = "but1";
        label = "Button 1";
        is_default = false;
      }
      : button {
        key = "but2";
        label = "Button 2";
        is_default = false;
      }
      : button {
        key = "but3";
        label = "Button 3";
        is_default = false;
      }
    }
  }
  : boxed_row {
    : button {
      key = "cancel";
      label = "Close";
      is_default = true;
      is_cancel = true;
    }
  }
}
```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE1.DCL *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Click chuột phải và copy tập tin trên. Mở NotePad và dán nó vào. Lưu lại nó dưới tên SAMPLE1.DCL Phải đảm bảo là bạn đã thay đổi giá trị của hộp thoại “Save as Type” thành “All Files” trước khi lưu tập tin, nếu không nó sẽ bị lưu với kiểu tập tin là txt. Lưu tập tin này trong đường dẫn tìm kiếm của AutoCad.

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. (Tiếp theo ta sẽ copy chương trình Autolisp mẫu và chỉnh sửa lại. Các mã mới sẽ được trình bày màu đỏ.)

```
(defun C:SAMPLE1 ()
```

```

;;;--- Load the dcl file (Tải tập tin DCL)
(setq dcl_id (load_dialog "SAMPLE1.dcl"))

;;;--- Load the dialog definition if it is not already loaded (Tải định nghĩa hộp thoại nếu
nó chưa được tải)
(if (not (new_dialog "SAMPLE1" dcl_id))
    (progn
      (alert "The SAMPLE1.DCL file could not be loaded!")
      (exit)
    )
)

;;;--- If an action event occurs, do this function (Nếu có một sự hành động, thực hiện hàm)
(action_tile "cancel" "(done_dialog)")

;;;--- Display the dialog box (Hiển thị hộp thoại)
(start_dialog)

;;;--- Unload the dialog box (Thoát khỏi hộp thoại)
(unload_dialog dcl_id)

;;;--- Suppress the last echo for a clean exit (Triệt tiêu việc lặp lại lệnh cuối và thoát êm)
(princ)

)

```

I removed several lines from the model. I took out the part that checked to see if we hit the Cancel or Okay buttons. We don't need either in this program. I also removed the action tile for the okay button and removed "(setq ddiag 1)" from the cancel button.

Tôi xóa đi một vài dòng từ chương trình mẫu. Tôi lấy đi phần kiểm soát xem liệu ta có kích các nút Okay hay Cancel không. Ta không cần điều đó trong chương trình này. Tôi cũng xóa đi phần tử hành động đối với nút Okay và xóa đi “setq ddiag 1” từ phần tử hành động đối với nút Cancel.

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE1.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Click chuột phải và copy tập tin trên. Mở NotePad và dán nó vào. Lưu lại nó dưới tên SAMPLE1.LSP . Phải đảm bảo là bạn đã thay đổi giá trị của hộp thoại "Save as Type" thành "All Files" trước khi lưu tập tin, nếu không nó sẽ bị lưu với kiểu tập tin là txt. Lưu tập tin này trong đường dẫn tìm kiếm của AutoCad.

Let's load the program and see what the DCL file looks like. On the command line type this: (Ta hãy tải chương trình và xem tập tin DCL ra sao. Trên dòng lệnh nhập như sau:)

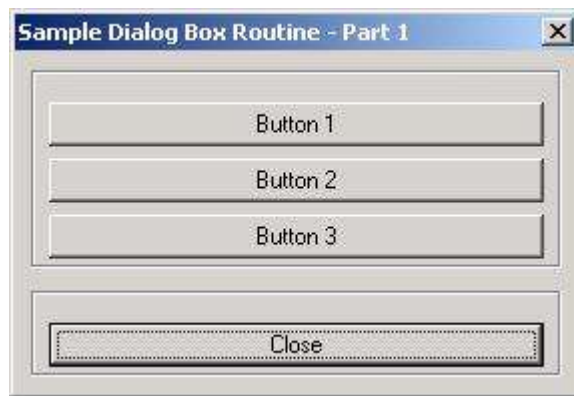
Command: (load "sample1") and press enter (Nhập (load “sample1”) rồi nhấn Enter)

You should see this (Bạn sẽ thấy như sau:)

C:Sample1
Command:

Now type Sample1 and press enter. If everything went according to plan you should see this on your screen:

Bây giờ nhập Sample1 và nhấn Enter. Nếu mọi thứ đúng như kế hoạch, bạn sẽ thấy trên màn hình:



The buttons do not work yet. Except for the close button. It should work fine. We need to add the function to print three different messages when the user presses each button. Let's send a parameter to the function to decide which message to display. If the parameter equals 1 we will print the first message. If 2 then print the second message. If 3 print the third message. Something like this:

Các nút vẫn chưa làm việc ngoại trừ nút Close. Chúng sẽ phải làm việc thôi. Ta cần thêm hàm để in ba thông báo khác nhau khi người sử dụng nhấn vào mỗi nút. Ta sẽ gửi thông số vào hàm để quyết định thông báo nào sẽ hiển thị. Nếu thông số bằng 1 ta sẽ in thông báo thứ nhất. Nếu thông số bằng 2 ta sẽ in thông báo thứ hai. Nếu thông số là 3 ta sẽ in thông báo thứ ba. Hàm sẽ như sau:

```
(defun doButton(a)
  (cond
    ((= a 1) (alert "Button 1 was pressed!"))
    ((= a 2) (alert "Button 2 was pressed!"))
    ((= a 3) (alert "Button 3 was pressed!"))
  )
)
```

Now we need to add the action calls for each of the buttons: (Bây giờ ta cần phải thêm các lệnh gọi phân tử hành động cho mỗi một nút)

```
(action_tile "but1" "(doButton 1)")
(action_tile "but2" "(doButton 2)")
```



```
(action_tile "but3" "(doButton 3)")
```

This will send a parameter of 1,2, or 3 to the doButton function depending on which button is pressed. (Điều này sẽ gửi các thông số 1,2 và 3 vào hàm doButton tùy theo nút nào được nhấn)

Let's add the doButton function and the action calls to the autolisp program. It should look like this:

Ta hãy thêm hàm doButton và các lệnh gọi hành động vào chương trình Autolisp. Nó sẽ giống như sau:

```
(defun doButton(a)
  (cond
    ((= a 1) (alert "Button 1 was pressed!"))
    ((= a 2) (alert "Button 2 was pressed!"))
    ((= a 3) (alert "Button 3 was pressed!"))
  )
)

(defun C:SAMPLE1()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE1.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE1" dcl_id))
    (progn
      (alert "The SAMPLE1.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "but1" "(doButton 1)")
  (action_tile "but2" "(doButton 2)")
  (action_tile "but3" "(doButton 3)")
  (action_tile "cancel" "(done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- Suppress the last echo for a clean exit
  (princ)

)
```

Save it and test it out. Everything working okay? (Lưu nó lại và thử chạy. Mọi thứ làm việc tốt chứ?)



When you get your program tested and everything is working, move the blue line above, [`(defun C: SAMPLE1 ()`] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

Khi bạn đã chạy thử chương trình và mọi thứ đều làm việc tốt, hãy chuyển dòng chữ màu xanh [`(defun C: SAMPLE1 ()`] lên trên cùng của tập tin. Điều này sẽ làm cho tất cả các biến của bạn thành biến cục bộ và đặt lại chúng về nil khi chương trình kết thúc.

That's it. We're done. Vậy đó, Ta đã hoàn thành.

[Back](#)

Dialog Control Language - Part 2 (Phần thực hành 2 về DCL)

Part 2 - Edit_Box (Phần 2 - Hộp hiệu chỉnh)

Let's build a working DCL file showing us exactly how to handle edit boxes. (Ta sẽ xây dựng tập tin DCL để trình bày chính xác cách điều khiển hộp hiệu chỉnh)

We will build a DCL file containing two edit boxes plus a Okay and Cancel button. The information in the two edit boxes will be displayed after selecting the Okay button. (Ta sẽ tạo một tập tin DCL chứa hai hộp hiệu chỉnh và bộ nút Okay và Cancel. Thông tin trong hai hộp hiệu chỉnh sẽ được hiển thị sau khi nhấn nút Okay.)

Layout thoughts: I will place the edit boxes in a column, (stacked on top of each other). Then I'll put the Okay and Cancel buttons at the bottom in a row. So...I'll need something like this:

Thiết kế hộp thoại: Tôi sẽ đặt các hộp hiệu chỉnh trong một cột (các hộp chồng lên nhau). Sau đó đặt các nút Okay và Cancel vào hàng dưới cùng. Vậy nên tôi cần một thứ như sau:

```

: column {
  : boxed_column {
    : edit_box {
      // Put code for edit_box 1 here (Đặt mã của hộp hiệu chỉnh 1 ở đây)
    }
    : edit_box {
      // Put code for edit_box 3 here (Đặt mã của hộp hiệu chỉnh 3 ở đây)
    }
  }
}
: boxed_row {
  : button {
    // Put code for the Okay button here (Đặt mã của nút Okay ở đây)
  }
  : button {
    // Put code for the Cancel button here (Đặt mã của nút Cancel ở đây)
  }
}
}

```

Let's copy in the code for the header and all of the controls above. I'll show them in red. Notice the key names and labels had to be changed. (Ta sẽ copy phần tiêu đề và tất cả các mã điều khiển ở mẫu trên. Tôi sẽ thể hiện chúng màu đỏ. Chú ý là các tên khoá và các nhãn đã được thay đổi.)

```

SAMPLE2 : dialog {
  label = "Sample Dialog Box Routine - Part 2";
  : column {
    : boxed_column {
      : edit_box {
        key = "username";
        label = "Enter your Name:";
        edit_width = 15;
        value = "";
        initial_focus = true;
      }
      : edit_box {
        key = "userage";
        label = "Enter your Age:";
        edit_width = 15;
        value = "";
      }
    }
  }
  : boxed_row {
    : button {
      key = "accept";
    }
  }
}

```

}

copy chương trình Autolisp màu và chỉnh sửa lại. Các mã mới được thể hiện bằng màu đỏ.)

```
(defun C: SAMPLE2()

  ;;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE2.dcl"))

  ;;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE2" dcl_id))
      (progn
        (alert "The SAMPLE2.DCL file could not be loaded!")
        (exit)
      )
  )

  ;;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)") ;Chú ý thẳng SaveVars
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;;--- Display the dialog box
  (start_dialog)

  ;;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;;--- If the user pressed the Cancel button
  (if(= ddiag 1)
      (princ "\n Sample2 cancelled!")
  )

)
```

```

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn
    (princ "\n The user pressed Okay!")
  )
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE2.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Let's load the program and see what the DCL file looks like. On the command line type this:

Command: (load "sample2") and press enter

You should see this

C:Sample2
Command:

Now type Sample2 and press enter. If everything went according to plan you should see this on your screen:



Looking good so far. We need to add the [SaveVars](#) function to save the strings in the edit boxes when the Okay button is pressed. *Look at the blue text in the Sample2.lsp program above. (Trông quá đẹp. Ta cần thêm hàm SaveVars để lưu lại chuỗi trong hộp hiệu chỉnh khi nút Okay được nhấn. Hãy xem dòng chữ màu xanh trong chương trình Autolisp Sample2.lsp ở trên)*

The edit box for the name needs to be a string. Dialog boxes return strings, so we do not need to modify it. All we have to do is use the get_tile function. *(Hộp hiệu chỉnh dùng cho tên cần nhập một chuỗi. Các hộp*

thoại trả về giá trị là các chuỗi, vì thế ta không cần phải sửa nó. Mọi thứ ta cần làm là sử dụng hàm get_tile)

```
(setq userName(get_tile "username"))    ; Cú pháp này tớ chưa thông, tạm hiểu là hàm
;;get_tile trả về một chuỗi được nhập trong
;;;edit_box có khóa là đối số của nó.
```

The edit box for Age needs to be an integer. We will have to modify the get_tile results by using the ATOI function. This function converts a string to an integer. (Hộp hiệu chỉnh dùng cho tuổi cần nhập một số nguyên. Ta sẽ phải sửa kết quả của hàm get_tile bằng cách sử dụng hàm ATOI. Hàm này đổi một chuỗi thành một số nguyên)

```
(setq userAge( atoi (get_tile "userage")))
```

If we needed to convert to an Real number we would use DISTOF function instead of the ATOI function.
(Nếu ta cần đổi thành một số thực thì phải sử dụng hàm DISTOF thay cho hàm ATOI)

Our SaveVars routine would look like this: (Vòng lưu biến của ta sẽ trông giống như sau:)

```
(defun saveVars()
  (setq userName(get_tile "username"))
  (setq userAge( atoi (get_tile "userage")))
)
```

Our program would now look like this: (Chương trình lúc này sẽ như sau:)

```
(defun saveVars()
  (setq userName(get_tile "username"))
  (setq userAge(atoi(get_tile "userage")))
)

(defun C:SAMPLE2()

  ;;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE2.dcl"))

  ;;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE2" dcl_id))
    (progn
      (alert "The SAMPLE2.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2)(saveVars)(done_dialog)")
  (action_tile "cancel" "(setq ddiag 1)(done_dialog)")

  ;;;--- Display the dialog box
  (start_dialog)
```

```

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the user pressed the Cancel button
(if(= ddiag 1)
  (princ "\n Sample2 cancelled!")
)

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn
    (princ "\n The user pressed Okay!")
  )
)

;;;--- Suppress the last echo for a clean exit
(princ)
)

```

Last item. We need to replace the line in the program: `(princ "\n The user pressed Okay!")` with something to modify and display the `userName` and `userAge` data. Let's do something simple. We will find out how many days old this person is. (Tí cuối cùng. Ta cần thay thế dòng thông báo `(princ "\n The user pressed Okay!")` trong chương trình bằng tí sửa đổi và hiển thị Tên người sử dụng cùng với số ngày tuổi của hân. Ta sẽ làm tí đơn giản này. Hãy tìm ra số ngày tuổi của gã.)

```

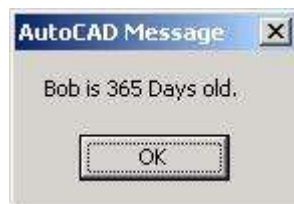
;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn

    ;;;;--- Multiply the users age x 365 to get the number of days. (Nhân số tuổi với 365)
    (setq userAge(* userAge 365))

    ;;;;--- Display the results (Hiển thị kết quả)
    (alert (strcat userName " is " (itoa userAge) " days old. "))
  )
)

```

Add the above to the file, save it and test it out. Everything working okay? (Thêm những điều trên vào chương trình. Lưu lại và chạy thử. Mọi thứ chạy tốt chứ?)



