初級 AutoLISP 講座 1

取りあえず、動かしてみようよ

あなたのシステムで LISP は動く？

まず、自分の環境で LISP が動くのかを確認してみましょう。やり方はいたって簡単。まず AutoCAD が立ち上がった状態で、コマンドプロンプトから次のように入力してみましょう。

```
コマンド ： (+ 1 1)
2
```

ちゃんと下に 2 が表示されましたか？表示されていれば OK です。あなたのシステムで AutoLISP は正常に動いています。

- 入力例に示されている コマンド：の部分は AutoCAD が表示してくれるので、実際に入力するのは (+ 1 1) の部分です。漢字入力モードにも気を付けましょう、半角文字でないと AutoCAD が LISP の文だと解釈してくれません、漢字入力モードも OFF にしておきましょう。(+ 1 1) が入力し終わったら、リターンキーを押しましょうね。そうしないと 2 が表示されませんよ。

いまやったのはいったい何？

足し算をやっているのが解りましたか？でも書き方がなんか変ですよね。普通は 1 + 1 って書くでしょ。でも、LISP ってこういうものなんです。たとえば、あなたが英語を勉強したときに「日本語と構文が違う」って文句いいました？言わなかったでしょ(^^) だから、「そういう言葉なんだな」くらいに思ってればいいんです。

この文をもう少し、じっくり眺めてみましょう

この文は、開いた括弧 (で始まって 閉じた括弧) で終わっています。この構造は LISP では非常に重要です（試験に出しますよぉ）「開いた括弧は必ず閉じる」と念仏の様に心の中で三回唱えましょう。小さなプログラムを見ている段階では気になりませんが、少し大きくなってくると、足の指まで使って括弧の数を数えなければならなくなります。覚悟しておきましょう。

次に + ですが、これが LISP の関数です。関数というと算数の嫌いな人が逃げ出しちゃうんですけど、算数で出てくる関数とはちょっと違います。逃げないで下さいね。

「関数とは何か仕事をしてくれる人」と覚えておきましょう。人（関数）によってしてもらえる仕事はさまざまです。自分でプログラムを組む様になると、少しずつ、顔（してもらえる仕事）と、名前（関数名）とが一致してきますので、焦らずにじっくりいきましょう。

最後に 1 1 の部分ですが、ここは引数と呼ばれています。この例でいくと「これ（1）とこれ（1）を」足し算してね、の部分ですね。つまり引数の値を変えてやればいろんな足し算が出来るってことです。

- + 1 1 の部分ですが、全てスペース文字で区切られています。（スペース文字をわかりやすくするために○ でスペースを表現してみましょう） +1○1 や +○11 ではダメで、+○1○1 としなければなりません。だって、続けてかいたら 1 なんだか 11 なんだかわかんないでしょ (^^)

さて、結果はいかに

もう、解っていると思いますが。最初に入力したのは、「1 と 1 を足し算した答えを教えてね」です。結果は次の行に表示されていますね。そう、2 です。（当たり前すぎましたか？）こうやって答えが帰ってくることを「値が返る」と言います。よく使う言葉なので、覚えておきましょう。

代入って、いったい．．

代入してみましょ

なんでもやってみることって大事ですよね、今回も取り敢えずやってみましょう。

```
コマンド ： (setq var 12)
12
```

ちゃんと、12 という値が返ってきましたか？返ってきていれば、成功です。失敗した場合は、スペースの位置や漢字入力モードが OFF になっているかを確認しましょう。もしかして、(setq var 12) の後にリターンキーを押すのを忘れたりはしてないですよね。

ちょっと、前回と似てませんか？

前回と同じで、括弧でくくられた中にいてありますよね。勘の良い方はお気づきでしょう、前回の + に相当する部分が、今回の setq です。今回のお話は、この setq という関数のお話です。

では、次の var って何でしょう

このあたりから難しくなりますよ、覚悟して聞いて下さいね。この var は「変数」と呼ばれるもので、いろんな値（数だけじゃなんです）を書いておける便利な場所です。場所が嫌なら、箱を想像してもらっても結構です。箱はたくさん用意されていて、いつでも必要なときに使うことが出来ます。ただ、あんまりにもたくさん用意されているので、ちゃんと名前を付けておかないと、あとで見つからなくなっちゃいます。そこで、今回は var という名前を付けてみました。名前の由来は variable の略です（ばーちゃんと呼ん で下さいね、かわいい名前でしょ (^^)）

●名前をつけるときにもお約束があります。 ( ) . ' " ; の 6 つの文字は、名前に使えない文字です。これを使うとお化けが出るんだそうです。恐いものが大好きな人は、是非一度お試し下さい。その他、数字だけの変数名もダメですよ、数だか変数だかわかんなくなっちゃいますもんね。まだあるんですよ、PAUSE と PI と T も使わないようにしましょうね。今までの経験からすると、数字で始まる変数名もお行儀がよいとはいえません。皆さんは、礼儀正しいプログラムを書くようにしましょう。そうそう、まだあった (^^; 小文字と大文字は区別されませんので、PAUSE と Pause は同じ名前になります、気を付けましょう。

では、何をやったんでしょう

「var という名前を付けた変数に、12 という値を入れておいてね」です。変数に値を入れることを「代入」といいます。解りました？結果は、12 という値が返ってきてましたね。えっ、これだけじゃ面白くない？ふふん、そうくると思った。いま書いた 12 って本当に入っているのか、見てみたいでしょ。

ちょっと見てみよっと

※ AutoCAD Release 版以外の擬似 AutoLISP 環境では！での変数確認はできないかもしれません。

では、変数に何が書いてあるのか確認してみましょう。確認の方法は

```
コマンド ： !var
12
```

ちゃんと、12 って出たでしょ。この方法は応用がきいて、こんな使い方もできるんですよ。

```
コマンド ： CIRCLE
．．．中略．．．
D=直径/<半径> :!var
```

これで、半径 12 の円が描けましたか？やってみて下さいね。

●今度は！と var の間にスペース入れちゃダメですよ。ややこしいけどこういうものなんだもん。私に文句いわないのぉ。

最後にお掃除しましょう

いま使った変数 var に入っている 12 は、AutoCAD を終了するまで消えません。いくらたくさん用意してある箱でも、使えばいつかはなくなります。次の変数が使える様に、きれいにお掃除しておきましょう。お掃除の方法は

```
コマンド : (setq var nil)
nil
```

nil っていうのは、「なんにも無いの」と思って下さい。「var っていう変数は、もうなんにも無いの」できれいにお掃除完了です。

聞かれれば、答えます

シャイな AutoCAD

AutoCAD はとってもシャイな奴で、こちらが話かけないと何も答えてくれません。（恥ずかしがりやなところなんて、まるで私みたいですねぇ）しかも、彼に解りやすい言葉で話し掛けないと無視されちゃいます。答えも、彼の言葉で返ってきますので、理解してあげましょうね。今回は、「○○は○○です」がたくさん登場しますが。解らなければ、○○の中身について深く考えるのは止めましょう。マニュアルの引き方と、「こんな風な話し方をする奴なんだな」というところを理解してあげて下さい。

予備知識から

今回は、実践の前に予備知識からです。どうしてもこれを説明しないといけないもので．．．(^^;
前回までに出てきた「値」は「数」でしたよね。1 と 1 を足し算したり、var に 12 を代入して円を書いたりしました。でも、AutoLISP で扱える値は数だけではありません、他にもたくさんあるんです。今回は「文字列」という値が登場します。「文字列」は複合図形の名前や、画層の名前などを表すのに使われます。例えば、文字スタイルの "STANDARD" などがそうですね。文字列は変数と混同しないように、ダブルクォーテーションマーク(") で囲むように書くというお約束になっています。OK ?

話しかけてみましょ

では、画層について聞いてみましょう。AutoCAD は常に "0" と言う名前の画層を用意していますので、これについて聞いてみましょう。（数の 0 じゃなくて "0" っていう名前ですから、勘違いしないでね）

```
コマンド ： (tblsearch "LAYER" "0")
((0 . "LAYER") (2 . "0") (70 . 0) (62 . 7) (6 . "CONTINUOUS"))
```

なんか、たくさん出てきましたけど。一つずつ見ていけばそう難しいものではありません。ただし、皆さんの嫌いなマニュアルの手を借りないといけませんけど (^^;