When you get your program tested and everything is working, move the blue line above, [`(defun C: SAMPLE2 ()`] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done. (Biết rồi, khổ lắm...)

[Back](#)

Dialog Control Language - Part 3 (Phần thực hành 3 về DCL)

Part 3 - List_Box (Phần 3 - Hộp danh sách)

Let's build a working DCL file showing us exactly how to handle list boxes. (Ta sẽ xây dựng một tập tin DCL hoạt động để chỉ ra chính xác cách điều khiển một hộp danh sách)

We will build a DCL file containing two list boxes plus an Okay and Cancel button. We will set the first list box to single selection and the second to multiple selection. The selected items will be displayed on the screen after the user presses the Okay button. (ta sẽ tạo ra một tập tin DCL chứa hai hộp danh sách với bộ nút Okay và Cancel. Ta sẽ đặt hộp danh sách thứ nhất là lựa chọn đơn và hộp danh sách thứ hai là đa lựa chọn. Các khoản mục được lựa chọn sẽ được hiển thị trên màn hình sau khi người sử dụng nhấn nút Okay.)

Layout thoughts: I will place the list boxes in a row, (side by side). Then I'll put the Okay and Cancel buttons in a row at the bottom of the dialog box. So...I'll need something like this:

Thiết kế hộp thoại: Tôi sẽ đặt các hộp danh sách trong một hàng (Các hộp cạnh nhau). Sau đó đặt các nút Okay và Cancel vào hàng dưới cùng của hộp thoại. Vì thế tôi sẽ cần một thứ như sau:

```
: column {
  : boxed_row {
    : list_box {
      // Put code for list_box 1 here (Đặt mã cho hộp danh sách 1 ở đây)
    }
    : list_box {
      // Put code for list_box 2 here (Đặt mã cho hộp danh sách 2 ở đây)
    }
  }
}
```



```

}
: boxed_row {
: button {
    // Put code for the Okay button here (Đặt mã cho nút Okay ở đây)
}
: button {
    // Put code for the Cancel button here (Đặt mã cho Nút Cancel ở đây)
}
}
}
}

```

Let's copy in the code for the header and all of the controls above from the "[Controls](#)" section of this tutorial. I'll show them in red. Notice the key names and labels had to be changed. (Ta hãy copy các mã tiêu đề và tất cả các mã điều khiển ở trên từ phần các nút điều khiển trong tài liệu này. Tôi sẽ thể hiện chúng bằng màu đỏ. Chú ý là các tên khóa và các nhãn đã được thay đổi)

```

SAMPLE3 : dialog {
    label = "Sample Dialog Box Routine - Part 3";
: column {
: boxed_row {
: list_box {
    label = "Choose Item";
    key = "mylist1";
    height = 15;
    width = 25;
    multiple_select = false;
    fixed_width_font = true;
    value = "";
}
: list_box {
    label = "Choose Items";
    key = "mylist2";
    height = 15;
    width = 25;
    multiple_select = true;
    fixed_width_font = true;
    value = "";
}
}
: boxed_row {
: button {
    key = "accept";
    label = " Okay ";
    is_default = true;

```

```

    }
    : button {
      key = "cancel";
      label = " Cancel ";
      is_default = false;
      is_cancel = true;
    }
  }
}
}

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE3.DCL *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. (Tiếp theo ta sẽ lấy một phiên bản của chương trình Autolisp mẫu và chỉnh sửa lại. Tất cả mã mới được thể hiện bởi màu đỏ)

```

(defun C: SAMPLE3 ()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE3.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE3" dcl_id))
    (progn
      (alert "The SAMPLE3.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)") ;Chú ý hàm SaveVars
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- If the user pressed the Cancel button
  (if (= ddiag 1)
    (princ "\n Sample3 cancelled!")
  )

  ;;--- If the user pressed the Okay button
  (if (= ddiag 2)
    (progn

```

```

        (princ "\n The user pressed Okay!")
    )
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE3.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Let's load the program and see what the DCL file looks like. On the command line type this:

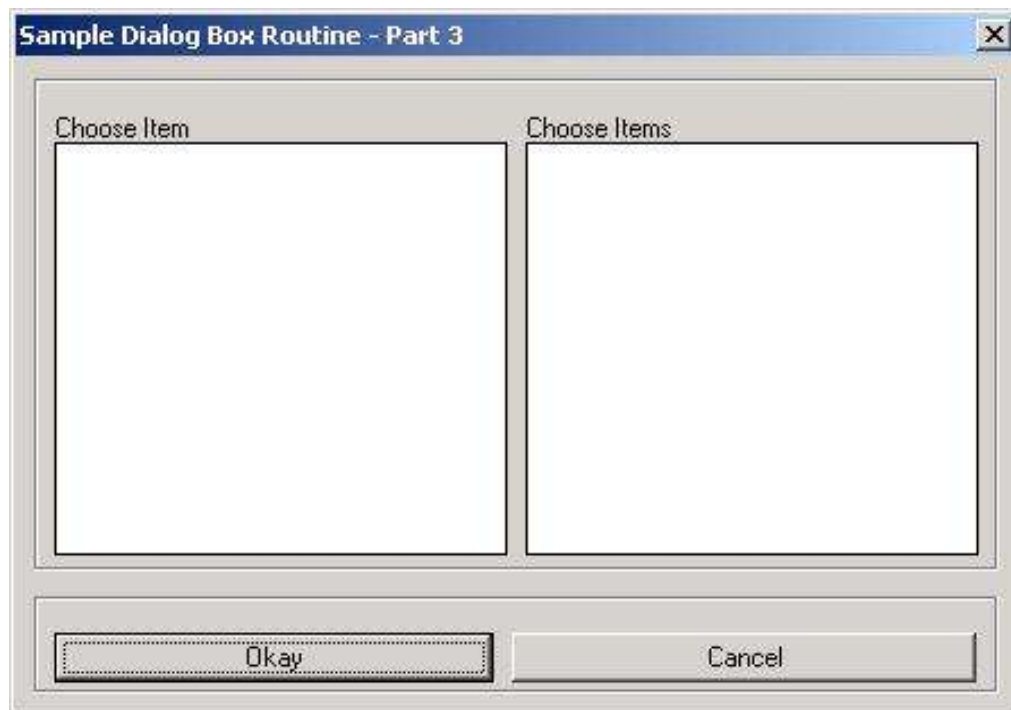
Command: (load "sample3") and press enter

You should see this

C:Sample3

Command:

Now type Sample3 and press enter. If everything went according to plan you should see this on your screen:



Looks good but, there is nothing to choose. Let's add some data to the list boxes. We will need two list. We will call the list for the first list box myList1 and the second myList2. (Trông đẹp đấy nhưng chưa có gì để chọn. Ta sẽ thêm dữ liệu vào hộp danh sách. Ta cần hai danh sách. Ta sẽ gọi hàm list để tạo danh sách thứ nhất là myList1 và danh sách thứ hai là myList2.)

```
(setq myList1 (list "Electrical" "Structural" "Plumbing" "Foundation"))
```

```
(setq myList2 (list "Plastic" "Steel" "Aluminum" "Concrete"))
```

Alrighty then, we have our list built. All we have to do is put them in the dialog box. We will use the start_list, add_list, and end_list functions. Start_list tells DCL which list_box we are going to edit. The identification is made by using the list_box KEY. The first list has a key of "mylist1" and the second has a key of "mylist2". So the start_list function would look like this:

Vậy là ta đã tạo các danh sách của mình. Điều phải làm là đặt nó vào trong hộp danh sách. Ta sẽ sử dụng các hàm start_list, add_list và end_list. Hàm start_list sẽ bảo DCL hộp danh sách nào sẽ phải sửa lại. Việc chỉ định được thực hiện bằng cách sử dụng khóa của hộp danh sách. Danh sách thứ nhất có khoá là “mylist1” và danh sách thứ hai có khoá là “mylist2”. Do đó hàm start_list sẽ như sau:

```
(start_list "mylist1" 3) ; The 3 means we want to delete the old contents and start new. (Tham số 3 có nghĩa là ta muốn xoá các nội dung cũ và bắt đầu danh sách mới)
```

Next we use the add_list function to tell DCL which list to put in the list_box. We use the mapcar function to apply add_list to each member in the list. Our list for the first list_box is named myList1. So... (Tiếp theo sử dụng hàm add_list để bảo DCL danh sách nào sẽ được đặt vào trong hộp danh sách. Ta sử dụng hàm mapcar để áp dụng hàm add_list cho mỗi thành viên trong danh sách đó. Danh sách của chúng ta dùng cho hộp danh sách thứ nhất là myList1. Vì thế ta có:

```
(mapcar 'add_list myList1)
```

Finally we use the end_list function to tell DCL to display the new contents because we are through editing the list. (Cuối cùng ta dùng hàm End_list để bảo DCL hiển thị nội dung mới vì ta đã sửa xong danh sách đó)

```
(end_list)
```

To look at it all together: Hãy xem toàn bộ công việc này:

```
(start_list "mylist1" 3)
(mapcar 'add_list myList1)
(end_list)
```

```
(start_list "mylist2" 3)
(mapcar 'add_list myList2)
(end_list)
```

Let's add all of this to the AutoLisp program and see what it looks like. (Thêm các mã này vào chương trình)

```
(defun C:SAMPLE3()

  (setq myList1(list "Electrical" "Structural" "Plumbing" "Foundation"))
  (setq myList2(list "Plastic" "Steel" "Aluminum" "Concrete"))

  ;;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE3.dcl"))

  ;;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE3" dcl_id))
      (progn
        (alert "The SAMPLE3.DCL file could not be loaded!")
        (exit)
      )
  )

  (start_list "mylist1" 3)
  (mapcar 'add_list myList1)
  (end_list)

  (start_list "mylist2" 3)
  (mapcar 'add_list myList2)
  (end_list)

  ;;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)") ;Chú ý hàm SaveVars
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;;--- Display the dialog box
  (start_dialog)

  ;;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;;--- If the user pressed the Cancel button
  (if(= ddiag 1)
      (princ "\n Sample3 cancelled!")
  )

  ;;;--- If the user pressed the Okay button
  (if(= ddiag 2)
      (progn
        (princ "\n The user pressed Okay!")
      )
  )

  ;;;--- Suppress the last echo for a clean exit
  (princ)

)
```

Notice the location of the red lines. We create the list before loading the dialog box. We add the list to the dialog box after it is loaded with `new_dialog` and before the `action_tile` statements. This is the order you should use.

Chú ý tới vị trí của các dòng màu đỏ. Ta tạo danh sách trước khi tải hộp thoại. Ta thêm danh sách vào hộp thoại sau khi nó được tải với hàm `new_dialog` và trước hàm thông báo phản tử hành động. Đây là trật tự bạn nên dùng

Looking good so far. We need to add the `SaveVars` function to save the selected items from the list boxes when the Okay button is pressed. *Look at the blue text in the `Sample3.lsp` program above.*

Tiếp tục thêm chút nữa. Ta cần thêm hàm `SaveVars` để lưu các khoản mục được chọn từ hộp danh sách khi nút Okay được nhấn. *Xem dòng chữ màu xanh trong chương trình Autolisp `Sample3.lsp` ở trên.*

Let's steal the `saveVars` routine from the `list_box` control on the "[List and how to handle them](#)" page of this tutorial and modify it. I'll show the modifications in red.

Ta sẽ ăn cắp vòng lưu biến `Savevars` từ chương trình điều khiển hộp danh sách trong trang “Danh sách và cách điều khiển chúng” của tài liệu tự học này và sửa đổi nó. Tôi sẽ thể hiện phần sửa đổi bằng màu đỏ.

```
(defun saveVars(/ readlist count item)

  ;;--- Setup a list to hold the selected items
  (setq retList(list))

  ;;--- Save the list setting
  (setq readlist(get_tile "mylist1"))

  ;;--- Setup a variable to run through the list
  (setq count 1)

  ;;--- cycle through the list getting all of the selected items
  (while (setq item (read readlist))
    (setq retlist(append retList (list (nth item myList1))))
    (while
      (and
        (/= " " (substr readlist count 1))
        (/= "" (substr readlist count 1))
      )
      (setq count (1+ count))
    )
    (setq readlist (substr readlist count))
  )
)
```

Wow! That was easy. Wait a minute, we have two list boxes. We will have to create a function out of this or simply copy this and do it twice. For now, let's just do it twice. (Ồ , quá dễ rồi. Hượm đã, ta có hai hộp danh sách. Ta sẽ phải tạo một hàm bao gồm cả hai hộp danh sách này hoặc đơn giản là copy nó thành hai lần.

Bây giờ ta sẽ làm như vậy.)