第一回で出てきた「開いた括弧は必ず閉じる」を覚えてますか？どの開いた括弧がどの閉じた括弧に対応しているのか注意深く見て下さい。いくつかの部分に別れていることに気が付くと思います。さあ、マニュアルを開きましょう。(※ AutoCAD 2000 の場合) オンラインヘルプ - DXF リファレンス - TABLES セクション - LAYER の項目です。

(0 . "LAYER")	この情報は画層についてだよ
(2 . "0")	名前は "0" です
(70 . 0)	フリーズもロックもされていません
(62 . 7)	色は 7 (白）に設定されています
(6 . "CONTINUOUS")	線種は "CONTINUOUS"（実線）に設定されています

大体、こんな感じの内容になります。ただし、設定状態によってはこの通りになっていないかもしれません。びっくりしないでね。

●ぞろぞろ出てくる括弧だらけの値については、またこの次のお話にしましょう。

おまけに、もう一つ

もう一つ、聞いてみましょう。次は、いつも書いている図形についてです。図形について聞くんですから、事前に直線を一本引いておきましょうね。

コマンド ： **(entsel)**
オブジェクトを選択: ← ここで直線をマウス(とか)で選択です
(<図形名: 2560540> (6.90465 5.31244 0.0))

また今度も、ぞろぞろと出てきましたね。でも、画層について聞いたときよりも少ないですよね。今度は、二つの部分に別れているのが解りますか？

<図形名: 2560540>	図形の名前
(6.90465 5.31244 0.0)	直線を指示したときの座標

今まで使っていた ( ) では無く、< > になっているところにご注目。この「図形の名前」というのは、私たちが変数に名前を付けるように、AutoCAD も図形の一つ一つに名前を付けているんです。ですから、その名前を教えてくれたんですね。この直線についてもっと詳しく知りたい場合は、「この名前の図形（<図形名: 2560540>）について教えてね」と聞けば、もっと詳しく教えてくれます (^^)
次の (6.90465 5.31244 0.0) は図形を選択したときの座標です、書き方は(X座標 Y座標 Z座標) になっています。もう一度 (entsel) から初めて、直線を違う位置で選択すると、この値は違うものになっていますよ。

今回やったのは

「"0" という名前のついたレイヤーの情報を教えてね」と「この図形は？」でした。もう解ってる？ これは、失礼しました。「文字列」という言葉も登場しましたね、覚えておきましょう。

けっこう、親切でしょ

聞きさえすれば AutoCAD は意地悪せずにいろんなことを教えてくれます。いい奴でしょ。

避けて通れない道

屍を乗り越えて

たくさんの人々がこの魔物に挑み、そして敗れてきました。その魔物の名前は「リスト」なんと恐ろしい名前でしょう。その容姿は変幻自在、さまざまなものに姿を変えてあなたを襲ってきます。（負けちゃだめだよ(^^)）
でも、この魔物はいつか退治しなければなりません。なぜなら「避けて通れない道」だからです。

敵をしり、己をしれば．．．

リストの正体は「値」です、いままで出て来た「数」や「文字列」と同じ種類のものなんです。「なーんだ、たいしたことないじゃん」と思ったあなたは油断のしすぎです、なめてかかると返り討ちにあいますよ。では、気を引き締めて取り掛かりましょう。

リストを作ってみましょ

では、どんなに恐ろしいか、見てみましょう。見る為には、リストを作らなくちゃいけませんね。

```
コマンド : (list 1 4)
(1 4)
```

これが「リスト」です。どうです、恐いでしょ。えっ、あんまり恐くない？あなたは、とても勇気のある人なんですねぇ。

もうご存じの通り list は関数です。後の 1 4 が引数ですね。だから、これは「1 と 4 でリストを作ってね」となります。いままでやってきた数や文字列などの値は、一度に「一つのこと」（1 なら 1 という数、"0" なら "0" という文字列のことです）しか扱っていませんでしたが、今回返ってきた値（リスト）には 1 という数と 4 という数が一緒になって返ってきています。こういった「いろんな値が一緒になった値」がリストです。（これは、必ず試験にでます。赤ペンチェックしておきましょう(^^)）

ちょっと寄り道（代入って覚えてます？）

少し前の話しになりますが、第二回の代入は、まだ忘れてませんよね。「変数には、値が代入出来る」ですよ、覚えてますか？

```
コマンド : (setq var 12)
12
```