Our program would now look like this: (Chương trình của chúng ta sẽ trông giống như sau:)

```
(defun saveVars(/ readlist count item)

  ;;--- Setup a list to hold the selected items
  (setq retList(list))

  ;;--- Save the list setting
  (setq readlist(get_tile "mylist1"))

  ;;--- Setup a variable to run through the list
  (setq count 1)

  ;;--- cycle through the list getting all of the selected items
  (while (setq item (read readlist))
    (setq retlist(append retList (list (nth item myList1))))
    (while
      (and
        (/= " " (substr readlist count 1))
        (/= "" (substr readlist count 1))
      )
      (setq count (1+ count))
    )
    (setq readlist (substr readlist count))
  )

  ;;--- Setup a list to hold the selected items
  (setq retList2(list))

  ;;--- Save the list setting
  (setq readlist(get_tile "mylist2"))

  ;;--- Setup a variable to run through the list
  (setq count 1)

  ;;--- cycle through the list getting all of the selected items
  (while (setq item (read readlist))
    (setq retlist2(append retList2 (list (nth item myList2))))
    (while
      (and
        (/= " " (substr readlist count 1))
        (/= "" (substr readlist count 1))
      )
      (setq count (1+ count))
    )
    (setq readlist (substr readlist count))
  )
)

(defun C:SAMPLE3()
```

```

(setq myList1(list "Electrical" "Structural" "Plumbing" "Foundation"))
(setq myList2(list "Plastic" "Steel" "Aluminum" "Concrete"))

;;;--- Load the dcl file
(setq dcl_id (load_dialog "SAMPLE3.dcl"))

;;;--- Load the dialog definition if it is not already loaded
(if (not (new_dialog "SAMPLE3" dcl_id))
    (progn
      (alert "The SAMPLE3.DCL file could not be loaded!")
      (exit)
    )
)

(start_list "mylist1" 3)
(mapcar 'add_list myList1)
(end_list)

(start_list "mylist2" 3)
(mapcar 'add_list myList2)
(end_list)

;;;--- If an action event occurs, do this function
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the user pressed the Cancel button
(if(= ddiag 1)
    (princ "\n Sample3 cancelled!")
)

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
    (progn
      (princ "\n The user pressed Okay!")
    )
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Last item. We need to replace the line in the program: `(princ "\n The user pressed Okay!")` with something to modify and display the selected items. Let's do something simple. We will tell the user what was selected out of each list.. (Khoản cuối cùng. Ta cần thay thế dòng `(princ "\n The user pressed Okay!")` trong chương trình bằng điều gì đó để sửa đổi và hiển thị các khoản mục được chọn. Ta hãy làm điều đơn giản

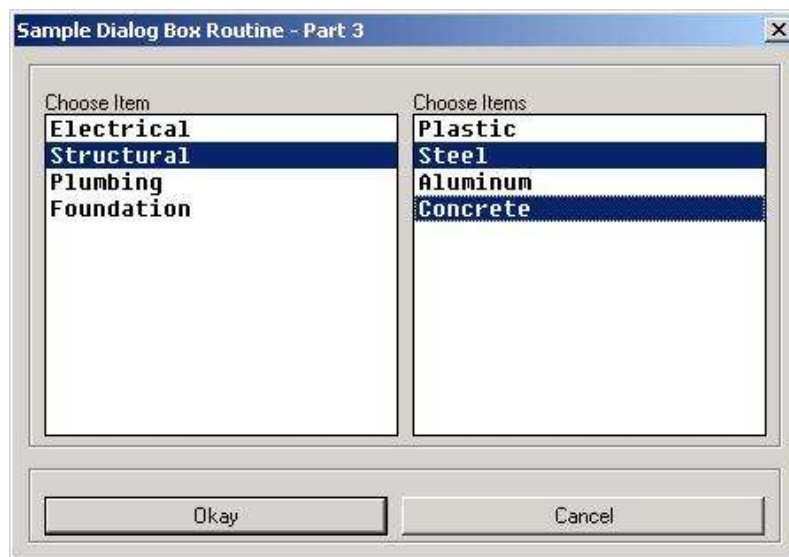
sau. Bảo cho người sử dụng biết cái gì đã được chọn ra từ mỗi một danh sách.)

```
;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn

    ;;;;--- Inform the user of his selection from the first list
    (princ (strcat "\n You chose " (car retList) " from the first list box.))

    ;;;;--- Inform the user of his selections from the second list
    (princ "\n Your choice(s) from the second list box :")
    (foreach a retList2
      (princ "\n ")
      (princ a)
    )
  )
)
```

Add the above to the file, save it and test it out. Everything working okay? (Thêm những điều trên vào tập tin, lưu lại và chạy thử. Mọi thứ làm việc tốt cả chứ?)



```
Command: (load "sample3")
C: SAMPLE3

Command: Sample3

You chose Structural from the first list box.
Your choice(s) from the second list box :
Steel
Concrete

Command: |
```

When you get your program tested and everything is working, move the blue line above, [`(defun C: SAMPLE3 ()`] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done.

[Back](#)

Dialog Control Language - Part 4 (Phần thực hành 4 về DCL)

Part 4 - PopUp_List (Phần 4- Danh sách thả xuống)

Let's build a working DCL file showing us exactly how to handle popup list. (Ta sẽ xây dựng một tập tin DCL hoạt động để chỉ cho thấy chính xác cách điều khiển danh sách thả xuống)

We will build a DCL file containing two pop_up list plus an Okay and Cancel button. The selected items will be displayed on the screen after the user presses the Okay button. (Ta sẽ tạo một tập tin DCL chứa hai danh sách thả xuống với bộ nút Okay và Cancel. Các khoản mục được chọn sẽ hiển thị trên màn hình sau khi người sử dụng nhấn nút Okay)

Layout thoughts: I will place the popup_list in a row, (side by side). Then I'll put the Okay and Cancel buttons in a row at the bottom of the dialog box. So...I'll need something like this:

Thiết kế hộp thoại: Tôi sẽ đặt các danh sách thả trong một hàng, (nằm cạnh nhau). Sau đó tôi đặt các nút Okay và Cancel vào hàng dưới cùng của hộp thoại. Vì thế tôi sẽ cần như sau:

```
: column {
  : boxed_row {
    : popup_list {
      // Put code for popup_list 1 here (Đặt mã cho danh sách thả xuống 1 ở đây)
    }
    : popup_list {
      // Put code for popup_list 2 here (Đặt mã cho danh sách thả xuống 2 ở đây)
    }
  }
}
```

```

: boxed_row {
: button {
    // Put code for the Okay button here (Đặt mã cho nút Okay ở đây)
}
: button {
    // Put code for the Cancel button here (Đặt mã cho nút Cancel ở đây)
}
}
}
}

```

Let's copy in the code for the header and all of the controls above from the "[Controls](#)" section of this tutorial. I'll show them in red. Notice the key names and labels had to be changed. (Ta hãy copy các mã phần tiêu đề và tất cả các nút điều khiển trên đây từ phần “Các nút điều khiển” của tài liệu tự học này. Tôi sẽ trình bày chúng bằng màu đỏ. Chú ý là các tên khoá và các nhãn đã được thay đổi.)

```

SAMPLE4 : dialog {
    label = "Sample Dialog Box Routine - Part 4";
: column {
: boxed_row {
: popup_list {
    key = "mylist1";
    label = "Select Item";
    fixed_width_font = true;
    width = 30;
    value = "";
}
: popup_list {
    key = "mylist2";
    label = "Select Item";
    fixed_width_font = true;
    width = 30;
    value = "";
}
}
: boxed_row {
: button {
    key = "accept";
    label = " Okay ";
    is_default = true;
}
: button {
    key = "cancel";
    label = " Cancel ";
    is_default = false;
    is_cancel = true;
}
}
}
}

```

```

    }
  }
}

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE4.DCL *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. (Tiếp theo ta sẽ lấy một phiên bản của chương trình Autolisp mẫu và sửa đổi lại nó. Tất cả mã mới đều thể hiện bởi màu đỏ.)

```

(defun C: SAMPLE4 ()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE4.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE4" dcl_id))
    (progn
      (alert "The SAMPLE4.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)") ; Chú ý hàm SaveVars
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- If the user pressed the Cancel button
  (if(= ddiag 1)
    (princ "\n Sample4 cancelled!")
  )

  ;;--- If the user pressed the Okay button
  (if(= ddiag 2)
    (progn
      (princ "\n The user pressed Okay!")
    )
  )

  ;;--- Suppress the last echo for a clean exit
  (princ)

)

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE4.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Let's load the program and see what the DCL file looks like. On the command line type this:

Command: (load "sample4") and press enter

You should see this

C:Sample4

Command:

Now type Sample4 and press enter. If everything went according to plan you should see this on your screen:



Looks good but, there is nothing to choose. Let's add some data to the popup list boxes. We will need two list. We will call the list for the first popup list box myList1 and the second myList2. (Trông thì ngon đấy nhưng chưa có gì để chọn cả. Ta sẽ thêm dữ liệu vào các hộp danh sách thả xuống. Ta cần hai danh sách. Gọi danh sách cho hộp danh sách thả xuống thứ nhất là myList1 và cho hộp thứ hai là myList2.)

```
(setq myList1(list "Electrical" "Structural" "Plumbing" "Foundation"))
```

```
(setq myList2(list "Plastic" "Steel" "Aluminum" "Concrete"))
```

Alrighty then, we have our list built. All we have to do is put them in the dialog box. We will use the start_list, add_list, and end_list functions. Start_list tells DCL which popup list box we are going to edit. The identification is made by using the popup list box KEY. The first popup list box has a key of "mylist1" and the second has a key of "mylist2". So the start_list function would look like this: (Rồi, vậy là ta đã tạo xong các danh sách. Ta chỉ việc đặt chúng vào hộp thoại. Ta sẽ sử dụng các hàm start_list, add_list, và end_list. Hàm start_list dùng để xác định hộp danh sách nào sẽ được chỉnh sửa. Để chỉ định hộp, sử dụng khoá của hộp. Hộp danh sách thả xuống thứ nhất có khoá là "mylist1" và hộp thứ hai có khoá là "mylist2". Vì thế hàm

start_list sẽ như sau:

(start_list "mylist1" 3) ; The 3 means we want to delete the old contents and start new. (Thêm số 3 có nghĩa là ta sẽ xóa toàn bộ nội dung cũ và bắt đầu danh sách mới.)

Next we use the add_list function to tell DCL which list to put in the popup list box. We use the mapcar function to apply add_list to each member in the list. Our list for the first popup list box is named myList1. So... (Tiếp theo ta sử dụng hàm add_list để xác định mục nào cần đặt vào trong hộp danh sách thả xuống. Ta sử dụng hàm mapcar để áp dụng hàm ADD_list cho từng thành viên của danh sách. Danh sách ta dùng cho hộp danh sách thả xuống thứ nhất là myList1. Vì thế...)

```
(mapcar 'add_list myList1)
```

Finally we use the end_list function to tell DCL to display the new contents because we are through editing the popup list box. (Cuối cùng ta sử dụng hàm end_list để hiển thị nội dung mới của hộp danh sách thả xuống vì ta đã chỉnh sửa xong.)

```
(end_list)
```

To look at it all together: (Gộp chúng lại với nhau:)

```
(start_list "mylist1" 3)
(mapcar 'add_list myList1)
(end_list)
```

```
(start_list "mylist2" 3)
(mapcar 'add_list myList2)
(end_list)
```

Let's add all of this to the AutoLisp program and see what it looks like. (Thêm tất cả các mã trên vào chương trình Autolisp và ta thấy như sau:

```
(defun C:SAMPLE4()

  (setq myList1(list "Electrical" "Structural" "Plumbing" "Foundation"))
  (setq myList2(list "Plastic" "Steel" "Aluminum" "Concrete"))

  ;;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE4.dcl"))

  ;;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE4" dcl_id))
      (progn
        (alert "The SAMPLE4.DCL file could not be loaded!")
        (exit)
      )
  )
)
```

```

)

(start_list "mylist1" 3)
(mapcar 'add_list myList1)
(end_list)

(start_list "mylist2" 3)
(mapcar 'add_list myList2)
(end_list)

;;;--- If an action event occurs, do this function
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the user pressed the Cancel button
(if(= ddiag 1)
  (princ "\n Sample4 cancelled!")
)

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn
    (princ "\n The user pressed Okay!")
  )
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Notice the location of the red lines. We create the list before loading the dialog box. We add the list to the dialog box after it is loaded with `new_dialog` and before the `action_tile` statements. This is the order you should use. (Chú ý tới vị trí của các dòng màu đỏ. Ta tạo danh sách trước khi tải hộp thoại. Thêm danh sách vào hộp thoại sau khi nó đã được tải bởi hàm `new_dialog` và trước hàm thông báo phản từ hành động `action_tile`. Đó là trật tự mà bạn nên sử dụng.)

Looking good so far. We need to add the `SaveVars` function to save the selected items from the popup list boxes when the Okay button is pressed. *Look at the blue text in the Sample4.lsp program above.* (Tiếp tục nữa nhé. Ta cần phải thêm hàm `SaveVars` để lưu lại các khoản mục đã được chọn từ hộp danh sách thả xuống khi nút Okay bị nhấn. Hãy xem dòng chữ màu xanh trong chương trình Autolisp `Sample4.lsp` ở trên.)

Let's steal the `saveVars` routine from the popup list box control on the "[Saving data from the dialog box](#)" page of this tutorial and modify it. I'll show the modifications in red. (Ta sẽ chôm vòng lưu biến `SaveVars` từ mục điều khiển hộp danh sách thả xuống trong phần “Lưu dữ liệu từ hộp thoại” của tài liệu này và sửa đổi lại. Tôi sẽ thể hiện các phần sửa đổi bằng màu đỏ.)

```

(defun saveVars()

  ;;--- Get the selected item from the first list
  (setq sStr1(get_tile "mylist1"))

  ;;--- Make sure something was selected...
  (if(= sStr1 "")>
    (setq myItem1 nil)
    (setq myItem1 (nth (atoi sStr1) myList1))
  )

)

```

Wow! That was easy. Wait a minute, we have two pop_up list boxes. We will have to create a function out of this or simply copy this and do it twice. For now, let's just do it twice. (Ồ, dễ quá nhỉ. Khoan đã đừng tưởng bở. Ta có hai hộp danh sách thả xuống cơ mà. Ta sẽ phải tạo một hàm cho cả hai hộp danh sách này hay đơn giản hơn ta copy điều này hai lần. Ở đây ta sẽ thực hiện hai lần copy.)

Our program would now look like this: (Chương trình của ta trông giống như sau:)

```

(defun saveVars()

  ;;--- Get the selected item from the first list
  (setq sStr1(get_tile "mylist1")) ;; Hàm (get_tile "mylist1") trả về số thứ tự của
khoản mục được chọn.

  ;;--- Make sure something was selected...
  (if(= sStr1 "")> ;; === Symbol ">" is excess and needs to obmit ===;;
    (setq myItem1 "Nothing")
    (setq myItem1 (nth (atoi sStr1) myList1))
  )

  ;;--- Get the selected item from the second list
  (setq sStr2(get_tile "mylist2"))

  ;;--- Make sure something was selected...
  (if(= sStr2 "")> ;; === Symbol ">" is excess and needs to obmit ===;;
    (setq myItem2 "Nothing")
    (setq myItem2 (nth (atoi sStr2) myList2))
  )

)

(defun C:SAMPLE4()

  (setq myList1(list "Electrical" "Structural" "Plumbing" "Foundation"))
  (setq myList2(list "Plastic" "Steel" "Aluminum" "Concrete"))

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE4.dcl"))

```



```

;;;--- Load the dialog definition if it is not already loaded
(if (not (new_dialog "SAMPLE4" dcl_id))
  (progn
    (alert "The SAMPLE4.DCL file could not be loaded!")
    (exit)
  )
)

(start_list "mylist1" 4)
(mapcar 'add_list myList1)
(end_list)

(start_list "mylist2" 4)
(mapcar 'add_list myList2)
(end_list)

;;;--- If an action event occurs, do this function
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the user pressed the Cancel button
(if (= ddiag 1)
  (princ "\n Sample4 cancelled!")
)

;;;--- If the user pressed the Okay button
(if (= ddiag 2)
  (progn
    (princ "\n The user pressed Okay!")
  )
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Last item. We need to replace the line in the program: `(princ "\n The user pressed Okay!")` with something to modify and display the selected items. Let's do something simple. We will tell the user what was selected out of each popup list.. (Cuối cùng, ta cần thay thế dòng `(princ "\n The user pressed Okay!")` bằng cái gì đó để sửa đổi và hiển thị các khoản mục được chọn. Hãy làm điều đơn giản này. Báo cho người sử dụng biết cái gì đã được chọn ra từ danh sách thả xuống)

```

;;;--- If the user pressed the Okay button
(if (= ddiag 2)

```

```

(progn

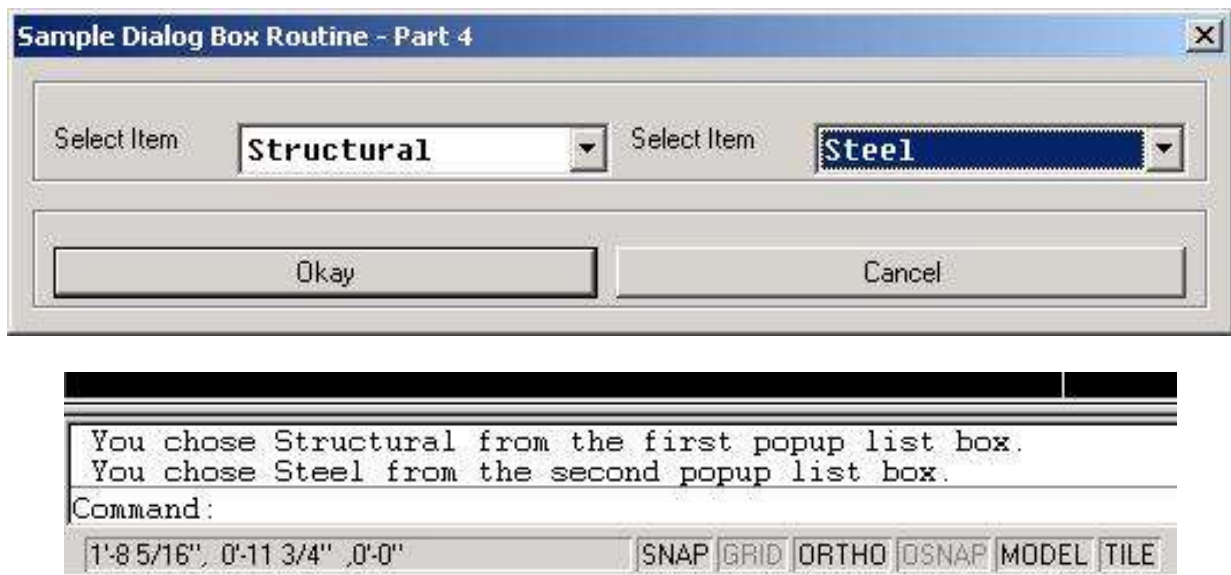
  ;;--- Inform the user of his selection from the first list
  (princ (strcat "\n You chose " myItem1 " from the first popup list box.))

  ;;--- Inform the user of his selections from the second list
  (princ (strcat "\n You chose " myItem2 " from the second popup list box.))

)
)

```

Add the above to the file, save it and test it out. Everything working okay? (Bổ sung những điều này vào tập tin, lưu lại và chạy thử nó. Mọi thứ làm việc tốt cả chứ ?)



When you get your program tested and everything is working, move the blue line above, [(defun C:SAMPLE4 ())] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done.

[Back](#)

Dialog Control Language - Part 5 (Phần thực hành 5 về DCL)

Part 5 - Radio Buttons (Phần 5 – Các nút lựa chọn)

Let's build a working DCL file showing us exactly how to handle radio buttons. (Ta sẽ xây dựng một tập tin DCL hoạt động để thể hiện chính xác cách điều khiển các nút lựa chọn)

The first thing you need to know about a radio button is how stupid they are. They have no brains. They do not know what the other radio buttons are doing. You can layout six radio buttons and select everyone of them like they were toggles. That's not the way we use radio buttons. They are supposed to be smart. They should know what to do if a radio button around them is selected. They should turn themselves off because only one radio button in a group is supposed to be checked. That's where `radio_column` and `radio_row` come in to play. They are the brains for the radio buttons. They watch all the buttons in their row or column to make sure only one is turned on. Okay..moving on. (Trước tiên, bạn cần biết là các nút lựa chọn rất ngu ngốc. Chúng không có óc. Chúng không cần biết tới các nút lựa chọn khác đang làm gì. Bạn có thể bố trí sáu nút lựa chọn và chọn một trong số đó như kiểu bật hay tắt chúng. Đó không phải cách mà ta sử dụng các nút lựa chọn. Chúng có nhiệm vụ là linh hoạt. Chúng phải biết làm gì nếu một nút lựa chọn quanh chúng được lựa chọn. Chúng sẽ tự tắt bởi vì chỉ duy nhất một nút lựa chọn trong nhóm có nhiệm vụ kiểm soát. Đó là nơi để các cột lựa chọn và các hàng lựa chọn hoạt động. Chúng là bộ óc của các nút lựa chọn. Chúng theo dõi tất cả các nút lựa chọn trong hàng hay cột của chúng để đảm bảo rằng chỉ có duy nhất một nút được kích hoạt. Rồi, tiếp tục.

We will build a DCL file containing 4 `radio_buttons` plus an Okay and Cancel button. The selected item will be displayed on the screen after the user presses the Okay button. (Ta sẽ tạo một tập tin DCL chứa bốn nút lựa chọn với bộ nút Okay và Cancel. Khoản mục được chọn sẽ được hiển thị trên màn hình sau khi người sử dụng nhấn nút Okay)

Layout thoughts: I will place the `radio_buttons` in a column, (stacked on top of each other). Then I'll put the Okay and Cancel buttons in a row at the bottom of the dialog box. So...I'll need something like this:

Thiết kế hộp thoại: Tôi sẽ đặt các nút lựa chọn trong một cột (Các nút chồng lên nhau). Rồi đặt các nút Okay và Cancel trong một hàng ở đáy hộp thoại. Vậy nên tôi sẽ cần một thứ như sau:

```
: column {
: radio_column {
// Put code for radio_column here (Đặt mã của cột lựa chọn ở đây)
: radio_column {
// Put code for radio_button 1 here (Đặt mã của nút lựa chọn 1 ở đây)
}
: radio_button {
// Put code for radio_button 2 here (Đặt mã của nút lựa chọn 2 ở đây)
}
: radio_button {
// Put code for radio_button 3 here (Đặt mã của nút lựa chọn 3 ở đây)
}
: radio_button {
// Put code for radio_button 4 here (Đặt mã của nút lựa chọn 4 ở đây)
}
}
: boxed_row {
: button {
```

```

        // Put code for the Okay button here  (Đặt mã của nút Okay ở đây)
    }
    : button {
        // Put code for the Cancel button here  (Đặt mã của nút Cancel ở đây)
    }
}
}

```

Let's copy in the code for the header and all of the controls above from the "[Controls](#)" section of this tutorial. I'll show them in red. Notice the key names and labels had to be changed. (Ta sẽ copy các mã của phần tiêu đề và tất cả các nút điều khiển trên đây từ phần “Các nút điều khiển” của tài liệu này. Tôi sẽ thể hiện chúng bằng màu đỏ. Chú ý rằng các tên khoá và các nhãn đã được thay đổi.)

```

SAMPLE5 : dialog {
    label = "Sample Dialog Box Routine - Part 5";
    : column {
        : radio_column {
            key = "mychoice";
        : radio_button {
            key = "but1";
            label = "Apples";
        }
        : radio_button {
            key = "but2";
            label = "Oranges";
        }
        : radio_button {
            key = "but3";
            label = "Bananas";
        }
        : radio_button {
            key = "but4";
            label = "Lemons";
        }
    }
    : boxed_row {
        : button {
            key = "accept";
            label = " Okay ";
            is_default = true;
        }
        : button {
            key = "cancel";
            label = " Cancel ";
            is_default = false;
            is_cancel = true;
        }
    }
}

```

```

    }
  }
}

```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE5.DCL *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. **Tiếp theo ta lấy một phiên bản của chương trình Autolisp mẫu và sửa đổi lại nó. Tất cả các mã mới được thể hiện màu đỏ.**

```

(defun C:SAMPLE5()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE5.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE5" dcl_id))
    (progn
      (alert "The SAMPLE5.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- If the user pressed the Cancel button
  (if(= ddiag 1)
    (princ "\n Sample5 cancelled!")
  )

  ;;--- If the user pressed the Okay button
  (if(= ddiag 2)
    (progn
      (princ "\n The user pressed Okay!")
    )
  )

  ;;--- Suppress the last echo for a clean exit
  (princ)
)

```

)

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE5.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Let's load the program and see what the DCL file looks like. On the command line type this:

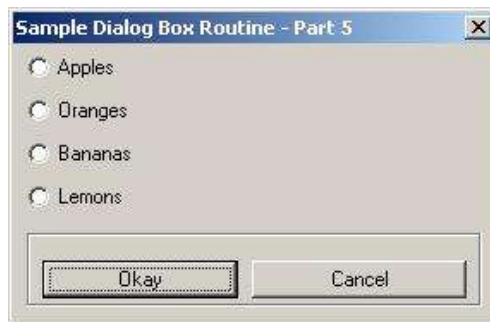
Command: (load "sample5") and press enter

You should see this

C:Sample5

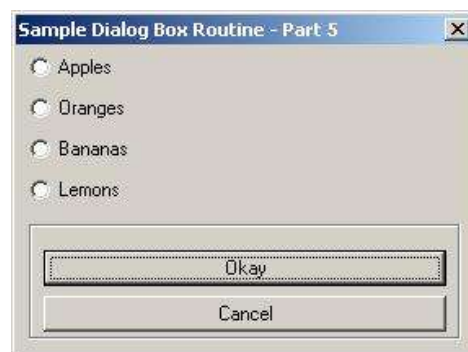
Command:

Now type Sample5 and press enter. If everything went according to plan you should see this on your screen:



That doesn't look very good does it? Let's change the [boxed_row](#) into a [boxed_column](#) in our DCL file. (See the blue text in the DCL file above) Make the changes then Save the Sample5.DCL file. No need to load the autolisp program again, it's loaded. Just run the Sample5 program again. Now it should look like this:

Trông nó không đẹp lắm phải không? Ta sẽ thay đổi hàng hộp thành cột hộp trong tập tin DCL. (Xem dòng chữ màu xanh trong tập tin DCL ở trên). Thực hiện sự thay đổi trong tập tin Sample5.DCL. Không cần tải lại nó vì nó đã được tải rồi. Chỉ cần chạy chương trình Sample5.lsp một lần nữa. Bây giờ kết quả sẽ như sau:



It still doesn't look right. It's our label "Sample Dialog Box Routine - Part 5" that is causing the problem. Let's shorten it to "SDBR - Part 5" and try it again: (Trông vẫn chưa đẹp lắm. Nguyên nhân là do cái nhãn “Sample Dialog Box Routin – Part 5” của ta quá dài. Cắt ngắn nó thành “SDBR – Part 5” và thử lại:)



Looks better! (Trông bây giờ rồi chứ?)

Looking good so far. We need to add the [SaveVars](#) function to save the selected items from the `radio_column` when the Okay button is pressed. *Look at the blue text in the Sample5.lsp program above.* (Ta hãy rặn thêm tí nữa nha. Ta cần thêm hàm `SaveVars` để lưu lại các khoản mục được chọn từ cột lựa chọn khi nút Okay bị nhấn. Xem dòng chữ màu xanh trong chương trình `Sample.lsp` ở trên.)

Let's steal the `saveVars` routine from the `radio_column` control on the "[Saving data from the dialog box](#)" page of this tutorial and modify it. I'll show the modifications in red. (Chôm vòng lưu biến `SaveVars` từ mục điều khiển cột lựa chọn trên trang “Lưu dữ liệu từ hộp thoại” của tài liệu này và sửa đổi lại. Tôi sẽ thể hiện những phần sửa đổi bằng màu đỏ.)

We can do this two different ways. We can check each `radio_button` to find out which one is on or we can check the entire column of `radio_buttons` by getting the value of the `radio_column`. (Ta có thể làm điều này bằng hai cách khác nhau. Ta sẽ kiểm tra từng nút lựa chọn để xác định nút nào được bật hoặc ta có thể kiểm tra cả cột các nút lựa chọn bằng cách lấy giá trị của cột lựa chọn.)

First method: Checking the `Radio_Column`: (Phương pháp 1: Kiểm tra Cột lựa chọn)

```
(defun saveVars()

  ;;--- Get the key of the choice made
  ;; [ returns "but1" "but2" "but3" or "but4" whichever is selected.]