こんなこと、や

```
コマンド : (setq var "STANDARD")
"STANDARD"
```

こんなこと、が出来るんでしたね。リストも値なのですから、当然代入も出来ちゃいます。

```
コマンド : (setq var (list 1 4))
(1 4)
```

ちょっと難しいと思ったら、開いた括弧と閉じた括弧の対応をよーく見て下さい。(list 1 4) はさっきやったのと同じです、返って来た値は (1 4) というリストでしたね。ここまで解れば大丈夫 (^^)

var という変数に (1 4) という値を代入しているのが解りましたか？AutoCAD は最初に (list 1 4) を解釈して、(1 4) という値を作ります、その後に (1 4) という値を var という変数に代入するんですね

まだまだ、こんなもんじゃない

一緒にできる値は、二つだけじゃありません。いーっぱい一緒にしてもいいんです。

コマンド : **(setq var (list 1 4 5 7 0 5))**
(1 4 5 7 0 5)

数の 1, 4, 5, 7, 0, 5 が一緒になったリストです。
さらに、一緒にできるのは数だけじゃありません。文字列だって一緒にできます。

コマンド : **(list 1 3 "BLOCK" 5)**
(1 3 "BLOCK" 5)

当然リストだって一緒にできちゃいます。

コマンド : **(list 1 40 (list "LAYER" 6) 2)**
(1 40 ("LAYER" 6) 2)

どうです、だんだん訳ワカメになってきましたか？("LAYER" 6) は「リストという一つの値」と考えると、少し頭の中がすっきりするかもしれませんね。少し慣れたらいろんなリストを作ってみて下さい。だいたいこれで「リスト」の正体が見えてくるのではないかと思います。でも、知っておかなければならない魔物がもう一人。

●ここで使った "LAYER" とか 6 とかには深い意味は全然ありませんので適当に変えてやってみて下さいね (^^)

もう一人の魔物

では、早速作ってみましょう。

コマンド : **(cons 0 "LAYER")**
(0 . "LAYER")

今度は、値と値の間に . が入っていますね。これが「ドットペア」と呼ばれる特殊なリストです。他のリストとは区別されていますが、同じ「値」であることには変わりがありません。「こういうのもあるんだな」と心にとめておきましょうね。

前回 AutoCAD が答えてくれたこと

画層について聞いたときにの答えをもう一度見てみましょう。

((0 . "LAYER") (2 . "0") (70 . 0) (62 . 7) (6 . "CONTINUOUS"))

まるっきり「リスト」そのものですよね。ドットペアのリストを、五つ一緒にしたリストなのが、解りますか？こんなふうに、リストにするとたくさんの値を一つの値として扱うことが出来るので、便利なことが多いんです。例えば、

コマンド : **(setq var (tblsearch "LAYER" "0"))**

とすれば、現在の画層 "0" の状態が、一つの変数に代入しておけるでしょ。

●きれい好きな人は、変数 var をきれいにお掃除しておきましょうね。

魔物は強すぎた

今回は「リスト」という魔物についてのお話でした。まだ、退治することはできませんでしたが、その姿については見えてきたのではないでしょうか？最終的には魔物を退治し、そして操れる様にならなくてはいけません。頑張りましょうね。

図形について教えて！

しつこく続くんだなぁ

AutoCAD に図形について聞いたときのことを思い出してください。中途半端で終わってましたよね。今回は、その続きです。

もう一度やってみてね

まず最初に直線を一本引いたんでしたよね。今回も同じでいきましょう。

```
コマンド ： (setq var (entsel))
オブジェクトを選択: ← ここで直線をマウス(とか)で選択です
(<図形名: 2560548> (6.0 3.0 0.0))
```

返ってきた値がリストなのは、前回説明したことでなんとなく解ると思います。以前やったのと違うのは、戻ってきた値を var という変数に代入しているところですね。

本当に代入されたかを調べるには !var をコマンドラインから入力すれば、代入された値が見られるんでしたよね。

```
コマンド ： !var
(<図形名: 2560548> (6.0 3.0 0.0))
```

取り出しちゃうぞっ

AutoCAD にこの直線についてもっと詳しく知りたいときは「この名前の図形について教えてね」と聞くんでしたよね。欲しいのは (<図形名: 2560548> (6.0 3.0 0.0)) の中の <図形名: 2560548> だけなんですけど。よけいな値まで一緒にくっついていますね。でも、ちゃんと取り出す方法はありますのでご心配なく。さっそく、二つあるうちの前の方の値を取り出してみましょう。

```
コマンド ： (car var)
<図形名: 2560548>
```

取り出せることは取り出せましたが、ここでパニックおこしちゃった人は多いんじゃないかな？ いきなり var の中から値が出てきたみたいで、びっくりしちゃったでしょ。 (car var) っていうのは (car (<図形名: 2560548> (6.0 3.0 0.0))) と同じことなんです。 !var と (car var) でこの感覚を是非つかんで下さい。

car は言わずと知れた「関数」で「このリストの中の一番最初にある値を頂戴ね」です。（これもやっぱり難しいのかなぁ (?_?)）

その次にある値はどうやって取り出すのかって？まあ、そんなに焦らずに。今回は使わないので、やめときましょうよ (^^;

これでやっと聞ける

図形の名前が取り出せたので、やっと AutoCAD に今回描いた直線について聞く準備ができました。では、いよいよ

```
コマンド ： (entget (car var))
((-1 . <図形名: 2560548>) (0 . "LINE") (5 . "29") (100 . "AcDbEntity")(67 . 0) (8 . "0") (100 . "AcDbLine") (10
5.0 2.0 0.0)(11 7.0 4.0 0.0) (210 0.0 0.0 1.0))
```

うひょー、今度は超強力ですね。でもよーく見ると、画層について聞いたときとリストの様子がよく似てませんか？

- (100 . "AcDbEntity") と (100 . "AcDbLine") は R13J じゃないと出てこないはずです。他のバージョンを使っているなら、出てこないのがあたりまえですから、「壊れた！」と叫ばないように注意しましょう。

恐怖の時間

マニュアルはたしかに読みにくいかもしれません。でも、きっと手元にはこれしか無いはずです。それに、なれてしまえば貴重な情報が満載なんですよ。ここは、忍耐です。

(※ AutoCAD 2000 の場合) オンラインヘルプ - DXF リファレンス - ENTITIES セクション - LINE の項目です。「図形に共通のグループコード」も参照してください。

(-1 . <図形名: 2560548>) は 「図形名は <図形名: 2560548> です」ですね。次の(0 . "LINE") は、画層のときと同じく「直線（ライン）についての情報です」でいいですね。こうやって調べていくと大体、こんな感じの内容になります。

(-1 . <図形名: 2560548>)	図形の名前は <図形名: 2560548> です
(0 . "LINE")	直線についての情報です
(5 . "29")	ハンドル番号は "29" です
(67 . 0)	図形はモデル空間にあります
(8 . "0")	画層の "0" に描いてあります
(10 5.0 2.0 0.0)	始点の座標は (5.0 2.0 0.0) です
(11 7.0 4.0 0.0)	終点の座標は (7.0 4.0 0.0) です
(210 0.0 0.0 1.0)	押し出し方向は WCS の Z 軸に平行です

この中で注目してほしいのは、

(10 5.0 2.0 0.0)	始点の座標は (5.0 2.0 0.0) です
(11 7.0 4.0 0.0)	終点の座標は (7.0 4.0 0.0) です

の二つだけです。これは必ず見つかりますし、理解もしやすいと思いますのでこの二つだけは、忘れずに理解しておきましょう。

- 探し方さえ覚えてしまえば、要領は一緒です。今回も「○○は○○です」が出てきましたが、始点 と 終点 以外の解らない○○の中身は、わからないままでもかまいません。ここで大切なのは、あくまでマニュアル（ヘルプ）を調べるということなんですから。

面白くない！

そうでしょ、気持ちはよくわかります。どこに書いてあるか解らない項目を探すのは、きっと苦痛以外のなにものでもないでしょう。でもこれは、慣れるしかありません。もし、中級の LISP 使いになったとしても結局はこうやってマニュアル（ヘルプ）を調べることが多いんです。

今回は AutoCAD に図形の詳細まで詳しく聞いてみました。シャイな奴のわりには、今回はよくしゃべってましたね (^^) きれい好きな人は var のお掃除も忘れずにね。