  (setq myChoice (get_tile "mychoice"))

)
```

Second method: Checking each `Radio_Button`: (Phương pháp 2: Kiểm tra từng nút lựa chọn)

```
(defun saveVars()

;;;--- Get the value of each item
  (setq choice1(atoi(get_tile "but1"))) // 0 = not chosen 1 = chosen
  (setq choice2(atoi(get_tile "but2"))) // 0 = not chosen 1 = chosen
  (setq choice3(atoi(get_tile "but3"))) // 0 = not chosen 1 = chosen
  (setq choice4(atoi(get_tile "but4"))) // 0 = not chosen 1 = chosen

)
```

Wow! That was easy. So...Which one do we use? For this tutorial, let's use both. Why not? (Ồ, quá dễ nhĩ. Vậy bạn muốn sử dụng cách nào? Để học,ta sẽ chơi cả hai thử. Sao lại không nhĩ?)

```
(defun saveVars()

;;;--- Get the key of the choice made
;;;    [ returns "but1" "but2" "but3" or "but4" whichever is selected.]

(setq myChoice(get_tile "mychoice"))

;;;--- Get the value of each item
  (setq choice1(atoi(get_tile "but1"))) // 0 = not chosen 1 = chosen
  (setq choice2(atoi(get_tile "but2"))) // 0 = not chosen 1 = chosen
  (setq choice3(atoi(get_tile "but3"))) // 0 = not chosen 1 = chosen
  (setq choice4(atoi(get_tile "but4"))) // 0 = not chosen 1 = chosen

)
```

Add this to the original Sample5.lsp program and we should have something that looks like this: (Thêm điều đó vào chương trình Sample5.lsp và ta sẽ có chương trình như SAU:

```
(defun saveVars()

;;;--- Get the key of the choice made
;;;    [ returns "but1" "but2" "but3" or "but4" whichever is selected.]

(setq myChoice(get_tile "mychoice"))

;;;--- Get the value of each item
  (setq choice1(atoi(get_tile "but1"))) // 0 = not chosen 1 = chosen
  (setq choice2(atoi(get_tile "but2"))) // 0 = not chosen 1 = chosen
  (setq choice3(atoi(get_tile "but3"))) // 0 = not chosen 1 = chosen
  (setq choice4(atoi(get_tile "but4"))) // 0 = not chosen 1 = chosen

)

(defun C:SAMPLE5()

;;;--- Load the dcl file
```



```

(setq dcl_id (load_dialog "SAMPLE5.dcl"))

;;;--- Load the dialog definition if it is not already loaded
(if (not (new_dialog "SAMPLE5" dcl_id))
    (progn
      (alert "The SAMPLE5.DCL file could not be loaded!")
      (exit)
    )
)

;;;--- If an action event occurs, do this function
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the user pressed the Cancel button
(if(= ddiag 1)
  (princ "\n Sample5 cancelled!")
)

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn
    (princ "\n The user pressed Okay!")
  )
)

;;;--- Suppress the last echo for a clean exit
(princ)
)

```

Last item. We need to replace the line in the program: `(princ "\n The user pressed Okay!")` with something to display the selected item. (Cuối cùng, ta cần thay thế dòng `(princ "\n The user pressed Okay!")` bằng cái gì đó để hiển thị khoản mục được chọn.)

```

;;;--- If the user pressed the Okay button
(if(= ddiag 2)
  (progn

    ;;;;--- Inform the user of his selection using the radio_column data
    (princ "\n Using Radio_column data...You chose ")
    (cond
      ((= myChoice "but1") (princ "Apples!"))
      ((= myChoice "but2") (princ "Oranges!"))
      ((= myChoice "but3") (princ "Bananas!"))
      ((= myChoice "but4") (princ "Lemons!"))
    )
  )
)

```

```

;;;--- Inform the user of his selection using the radio_buttons data
(princ "\n Using Radio_buttons data...You chose ")
(cond
  ((= Choice1 1) (princ "Apples!"))
  ((= Choice2 1) (princ "Oranges!"))
  ((= Choice3 1) (princ "Bananas!"))
  ((= Choice4 1) (princ "Lemons!"))
)
)
)

```

Add the above to the autolisp file, save it and test it out. Everything working okay? (Thêm những điều trên vào tập tin Autolisp, lưu lại và chạy thử chương trình. Mọi việc ổn cả chứ?)



```

Command: (load "sample5")
C: SAMPLE5

Command: sample5

Using Radio_column data... You chose Oranges!
Using Radio_buttons data... You chose Oranges!

Command: |

```

When you get your program tested and everything is working, move the blue line above, [(defun C: SAMPLE5 ())] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done.

[Back](#)

Dialog Control Language - Part 6 (Phần thực hành 6 về DCL)

Part 6 - Text and Toggles (Phần 6 - Văn bản và các nút kích hoạt)

Let's build a working DCL file showing us exactly how to handle text and toggles. (Ta sẽ xây dựng một tập tin DCL hoạt động để chỉ ra chính xác cách điều khiển văn bản và các nút kích hoạt)

We will build a DCL file containing 4 toggles, one text, plus a Cancel button. The selected item will be displayed on the screen in the text control. (Ta sẽ tạo một tập tin DCL gồm bốn nút kích hoạt, một nút văn bản với một nút Cancel. Khoản mục được chọn sẽ được hiển thị trên màn hình ở nút điều khiển văn bản)

Layout thoughts: I will place the text control on the top of the box. Then I'll put the toggles in a column, (stacked on top of each other). Last, I'll put the Cancel button at the bottom of the dialog box. So...I'll need something like this:

Thiết kế hộp thoại: Ta sẽ đặt nút điều khiển văn bản trên cùng của hộp thoại. Sau đó đặt các nút kích hoạt trong một cột (các nút chồng lên nhau). Cuối cùng đặt nút Cancel ở dưới cùng của hộp thoại. Do vậy ta cần một thứ giống như sau:

```
: column {
  : column {
    : text {
      // Put code for text here (Đặt mã của nút văn bản vào đây)
    }
  }
  : boxed_column {
    : toggle {
      // Put code for toggle 1 here (Đặt mã của nút kích hoạt 1 vào đây)
    }
    : toggle {
      // Put code for toggle 2 here (Đặt mã của nút kích hoạt 2 vào đây)
    }
    : toggle {
      // Put code for toggle 3 here (Đặt mã của nút kích hoạt 3 vào đây)
    }
    : toggle {
      // Put code for toggle 4 here (Đặt mã của nút kích hoạt 4 vào đây)
    }
  }
  : boxed_row {
    : button {
      // Put code for the Cancel button here (Đặt mã của nút Cancel vào đây)
    }
  }
}
```

Let's copy in the code for the header and all of the controls above from the "[Controls](#)" section of this tutorial. I'll show the changes that needed to be made in red. Notice the key names and labels had to be changed. (Ta sẽ copy các mã của tiêu đề và tất cả các nút điều khiển trên từ phần “Các nút điều khiển” của tài liệu này. Tôi sẽ thể hiện những sự thay đổi cần thiết bằng màu đỏ. Chú ý tên các khoá và các nhãn đã được thay đổi.)

```
SAMPLE6 : dialog {
    label = "Sample Dialog Box Routine - Part 6";
    : column {
        : column {
            : text {
                key = "text1";
                value = "Nothing selected.";
            }
        }
    : boxed_column {
        label = "Choose your lucky charms:";
        : toggle {
            key = "tog1";
            label = "Hearts";
            value = "0";
        }
        : toggle {
            key = "tog2";
            label = "Moons";
            value = "0";
        }
        : toggle {
            key = "tog3";
            label = "Stars";
            value = "0";
        }
        : toggle {
            key = "tog4";
            label = "Clovers";
            value = "0";
        }
    }
    : boxed_row {
        : button {
            key = "cancel";
            label = "Cancel";
            is_default = true;
            is_cancel = true;
        }
    }
}
```

(Ở đây thiếu mất 1 Ký tự ngoặc móc đóng kết thúc hộp thoại dialog đang mở. Chắc cụ Jeff buồn ngủ...)

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE6.DCL *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. (Tiếp theo ta lấy một phiên bản của chương trình Autolisp mẫu và sửa đổi lại nó. Tất cả các mã mới được thể hiện màu đỏ)

```
(defun C:SAMPLE6()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE6.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE6" dcl_id))
    (progn
      (alert "The SAMPLE6.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- If the user pressed the Cancel button
  (if(= ddiag 1)
    (princ "\n Sample6 cancelled!")
  )

  ;;--- If the user pressed the Okay button
  (if(= ddiag 2)
    (progn
      (princ "\n The user pressed Okay!")
    )
  )

  ;;--- Suppress the last echo for a clean exit
  (princ)

)
```

Remove everything listed in orange above. We do not need an Okay button. Thus we do not need to check to see if the user pressed Okay or Cancel. We also do not need the **SaveVars** routine in this program. Remove the orange items so your program looks like the one below.

(Xoá đi tất cả các mã được thể hiện bằng màu cam. Ta không cần nút Okay. Do vậy cũng không cần kiểm soát xem liệu người sử dụng có nhấn nút Okay hay Cancel không. Ta cũng không cần tới hàm SaveVars trong chương trình này. Sau khi xoá các mã màu cam chương trình của chúng ta sẽ như sau:)

```
(defun C:SAMPLE6 ()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE6.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE6" dcl_id))
      (progn
        (alert "The SAMPLE6.DCL file could not be loaded!")
        (exit)
      )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "cancel" "(done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- Suppress the last echo for a clean exit
  (princ)

)
```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE6.LSP *Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.*

Let's load the program and see what the DCL file looks like. On the command line type this: (Ta hãy tải chương trình này và xem tập tin DCL giống như thế nào. Trên dòng lệnh nhập:)

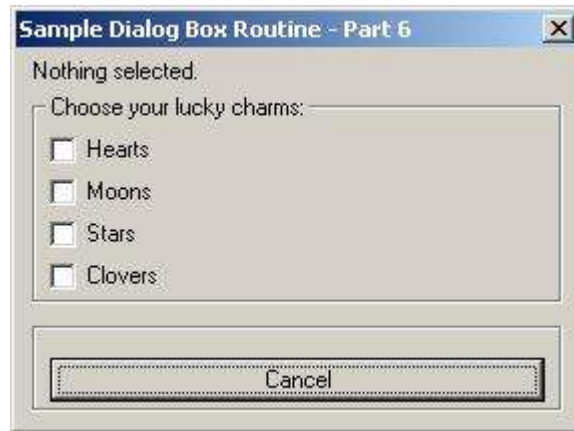
Command: (load "sample6") and press enter (và nhấn Enter)

You should see this (Bạn sẽ thấy như sau)

C:Sample6
Command:

Now type Sample6 and press enter. If everything went according to plan you should see this on your screen:

Bây giờ nhập Sample6 và nhấn Enter. Nếu mọi thứ tuân theo dự kiến bạn sẽ thấy trên màn hình như sau:



Doesn't look right. The dialog box is too wide. It's our label "Sample Dialog Box Routine - Part 6" that is causing the problem. I would shorten it but, my text control will need the room if everything is selected. I'll have to display "Hearts Moons Stars Clovers" all on one line. I'll leave it the way it is.

Trông chưa đẹp lắm, Hộp thoại quá rộng. Nguyên nhân là do nhãn “Sample Dialog Box Routine –Part 6” quá dài. Ta có thể cắt ngắn nó, nhưng nút điều khiển văn bản của chúng ta cần một không gian đủ rộng nếu một nút được chọn. Ta sẽ phải hiển thị toàn bộ “Hearts Moons Stars Clovers” trên một dòng. Vì vậy ta cứ để nó như thế.

Notice you can select and deselect items without a problem but the text control doesn't change. We need to add the `action_tiles` and function to accomplish this. (Chú ý là bạn có thể chọn hay không chọn các khoản mục tùy ý mà nút điều khiển văn bản không cần thay đổi. Ta cần thêm hàm các phần tử hành động `action_tiles` và hàm để hoàn thiện chúng)

First let's write a routine to check each toggle and build a string representing all of the selected items. (Trước hết ta sẽ viết một vòng kiểm soát từng nút kích hoạt và tạo một chuỗi hiển thị tất cả các mục được chọn.)

```
(defun chkToggle()

  (setq tog1(atoi(get_tile "tog1"))) // 0 = not chosen  1 = chosen
  (setq tog2(atoi(get_tile "tog2"))) // 0 = not chosen  1 = chosen
  (setq tog3(atoi(get_tile "tog3"))) // 0 = not chosen  1 = chosen
  (setq tog4(atoi(get_tile "tog4"))) // 0 = not chosen  1 = chosen

  (setq myStr "")
  (if(= tog1 1) (setq myStr(strcat myStr " Hearts")))
  (if(= tog2 1) (setq myStr(strcat myStr " Moons")))
  (if(= tog3 1) (setq myStr(strcat myStr " Stars")))
  (if(= tog4 1) (setq myStr(strcat myStr " Clovers")))

  ;;--- If nothing was selected...
  (if(= myStr "") (setq myStr "Nothing Selected!"))

  ;;--- Now set the text control to display the string
  (set_tile "text1" myStr)
```

)

Alrighty then. I used the `get_tile` function to get the value of each `toggle`. I used the `atoi` function to convert that data from a string to an integer. (*I could have left it a string and checked to see if `tog1` equalled "1" instead of the number 1.*) I set the variable `myStr` to an empty string and then appended all the checked `toggle labels` to it. I then changed the value of the text control by using the `set_tile` function.

Vậy là xong. Ta sử dụng hàm `get_tile` để lấy giá trị của từng nút kích hoạt. Sử dụng hàm `atoi` để đổi các dữ liệu từ chuỗi thành số nguyên. (*Ta có thể để nó ở dạng chuỗi và kiểm tra xem liệu `tog1` có phải là chuỗi "1" thay vì là số 1*). Tôi đặt biến `myStr` là một chuỗi rỗng "" và sau đó gán tất cả các nhãn của các nút kích hoạt được kiểm soát cho nó. Tôi thay đổi giá trị của nút điều khiển văn bản bằng cách sử dụng hàm `set_tile`.

Add this to the top of your autolisp program and save it. (Thêm điều này vào bên trên chương trình của bạn và lưu lại.)

The last step is to add the action calls to the AutoLisp program. We need one action call per toggle switch. Each action call should run the `chkToggle` function we just created. (Bước cuối cùng là thêm các lệnh gọi hành động vào chương trình Autolisp. Ta cần một lệnh gọi hành động cho mỗi một nút kích hoạt. Mỗi lệnh gọi hành động sẽ chạy hàm `chkToggle` mà ta vừa tạo ra.)

```
(action_tile "tog1" "(chkToggle)")
(action_tile "tog2" "(chkToggle)")
(action_tile "tog3" "(chkToggle)")
(action_tile "tog4" "(chkToggle)")
```

Let's add this to the AutoLisp program. I'll show the new `chkToggle` function and the action calls in red. It should look like this: (Thêm điều này vào chương trình. Tôi sẽ thể hiện hàm `chkToggle` mới và các lệnh gọi hành động bởi màu đỏ. Nó sẽ trông giống như sau:)

```
(defun chkToggle()

  (setq tog1(atoi(get_tile "tog1"))) // 0 = not chosen  1 = chosen
  (setq tog2(atoi(get_tile "tog2"))) // 0 = not chosen  1 = chosen
  (setq tog3(atoi(get_tile "tog3"))) // 0 = not chosen  1 = chosen
  (setq tog4(atoi(get_tile "tog4"))) // 0 = not chosen  1 = chosen

  (setq myStr "")
  (if(= tog1 1)(setq myStr(strcat myStr " Hearts")))
  (if(= tog2 1)(setq myStr(strcat myStr " Moons")))
  (if(= tog3 1)(setq myStr(strcat myStr " Stars")))
  (if(= tog4 1)(setq myStr(strcat myStr " Clovers")))

  ;;--- If nothing was selected...
  (if(= myStr "")(setq myStr "Nothing Selected!"))

  ;;--- Now set the text control to display the string
  (set_tile "text1" myStr)

)
```



```

(defun C:SAMPLE6()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE6.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE6" dcl_id) ) (exit))

  ;;--- If an action event occurs, do this function
  (action_tile "tog1" "(chkToggle)")
  (action_tile "tog2" "(chkToggle)")
  (action_tile "tog3" "(chkToggle)")
  (action_tile "tog4" "(chkToggle)")
  (action_tile "cancel" "(done_dialog)")

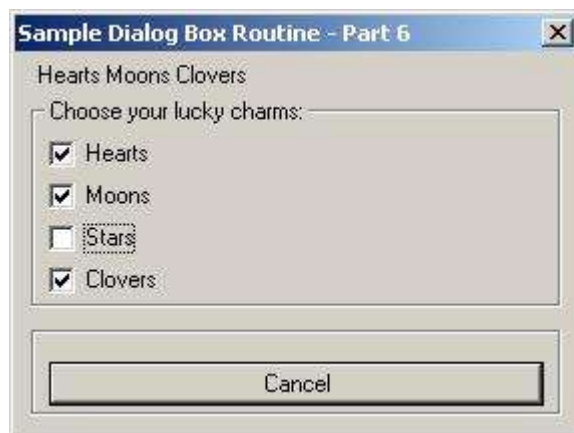
  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- Suppress the last echo for a clean exit
  (princ)
)

```

Save it and test it out. Everything working okay? (Lưu chương trình lại và chạy thử. Mọi việc tốt chứ?)



When you get your program tested and everything is working, move the blue line above, [(defun C:SAMPLE6()] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done.

[Back](#)

Dialog Control Language - Part 7 (Phần thực hành 7 về DCL)

Now it is time to take all of the parts and pieces and put them together. Let's build a working DCL file that will let us see most of the things we've learned and yet not get too deep into autolisp. (Bây giờ ta sẽ lấy toàn bộ các phần và các đoạn rồi ghép chúng lại với nhau. Ta sẽ xây dựng một tập tin DCL mà nó sẽ cho phép ta xem hầu hết những điều ta đã học nhưng còn chưa sâu về Autolisp)

Lets draw a polygon or circle, on a selected layer with the option to save the settings last used for defaults next time we run the program. If polygon is chosen, ask for the number of sides. So....we will need a list_box to hold all of the available layer names. We will need a radio_column to select a circle or polylgon. We will need an popup_list to hold the number of sides for the polygon. We will need a toggle to check the "Save Settings" option. And last, we will need an Okay and Cancel button.

Ta sẽ vẽ một đa giác hay một vòng tròn, trên một lớp đã chọn với tùy chọn để lưu các tham số đặt được sử dụng cuối cùng làm mặc định cho lần tiếp theo khi ta chạy chương trình. Nếu đa giác được chọn, nó sẽ yêu cầu số cạnh. Vì thế ta sẽ cần một hộp danh sách để lưu giữ tất cả các tên lớp có thể. Ta sẽ cần một cột lựa chọn để chọn đa giác hay vòng tròn. Ta cũng cần một danh sách thả xuống để lưu giữ số cạnh của đa giác. Ta cũng cần một nút kích hoạt để kiểm soát tùy chọn Save Setting. Cuối cùng ta cần một bộ nút Okay và Cancel.

Layout thoughts: The list will need to have several layers showing so the user won't have to scroll forever. So, it will probably be the tallest item on the dialog box. I'll put it in a column by itself. I'll try to fit the rest of the items in a column beside it. Then I'll put the Okay and Cancel buttons at the bottom in a row. So...I'll need something like this:

Thiết kế hộp thoại: Danh sách cần có vài lớp thể hiện để người sử dụng không phải cuộn mãi. Vì thế nó sẽ là khoản mục lớn nhất trong hộp thoại. Tôi đặt nó vào một cột riêng. Tôi sẽ cố gắng sắp xếp các khoản mục còn lại trong một cột bên cạnh nó. Sau đó tôi đặt các nút Okay và Cancel vào hàng dưới cùng của hộp thoại. Vì vậy tôi sẽ cần một thứ giống như sau:

```
: column {
  : row {
    : boxed_column {
      : radio_column {
        : radio_button {

          // Put code for radio button 1 here [ Circle ]
        }
        : radio_button {

          // Put code for radio button 2 here [ Polygon ]
        }
      }
    }
  }
  : popup_list {
```

```

        // Put code for popup_list here [ Number of Sides ]
    }
    : toggle {

        // Put code for toggle here [ Save Settings ]
    }
}
: row {
    : list_box {

        // Put code for list here [ Layer Names ]
    }
}
}
: row {
    : button {

        // Put code for button 1 here [ Okay ]
    }
    : button {

        // Put code for button 2 here [ Cancel ]
    }
}
}
}

```

Let's copy in the code for the header and all of the controls above. I'll show them in red. Notice the key names and labels had to be changed. (Ta sẽ copy tất cả mã cho tiêu đề và các nút điều khiển ở trên. Tôi sẽ thể hiện chúng bằng màu đỏ. Chú ý là các tên khoá và các nhãn đã được thay đổi)

```

SAMPLE7 : dialog {
    label = "Sample Dialog Box Routine - Part 7";
    : column {
        : row {
            : boxed_column {
                : radio_column {
                    key = "radios";
                    : radio_button {
                        label = "Draw Circle";
                        key = "drawcir";
                        value = "1";
                    }
                    : radio_button {
                        label = "Draw Polygon";
                        key = "drawpol";
                    }
                }
            }
        }
    }
}

```

```

        value = "0";
    }
}
: popup_list {
    key = "numsides";
    label = "Number of Sides";
    width = 25;
    fixed_width_font = true;
}
: toggle {
    key = "saveset";
    label = "Save settings";
}
}
: row {
    : list_box {
        label = "Select Layer";
        key = "layerList";
        height = 5;
        width = 15;
        multiple_select = false;
        fixed_width_font = true;
        value = "";
    }
}
: row {
    : button {
        key = "accept";
        label = " Okay ";
        is_default = true;
    }
    : button {
        key = "cancel";
        label = " Cancel ";
        is_default = false;
        is_cancel = true;
    }
}
}
}

```

Chú ý : Ở đây khóa của list_box tên “Select Layer” bị sai do khóa của list_box được xác định cả về kiểu chữ in và chữ thường .

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE.DCL Be sure to

change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.

Next we will get a copy of the AutoLisp model and revise it. All new code is shown in red. (Bây giờ ta lấy một phiên bản của chương trình Autolisp mẫu và sửa lại nó. Tất cả mã mới được thể hiện bởi màu đỏ.)

```
(defun C:SAMPLE7 ()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE7.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE7" dcl_id))
    (progn
      (alert "The SAMPLE7.DCL file could not be loaded!")
      (exit)
    )
  )

  ;;--- If an action event occurs, do this function
  (action_tile "cancel" "(setq ddiag 1) (done_dialog)")
  (action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

  ;;--- Display the dialog box
  (start_dialog)

  ;;--- Unload the dialog box
  (unload_dialog dcl_id)

  ;;--- If the cancel button was pressed - display message
  (if (= ddiag 1)
    (princ "\n \n ...SAMPLE7 Cancelled. \n ")
  )

  ;;--- If the "Okay" button was pressed
  (if (= ddiag 2)
    (princ "\n \n ...SAMPLE7 Complete!")
  )

  ;;--- Suppress the last echo for a clean exit
  (princ)

)
```

Right click and copy the above. Open NotePad and paste it. Save the file as SAMPLE7.LSP Be sure to change the "Save as Type" drop down box to "All Files" before saving it or it will put a ".txt" extension on the file name. Save this file somewhere in the AutoCAD search path.

Let's load the program and see what the DCL file looks like. On the command line type this:

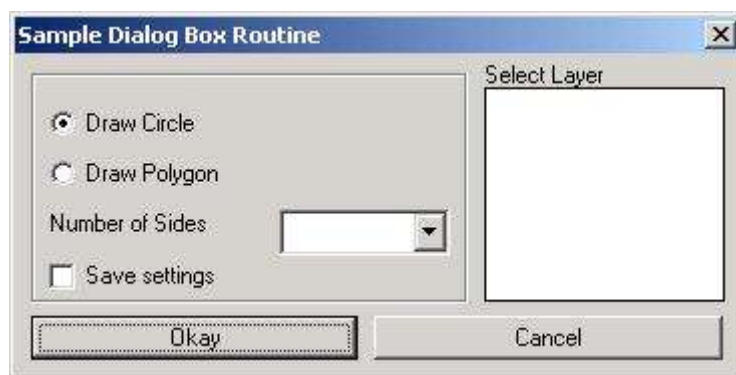
Command: (load "sample7") and press enter

You should see this

C:Sample7

Command:

Now type Sample and press enter. If everything went according to plan you should see this on your screen:
(Bây giờ nhập Sample7 và nhấn Enter. Nếu mọi thứ được thực hiện đúng bạn sẽ thấy như sau:)



Notice there are no items in either list. We haven't added that part yet. Also notice the Draw Circle is selected. We did that in the DCL file. We set the value of that radio_button to "1". We did not set the value of the toggle, so the default is unchecked. (Chú ý là không có khoản mục nào trong các hộp danh sách. Ta vẫn chưa thêm phần này vào. Và cũng lưu ý rằng nút lựa chọn "Draw Circle" được chọn. Điều này đã được làm trong tập tin DCL. Ta đã đặt giá trị của nút lựa chọn này là "1". Ta đã không đặt giá trị cho nút kích hoạt, vì thế giá trị mặc định là không kiểm soát.)

If you push the Cancel button the program will exit normally. If you press the Okay button an error will occur because we have not defined the saveVars routine. (Nếu bạn nhấn nút Cancel chương trình sẽ thoát bình thường. Nếu bạn nhấn nút Okay, một lỗi sẽ xảy ra do ta vẫn chưa xác định vòng lưu biến SaveVars.)

We have two things left to do. First we need to build the list for the popup_list control and the list_box control. Then we need to add the list to the dialog box. (Ta có hai điều phải làm. Trước hết bạn phải tạo các danh sách cho nút điều khiển danh sách thả xuống và nút điều khiển hộp danh sách. Sau đó ta phải thêm các danh sách này vào hộp thoại.)

Instead of building a function to get all of the layer names into a list, let's just build the list manually to keep things simple. Okay...you talked me into it. I'll do it both ways and you can use the one you want. I'm going to use the short version for this tutorial. (Thay vì xây dựng một hàm để lấy tất cả các tên lớp làm danh sách, ta sẽ tạo danh sách đó một cách thủ công để cho đơn giản. Rồi, bạn bảo tôi đây nhé. Tôi sẽ làm cả hai cách và bạn có thể sử dụng cách mà bạn muốn. Trong bài này tôi sẽ xài cách ngắn)

Long way: (Cách dài:)

```
;;;--- Set up a list to hold the layer names (lập một danh sách để lưu các tên lớp)
(setq layerList (list))

;;;--- Get the first layer name in the drawing (Lấy tên lớp đầu tiên trong bản vẽ)
(setq layr (tblnext "LAYER" T)) ; Chấn mớ đời cái cụ Jeff này, tự dưng thò ra cái thằng tblnext mà
chả thấy giới thiệu gì cả, chả biết nó con cái nhà ai mà lần, Thôi thì kệ thầy nó rồi làm quen sau vậy.