お料理しましょ

魔物の逆襲

再びリストのお話しです。そういえば、オートデスクフォーラムで誰かは忘れましたけど、「リストって電車みたいなもの」って言ってましたっけ？その方が理解しやすければ、「電車みたいだなぁ」と思って見て下さいね。このリストっていうのは、LISP をやっているとこれから何度も登場します、もう嫌だって言っても許してもらえません。だって、「避けて通れない道」なんですもん。

フライパンは．．

今回は、リストをお料理しますけど、フライパンやお鍋はいりません。用意するのは assoc subst という二つの関数です。その他、ax bx cx という三つの変数も用意しておきましょう。別に用意といっても取り立てて何かするわけではないのですが、名前だけは決めてあげましょうね。

●ax bx cx には特別な意味はありません。ただの a b c でもいいんですだけど、それだけじゃ寂しいので後ろに x を付けて見ました (^^;

材料を用意しましょう

用意するのは、リストです。では、元になるリストを作りましょう。

```
コマンド : (setq ax (list 1 "cabbage"))
(1 "cabbage")
コマンド : (setq bx (list 2 "pumpkin"))
(2 "pumkin")
コマンド : (setq cx (list 3 "carrot"))
(3 "carrot")
コマンド : (setq var (list ax bx cx))
((1 "cabbage") (2 "pumpkin") (3 "carrot"))
```

ちょっと長かったですか？でもこれで、変数 var にリストが代入できました。一発勝負が好きな人は次のような入力のしかたもありますよ

```
コマンド : (setq var (list (list 1 "cabbage")(list 2 "pumpkin")(list 3 "carrot")))
((1 "cabbage") (2 "pumpkin") (3 "carrot"))
```

どちらの方法でも var には同じ値が入りますよ。それと、今回はとっても紛らわしいので、必要の無くなった変数を、ちゃんとお掃除します。

```
コマンド : (setq ax nil)
nil
コマンド : (setq bx nil)
nil
コマンド : (setq cx nil)
nil
```

一発勝負に出た人は、ax bx cx という変数を使っていないので、お掃除する必要はありません。お掃除しちゃったんですから、もう ax bx cx の中には、もうなーんにもありません。ここまで入れた var 以外の値については、きれいさっぱり忘れましょう。

●変数 var までお掃除しちゃダメですよ。せっかくの苦労が水の泡になっちゃいます。もしお掃除しちゃったら、「ふりだしに戻る」です。余談ですが、cabbage はキャベツ pumpkin はカボチャ carrot はニンジンですよ (^^) この後 onion（タマネギ）も登場します。

鍋の中を覗くの図

ここで var の値を確認して見ましょう。

コマンド ： **!var**
((1 "cabbage") (2 "pumpkin") (3 "carrot"))

(1 "cabbage")と(2 "pumpkin")と(3 "carrot")という三つの値が一緒になったリストなのは、もうお解りでしょうか？ここで注目するのはそれぞれ三つの値すべてに 1 2 3 という番号を付けておいたというところです。そうすることで 2 という番号だけを頼りにこのリストから値を取りだすことができます。

コマンド ： **(assoc 2 var)**
(2 "pumpkin")

assoc は、指示されたリスト（var）の中の一番目の値である (1 "cabbage") という値を取り出します。その後 (1 "cabbage") の中にある一番最初の値が 2 と同じかを確認し、もし同じなら (1 "cabbage") を値として返します。もし、同じでなければ次の値である (2 "pumpkin) を取り出して．．．と同じ動作を繰り返して、リスト（var）から「2 という値が先頭にあるリスト」を探していきます。取り出した値は後で使いますので、変数に代入しておきましょう。

コマンド ： **(setq ax (assoc 2 var))**
(2 "pumpkin")

●ここで「取り出す」という言葉を使いましたが、元のリスト var の値が変化してしまうわけではありません。そのことは、!var で確認して下さいね。

下ごしらえを

予告どうりにタマネギの登場です。タマネギの下ごしらえにかかりましょう。

コマンド ： **(setq bx (list 4 "onion"))**
(4 "onion")