;;;--- Add the layer name to the list (Thêm tên lớp vào danh sách)
(setq layerList (append layerList (list (cdr (assoc 2 layr)))))

;;;--- Step through the layer table to get all layers (Thông qua bảng lớp để lấy tất cả các lớp)
(while (setq layr (tblnext "LAYER"))

  ;;;;--- Save each layer name in the list (Lưu mỗi tên lớp vào danh sách)
  (setq layerList (append layerList (list (cdr (assoc 2 layr)))))
)
```

Short Way: (Cách ngắn)

```
(setq layerList (list "0" "DIM" "HIDDEN" "STR" "TX" "WELD"))
```

We also need a list for the polygon's number of sides: (Ta cũng cần một danh sách số cạnh của đa giác:)

```
(setq numSides (list "4" "6" "8" "12" "16"))
```

We need to add these lines to our autolisp program. Add it just below the new_dialog call and above the action_tile statements. (Ta cần thêm các dòng này vào chương trình Autolisp. Thêm nó vào ngay dưới lệnh gọi hàm **new_dialog** và trước hàm thông báo phần tử hành động **action_tile**)

We now need to upload the list into the dialog box. So put these lines just below the lines you just added. (Bây giờ ta cần đưa danh sách vào trong hộp thoại. Vì thế , đặt các dòng này ngay dưới các dòng mà bạn vừa thêm)

```
;;;--- Add the layer names to the dialog box (Thêm các tên lớp vào hộp thoại)
(start_list "layerlist" 3)
(mapcar 'add_list layerList)
(end_list) ; Chú ý tên khoá là "layerlist" chứ không phải là "layerList" đâu nhé

;;;--- Add the number of sides to the dialog box (Thêm số cạnh đa giác vào hộp thoại)
(start_list "numsides" 3)
(mapcar 'add_list numSides)
(end_list)
```

When you are done it should look like this: (Khi bạn làm xong chương trình sẽ như sau:)

```
(defun C:SAMPLE7()
```

```

;;;--- Load the dcl file
(setq dcl_id (load_dialog "SAMPLE7.dcl"))

;;;--- Load the dialog definition if it is not already loaded
(if (not (new_dialog "SAMPLE7" dcl_id))
    (progn
      (alert "The SAMPLE7.DCL file could not be loaded!")
      (exit)
    )
)

(setq layerList(list "0" "DIM" "HIDDEN" "STR" "TX" "WELD"))

(setq numSides(list "4" "6" "8" "12" "16"))

;;;--- Add the layer names to the dialog box
(start_list "layerlist" 3)
(mapcar 'add_list layerList)
(end_list)

;;;--- Add the number of sides to the dialog box
(start_list "numsides" 3)
(mapcar 'add_list numSides)
(end_list)

;;;--- If an action event occurs, do this function
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the cancel button was pressed - display message
(if (= ddiag 1)
    (princ "\n \n ...SAMPLE7 Cancelled. \n ")
)

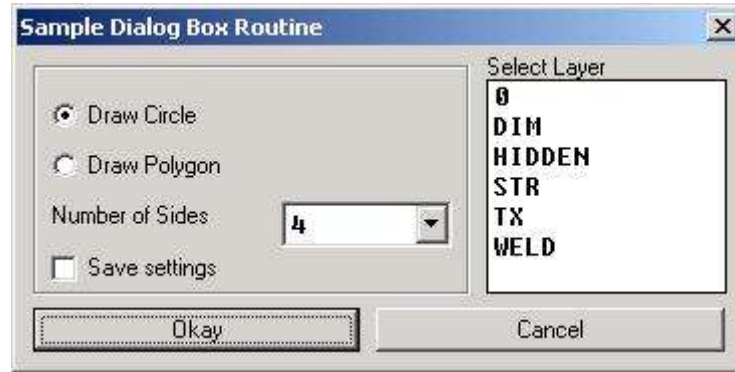
;;;--- If the "Okay" button was pressed
(if (= ddiag 2)
    (princ "\n \n ...SAMPLE7 Complete!")
)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

Save, Load, and Run. You should see this: (Lưu lại, tải chương trình và chạy. Bạn sẽ thấy như sau:)



Next, let's build the SaveVars routine. We will do this by starting with the saveVars routine in the AutoLisp Model, then copy and paste from the "Save Data from Dialog Box" section of this tutorial. We will then modify the names to match the key names in the DCL file. *I also changed a variable name to **numStr** to be used later to set the default settings. More on that later.* I'll show you the changes I made in red: (Tiếp theo ta sẽ xây dựng vòng lưu biến SaveVars. Ta sẽ thực hiện điều này bằng cách bắt đầu với vòng lưu biến trong chương trình Autolisp mẫu, sau đó copy và dán từ phần “Lưu dữ liệu từ hộp thoại” của tài liệu này. Sau đó ta sẽ sửa đổi các tên cho phù hợp tên khoá trong tập tin DCL. Ta cũng thay đổi tên biến thành numStr để sử dụng về sau khi đặt các giá trị mặc định. Những điều khác sẽ nói sau. Tôi sẽ thể hiện những sự thay đổi đó bằng màu đỏ.)

```
(defun saveVars()

  (setq radios (get_tile "radios"))

  ;;--- Get the number of sides selected from the list
  (setq numStr (get_tile "numsides"))
  (if (= numStr "")
    (setq numSides nil)
    (setq numSides (nth (atoi numStr) numSides)))
  )

  ;;--- See if the user wants to save the settings
  (setq saveSet (atoi (get_tile "saveSet")))

  ;;--- Get the selected item from the layer list
  (setq sStr (get_tile "layerlist"))

  ;;--- If the index of the selected item is not "" then something was selected
  (if (/= sStr "")
    (progn

      ;;--- Something is selected, so convert from string to integer
      (setq sIndex (atoi sStr))

      ;;--- And get the selected item from the list
      (setq layerName (nth sIndex layerList))
    )

    ;;--- Else, nothing is selected
    (progn
```

```

;;;--- Set the index number to -1
(setq sIndex -1)

;;;--- And set the name of the selected item to nil
(setq layerName nil)
)
)
)

```

Note: Since we did not specify a default value for the list_box, layerName will be set to nil if the user does not select a layer from the list before hitting the Okay button. (**Chú ý:** Do chúng ta không mô tả giá trị mặc định của hộp danh sách, tên lớp sẽ được đặt về nil khi người sử dụng không chọn một lớp nào từ danh sách lớp trước khi kích hoạt nút Okay.)

Save, Load, and Run. Check the values after pressing the Okay button by typing an exclamation point and then the variable name on the command line. Examples: To get the layer name type !layerName and press enter. To get the value of the toggle type !saveSet and press enter. Everything working okay? Alrighty then...let's move on. (Lưu lại, tải và chạy thử. Kiểm tra các giá trị sau khi nhấn nút Okay bằng cách nhập một dấu chấm than rồi đến tên biến trên dòng lệnh. Ví dụ: Để lấy giá trị biến tên lớp, nhập !layerName rồi nhấn Enter. Để lấy giá trị của nút kích hoạt, nhập !saveSet rồi nhấn Enter. Mọi thứ chạy tốt chứ? Rồi, tiếp tục...)

The only work left on the dialog box is to disable the "Number of Sides" popup_list when a circle is selected. Let's add two action_tiles on both of the radio_buttons. (Việc duy nhất còn lại với hộp thoại là tắt danh sách thả xuống "Number of sides" khi chọn vẽ vòng tròn. Ta sẽ thêm hai hàm phần tử hành động vào cả hai nút lựa chọn.)

```

;;;--- If an action event occurs, do this function
(action_tile "drawcir" "(toggleRadio 1)")
(action_tile "drawpol" "(toggleRadio 2)")

```

This will send a parameter of 1 to the toggleRadio function if the Circle is selected. It will send a parameter of 2 to the toggleRadio function if the Polygon is selected. What the hell is a toggleRadio function? (Điều này sẽ gửi tham số 1 vào hàm lựa chọn kích hoạt nếu việc vẽ vòng tròn được chọn. Gửi tham số 2 vào hàm lựa chọn kích hoạt nếu việc vẽ Đa giác được chọn. Vậy chứ hàm lựa chọn kích hoạt là thằng chó nào vậy?)

We have to create it and put it in our AutoLisp program. Like this: (Ta sẽ tạo nó và đặt nó vào chương trình Autolisp. Nó như sau:

```

(defun toggleRadio(a)

  ;if circle is selected
  (if (= a 1)
    (mode_tile "numsides" 1)    ;disable

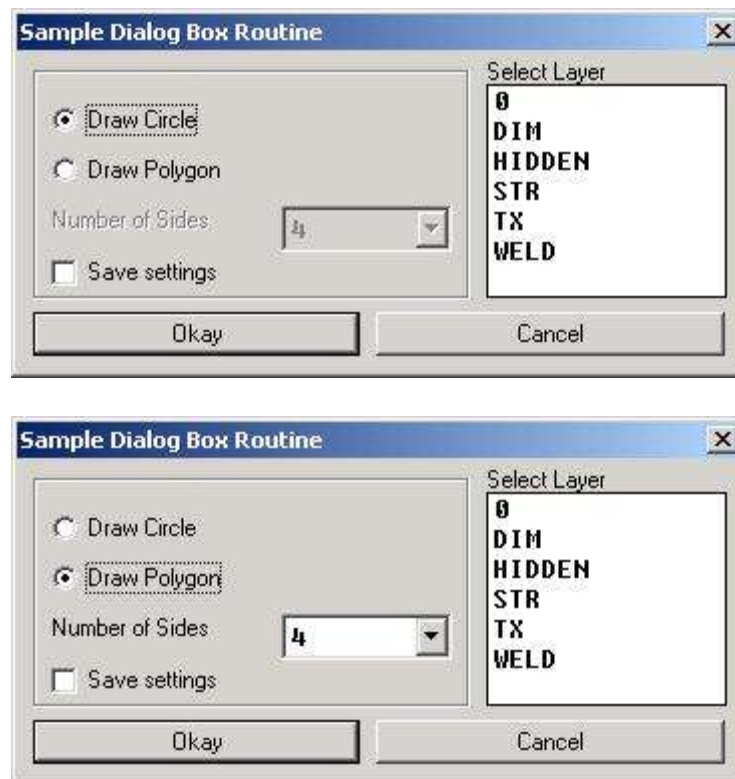
    ;else
    (mode_tile "numsides" 0)    ;enable
  )
)

```

Since our default for the radio buttons is to draw a circle, the numSides popup_list should be disabled before the dialog box starts. So just before the action_tile statements we need to add this line: (Do mặc định cho các nút lựa chọn của chúng ta là vẽ vòng tròn, danh sách thả xuống “number of sides” sẽ biến mất trước khi hộp thoại bắt đầu. Vì thế, ngay trước các hàm thông báo phần tử hành động ta phải thêm các dòng sau:)

```
(mode_tile "numsides" 1)
```

Copy, Save, and Run. You should see this: (Copy, lưu lại và chạy thử. Bạn sẽ thấy như sau:)



The numSides popup_list is disabled when the circle is selected. Enabled when polygon is selected. Cool. (Danh sách thả xuống “Number of sides” không hoạt động khi việc vẽ vòng tròn được chọn và hoạt động khi vẽ đa giác được chọn. Tuyệt cú mèo.)

Here is the AutoLisp program after the lines above were added: (Đây là chương trình Autolisp sau khi đã thêm các dòng trên.)

```
(defun saveVars()
  (setq radios(get_tile "radios"))

  ;;--- Get the number of sides selected from the list
  (setq sStr(get_tile "numsides"))
  (if(= sStr "")
    (setq numSides nil)
    (setq numSides(nth (atoi sStr) numSides)))
)
```

```

;;;--- See if the user wants to save the settings
(setq saveSet(atoi(get_tile "saveset")))

;;;--- Get the selected item from the layer list
(setq sStr(get_tile "layerlist"))

;;;--- If the index of the selected item is not "" then something was selected
(if(/= sStr "")
  (progn
    ;;;;--- Something is selected, so convert from string to integer
    (setq sIndex(atoi sStr))

    ;;;;--- And get the selected item from the list
    (setq layerName(nth sIndex layerList))
  )

  ;;;;--- Else, nothing is selected
  (progn
    ;;;;--- Set the index number to -1
    (setq sIndex -1)

    ;;;;--- And set the name of the selected item to nil
    (setq layerName nil)
  )
)
)

(defun toggleRadio(a)
  ;if circle is selected
  (if(= a 1)
    (mode_tile "numsides" 1) ;disable
    ;else
    (mode_tile "numsides" 0) ;enable
  )
)

(defun C:SAMPLE7()

  ;;;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE7.dcl"))

  ;;;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE7" dcl_id))
    (progn
      (alert "The SAMPLE7.DCL file could not be loaded!")
      (exit)
    )
  )

  (setq layerList(list "0" "DIM" "HIDDEN" "STR" "TX" "WELD"))
  (setq numSides(list "4" "6" "8" "12" "16"))

  ;;;;--- Add the layer names to the dialog box
  (start_list "layerlist" 3)

```

```

(mapcar 'add_list layerList)
(end_list)

;;;--- Add the number of sides to the dialog box
(start_list "numsides" 3)
(mapcar 'add_list numSides)
(end_list)

(mode_tile "numsides" 1)

;;;--- If an action event occurs, do this function
(action_tile "drawcir" "(toggleRadio 1)")
(action_tile "drawpol" "(toggleRadio 2)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the cancel button was pressed - display message
(if (= ddiag 1)
  (princ "\n \n ...SAMPLE7 Cancelled. \n ")
)

;;;--- If the "Okay" button was pressed
(if (= ddiag 2)
  (princ "\n \n ...SAMPLE7 Complete!")
)

;;;--- Suppress the last echo for a clean exit
(princ)
)

```

Now the only thing remaining is to do something when the user presses the okay button. We have all of the dialog box data stored in variable names.... (Bây giờ điều duy nhất còn lại là phải làm điều gì đó khi người sử dụng nhấn nút Okay. Ta có tất cả các dữ liệu của hộp thoại được lưu dưới các tên biến.)

Variable Name	DCL Control Item	Action Key	Type of Data Stored
radios	Radio_Column	"radios"	String [Action Key Name]
numSides	Popup_List	"numsides"	Integer [Number of sides]
saveSet	Toggle	"saveSet"	Integer [0 or 1]
layerName	List	"layerlist"	String [Name of Layer]

So now all we have to do is write the AutoLisp code inside the "Okay button was pressed" IF statement.

(Where the blue line is above.) (Ví thế bây giờ toàn bộ điều ta phải làm là viết mã Autolisp bên trong hàm thông báo If “Okay button was pressed”. (Nơi có dòng chữ màu xanh ở trên))

Now we will replace the blue line above: `(princ "\n \n ...SAMPLE Complete!")` with new code (Bây giờ ta sẽ thay thế dòng chữ màu xanh (`princ “\n \n ...Sample7 Complete!”`) bằng các mã mới.)

(Cần chú ý bổ sung thông báo có nhiều hành động được thực hiện trong hàm if)

First, let's go ahead and change the layer. (Trước hết thay đổi các lớp ở phần đầu chương trình)

```
(setq oldLay(getvar "clayer"))
(setvar "clayer" layerName)
```

That was easy. Now, Let's draw the selected entity (Điều đó quá dễ, bây giờ ta vẽ đối tượng được chọn)

```
(if(= radios "drawcir")
  (progn
    (setq pt(getpoint "\n Center point: "))
    (command "circle" pt pause)
  )
  ;;--- Else draw a polygon
  (progn
    (setq pt(getpoint "\n Center Point: "))
    (command "polygon" numSides pt "C" pause)
  )
)
```

Add the above to your program and save it. Test it out. Everything working okay? (Thêm tất cả các điều trên vào chương trình của chúng ta và lưu nó lại. Chạy thử. Mọi thứ làm việc tốt cả chứ?)

And finally all we have left is the code for the default settings. Since everything in the dialog box is a string, why don't we save all of the data as strings. Lucky for us, AutoDesk included 15 setvars for us to store data in. `USERR1` thru `USERR5` is used to store real numbers. `USERI1` thru `USERI5` is used to store integers. `USERS1` thru `USERS5` are used to store strings. We will use the system variables `USERS1` thru `USERS4` to save our data since we are saving the data as strings. All of these setvars are stored inside the drawing file. They will be there everytime you open your drawing. How convenient! So let's get to it.

Và cuối cùng ta còn phải làm là các mã cho việc đặt mặc định. Do mọi thứ trong hộp thoại đều là dạng chuỗi, vậy tại sao ta lại không lưu các dữ liệu dưới dạng chuỗi. Rất may mắn là Autodesk bao gồm 15 hàm đặt biến **setvars** để chúng ta có thể lưu các dữ liệu lại. Từ biến **USERR1** đến **USERR5** được sử dụng để lưu các số thực. Từ biến **USERI1** đến **USERI5** được sử dụng để lưu các số nguyên. Từ biến **USERS1** đến **USERS5** được dùng để lưu các chuỗi. Ta sẽ sử dụng các biến hệ thống **USERS1** đến **USERS4** để lưu dữ liệu của chúng ta do ta đang lưu dữ liệu dưới dạng chuỗi. Tất cả việc đặt biến **setvars** này được lưu bên trong tập tin bản vẽ. Chúng luôn ở đó mỗi khi bạn mở bản vẽ. Quá tiện lợi rồi! Vì thế ta sẽ lấy nó ra.

The first variable to store is `RADIOS` and it is already a string. So.... (Biến đầu tiên cần lưu là `RADIOS` và nó đã là một chuỗi. Cho nên)

```
(setvar "USERS1" radios)
```

The second variable to store is NUMSIDES and it is an integer representing the number of sides the polygon should have. I want to store the index of the selected item in the list not the actual number of sides. We saved the index as a variable named NUMSTR when we ran the saveVars routine. So... (Biến thứ hai phải lưu là NUMBERSIDES và nó là một số nguyên đại diện cho số cạnh của đa giác cần vẽ. Tôi muốn lưu số thứ tự của khoản mục được chọn trong danh sách đó chứ không phải là số các cạnh thật sự của đa giác. Ta sẽ lưu số thứ tự đó như một biến mang tên NUMSTR khi chúng ta chạy vòng lưu biến saveVars. Vì thế)

```
(setvar "USERS2" numStr)
```

The third variable to save is SAVESET. It is stored as an integer. We will have to convert it to a string to save it. (Biến thứ ba phải lưu là SAVESET. Nó được lưu như một số nguyên. Ta sẽ đổi nó thành một chuỗi để lưu nó lại.)

```
(setvar "USERS3" (itoa saveSet))
```

The fourth and final variable to save is LAYERNAME. It is stored as a string. I want to save the index of the selected item. We saved that earlier with the saveVars routine in a variable named sSTR. (Thứ tư và cũng là biến cuối cùng phải lưu là LAYERNAME. Nó được lưu ở dạng chuỗi. Tôi muốn lưu số thứ tự của khoản mục được chọn. Ta đã lưu điều này từ trước nhờ vòng lưu biến SaveVars khi đặt tên biến sSTR.)

```
(setvar "USERS4" sSTR)
```

The last thing we have to do is check for the default settings and load them if they exist. (Điều cuối cùng ta phải làm là kiểm tra các giá trị mặc định và tải nó nếu như nó tồn tại.)

```
(defun saveVars ()  
  
  (setq radios (get_tile "radios"))  
  
  ;;--- Get the number of sides selected from the list  
  (setq numStr (get_tile "numsides"))  
  (if (= numStr "")  
      (setq numSides nil)  
      (setq numSides (nth (atoi numStr) numSides)))  
  )  
  
  ;;--- See if the user wants to save the settings  
  (setq saveSet (atoi (get_tile "saveSet")))  
  
  ;;--- Get the selected item from the layer list  
  (setq sStr (get_tile "layerlist"))  
  
  ;;--- If the index of the selected item is not "" then something was selected  
  (if (/= sStr "")  
      (progn  
  
        ;;--- Something is selected, so convert from string to integer  
        (setq sIndex (atoi sStr))  

```

```

    ;;--- And get the selected item from the list
    (setq layerName (nth sIndex layerList))
  )

  ;;--- Else, nothing is selected
  (progn

    ;;--- Set the index number to -1
    (setq sIndex -1)

    ;;--- And set the name of the selected item to nil
    (setq layerName nil)
  )
)
)

(defun toggleRadio(a)
  ;if circle is selected
  (if (= a 1)
    (mode_tile "numsides" 1) ;disable
    ;else
    (mode_tile "numsides" 0) ;enable
  )
)

(defun C:SAMPLE7()

  ;;--- Load the dcl file
  (setq dcl_id (load_dialog "SAMPLE7.dcl"))

  ;;--- Load the dialog definition if it is not already loaded
  (if (not (new_dialog "SAMPLE7" dcl_id))
    (progn
      (alert "The SAMPLE7.DCL file could not be loaded!")
      (exit)
    )
  )

  ;; Build the list for the dialog box
  (setq layerList (list "0" "DIM" "HIDDEN" "STR" "TX" "WELD"))
  (setq numSides (list "4" "6" "8" "12" "16"))

  ;;--- Add the layer names to the dialog box
  (start_list "layerlist" 3)
  (mapcar 'add_list layerList)
  (end_list)

  ;;--- Add the number of sides to the dialog box
  (start_list "numsides" 3)
  (mapcar 'add_list numSides)
  (end_list)

  ;;--- Add the code here to check for defaults
  (if (/= (getvar "USERS1") "")

```



```

(progn
  (setq radios (getvar "users1"))
  (setq numStr (getvar "users2"))
  (setq saveSet (getvar "users3"))
  (setq layerIndex (getvar "users4"))
  (set_tile "radios" radios)
  (set_tile "numsides" numStr)
  (set_tile "saveSet" saveSet)
  (set_tile "layerlist" layerIndex)
)
)

;;;--- Only disable the numSides popup_list if a circle is being drawn
(if(= radios "drawcir")
  (mode_tile "numsides" 1)
)

;;;--- If an action event occurs, do this function
(action_tile "drawcir" "(toggleRadio 1)")
(action_tile "drawpol" "(toggleRadio 2)")
(action_tile "cancel" "(setq ddiag 1) (done_dialog)")
(action_tile "accept" "(setq ddiag 2) (saveVars) (done_dialog)")

;;;--- Display the dialog box
(start_dialog)

;;;--- Unload the dialog box
(unload_dialog dcl_id)

;;;--- If the cancel button was pressed - display message
(if (= ddiag 1)
  (princ "\n \n ...SAMPLE7 Cancelled. \n ")
)

;;;--- If the "Okay" button was pressed
(if (= ddiag 2)
  (progn

    ;;;;--- Save the old layer and reset to new layer
    (setq oldLay (getvar "clayer"))
    (setvar "clayer" layerName)

    ;;;;--- See what needs to be drawn
    (if(= radios "drawcir")
      (progn
        (setq pt (getpoint "\n Center point: "))
        (command "circle" pt pause)
      )

      ;;;;--- Else draw a polygon
      (progn
        (setq pt (getpoint "\n Center Point: "))
        (command "polygon" numSides pt "C" pause)
      )
    )
  )
)

```

```

;;;--- See if we need to save the settings
(if(= saveSet 1)
  (progn

    ;;;--- Add code here to save the settings as defaults
    (setvar "USERS1" radios)
    (setvar "USERS2" numStr)
    (setvar "USERS3" (itoa saveSet))
    (setvar "USERS4" sSTR)

  )
)

)

;;;--- Suppress the last echo for a clean exit
(princ)

)

```

When you get your program tested and everything is working, move the blue line above, [[\(defun C:SAMPLE7 \(\)](#)] all the way to the top of the file. This will make all of your variables local and will reset them all to nil when the program ends.

That's it. We're done.

[Back](#)

[AutoLisp Tutorial Home](#)

[AutoLisp Home](#)

[Home](#)

All questions/complaints/suggestions should be sent to JefferyPSanders.com

Last Updated February 26th, 2005

Copyright 2002-2007 JefferyPSanders.com. All rights reserved.

Programs

ADDLEN - Total the lengths of selected arcs, circles, ellipses, lines, lwpolylines, polylines, and/or splines.

AlgaCAD - Write AutoLisp programs to fill in standard drawings.

ALIAS - Shortcuts / Single Letter Commands on the fly.

ATTINC - Increment values of attributes.

ATTMAP - Map Attributes from drawing to drawing.

Autolisp Games - TicTacToe, HangMan, Mines, BattleShip, and Mummy.

Axcel - VB Application for Excel.

AxcelPts - VB Application for Excel.

BatchLisp - Run a selected autolisp program on multiple drawings.

BLKOUT - Extract nested blocks into individual files.

BLKTREE - Display blocks and nested blocks in a tree format.

BLOCKS - Count blocks filtered by name, attribute tag, or attribute value.

CAD2FILE - Get entity data from AutoCAD to a file or into Excel.

CASE Change text to upper case, lower case, or upper case the first letter of each word.

CHAIN Draw a chain like drawing a line. From point to point to point.

CLAY Create layer(s) from a master list.

CNTBLK Count blocks (Attribute Filter Also).

CSVTable Draw a table in autocad based on data from a selected CSV file.

CYL - Create hollow 3D circular, rectangular, or polygonal shapes.

Free AutoCAD Tools

GA Get the area and perimeter by selecting the interior of any area.

GETCELLS Function to get the values of a cell or cells from Excel into a list..

IMPORT XYZ Import coordinates from practically any type of file. Draw nodes, draw circles, draw lines, insert blocks, insert blocks and update attributes with the coordinates

ISODIM Create non-associative isometric dimensions.

LIBRARY Create your own slide library automatically.

LoadLSP AutoLisp loader. View all of your autolisp programs along with a description.

Match - Draw a matchline.

Num.lsp - Program to create charts automatically.

OverLapF - Remove Overlapping Lines.

REMF Remove Multiple Layer Filters.

RepAtt Replace the value of all matching attributes. Includes a value filter.

Rolling_ball Thanks to Guenther Bittner, the program is complete. Find the mid-boundary between two polylines. Verion 4+ now available.

SafeX - Explode a block containing attributes and replace the attributes with a text entity containing the original attribute value.

SBox Create a Shadow Box

SelectPlotter - Select a plotter/printer from a list box.

Simplex - Hatchable Balloon Font.

Words - Count words inside an autolisp drawing using filets. **Updated!**

XL Get data from Excel into AutoCAD using Visual Lisp. **Version 3.1**

XMINsert Explode a MINsert object.

Cogenerator

Make_LSP - AutoLisp Cogenerator. (AutoLisp program to write AutoLisp programs.)

Make_LSP Help and Tutorial

Misc. AutoLisp Files

(Some of these programs were created with the Make_LSP program.)

AngX - Rotates entities by picking a line with the correct angle.

Au - Prefixes selected text with "% %u " and suffixes with " ".

BrkLine - Draws a break line.

Case - Reverses the case of text. (Upper to Lower and Lower to Upper.)

ChTxHt - Changes the height of selected text.

ChTxLy - Changes the Layer of selected text.

ChTxRt - Changes the Rotation of selected text.

ChTxSt - Changes the Style of selected text.

Cloud - Creates a revision cloud.

CPYXREF - Copies entities from an XREF or Block.

CrText - Circular Text

DrLim - Draw a line representing the limits of the drawing. (LimMax to LimMin)

EBT - Erase Blank Text. Removes all text with a value of " " or ""

ErasNode - Erase all nodes in the drawing.

FL - Select a line and this program will put the feet(decimal) and the angle(deg min sec) centered on top of the line.

FZ and UFZ FZ - FreeZes all layers including the current layer. UFZ - UnFreeZes all layers.

Ma - Match text by selecting the items.

MixTxt - Rotates words through a text entity. Why? I don't know.

NodeSert - Insert blocks on every node in a drawing.

[NumberX](#) - This programs makes numbers out of vertical text starting with number 1 being the highest vertical text location and working downward.

[PipeLay](#) - This program creates a layout drawing of a pipe to pipe connection to help the shop fabricate this unusual connection. The shop wraps the paper around the pipe and marks the cut lines. Click the link to find out more.

[PreTx](#) - Prefix all text with a string.

[RRR](#) - Text search and replace routine.

[Sag](#) - Draws the sag for conductor lines in 3D.

[Simplex](#) - Draws "ballooned" simplex characters with polylines.

[Slot](#) - Draws a solid slotted hole.

[SpcText](#) - Puts spaces between each character in a text string.

[srtAN](#) - Function to sort alpha-numerically. See inside file for instructions.

[StText](#) STX - Stretches a text entity. USTX - Un-stretches a text entity.

[SwapEnt](#) - Swaps two entities.

[SwapTxt](#) - Quickly swaps two text entities.

[TextIn](#) - Prints a text file in AutoCAD

[TextOut](#) - Prints AutoCAD text to a file.

[TotalTx](#) - Totals numeric text entities.

More Coming soon.....

I've got several to add as soon as I find the time.

Request a program Want to suggest a program?

It's easy and cost you absolutely nothing. Click this-----> [I've got an idea!](#)

Want to see some **EXAMPLE** programs that were created through visitors ideas? [Click Here!](#)

Also be sure to see the **NUM.lsp** program for creating charts. This program was suggested by a visitor. I created the program for free and give everyone access to it.

Help Me!!

Need help with a program?

It's easy and cost you absolutely nothing. Click this-----> [Help!](#)