これで下ごしらえは完了です。

●(4 "onion") の 4 はなんでも構いません。"pumpkin" と同じにしたければ 2 がお薦めですね。

味付けをかえましょ

コマンド ： **(subst bx ax var)**
((1 "cabbage") (4 "onion") (3 "carrot"))

subst は、二番目の引数 ax で指示された値を、三番目の引数 var の中から探して、見つかった値を一番目の引数 bx と入れ替えた値を返す関数です。鍋の中のカボチャとタマネギを交換するだけで、随分苦労しちゃいましたね。

●これでもまだ var の値が変わった訳ではありません。var の値を変えるには setq が必要なんですよね。

やっぱりリストは．．．

リストを作るところからやったのが、良かったのか悪かったのか．．．(?_?) でも、自分でいろんなリストを作ったり、作ったリストを修正したりして覚えていくのが、いいんじゃないかと思うんですよねぇ。始めのうちは、きっと思い通りにならなくて苦労するでしょうけど (^^;
今回のお片付けは、ax bx var のお掃除をしておしまいです。はい、みなさんお疲れさまでした。

だったら位置が変えられる

もうネタばれ？

そろそろ、何をしようとしているのか解っちゃいました？ AutoCAD に図形について答えてもらったリストを変更して、変更した内容で図形を修正してもらおう、というのが狙いだったのです。さぁ、初級講座もそろそろ終わりが近くなってきましたよ。

新しい変数の登場

今回も新しい変数を用意しておきましょう。名前は ent にしましょうね （ent は entity の略です。えんと君とでも呼びましょうか (^^)

それと、もう一つ mod という名前の変数も用意しておきましょう。（mod は modify の略です。こっちは、モドさんにでもしようかな）

いままでと同じに

直線を描き、その情報を取得し、ent に代入しましょう。

```
コマンド ： (setq var (entsel))
オブジェクトを選択: ← ここで直線をマウス(とか)で選択です
(<図形名: 2560548> (6.0 3.0 0.0))

コマンド ： (setq ent (entget (car var)))
((-1 . <図形名: 2560548>) (0 . "LINE") (5 . "29") (100 . "AcDbEntity")(67 . 0) (8 . "0") (100 . "AcDbLine") (10 5.0 2.0 0.0)(11 7.0 4.0 0.0) (210 0.0 0.0 1.0))
```

そろそろ括弧の数を数えるのに足の指が必要になりましたか？ (^^)

●ここで var の値を見るには !var、ent の値を見るには !ent ですよ。

お料理の時間

いよいよ佳境に入って来ましたよ。覚悟はいいですか？要領は、前回やったのと同じ料理方法です、今回はタマネギじゃありませんけど (^^; では、下準備から始めましょう。

最初に、始点の位置を示す値を取り出しておきましょう。そうですねぇ、取り敢えず変数 ax にでもいれておきましょうか。

```
コマンド ： (setq ax (assoc 10 ent))
(10 5.0 2.0 0.0)
```

今回は、直線の始点の位置を変更してみますので、始点の位置の値を AutoCAD が図形について答えてくれたのと同じ形式で偽造してみましょうね。こっちは、変数 bx に入れておきましょう。

```
コマンド ： (setq bx (list 10 7.0 3.0 0.0))
(10 7.0 3.0 0.0)
```

10 は始点を表す為のグループコードでしたよね。その次の 3 つの値はそれぞれ X座標 Y座標 Z座標を示しています。

●X、Y、Z の座標は、今回描いた直線の始点の値と同じにすると全然面白くないので、違う値にしましょうね。Z 軸は、変更してもよく見えないことが多いので、変更するのは X と Y がお勧めです。

次は、ent に入っている始点の値を、偽造した値 bx に変更してみましょう。あっ、そうだ、変更するといっても ent が変更されるわけじゃなく、変更したリストが返ってくるだけなんでしたよね。戻ってきたリストは mod に代入しておきましょう。

```
コマンド ： (setq mod (subst bx ax ent))
((-1 . <図形名: 2560548>) (0 . "LINE") (5 . "29") (100 . "AcDbEntity")(67 . 0) (8 . "0") (100 . "AcDbLine") (10 7.0 3.0 0.0)(11 7.0 4.0 0.0) (210 0.0 0.0 1.0))
```

始点の値が変わっていることを確認してみて下さいね。

グラフィック画面にご注目

変更した mod の通りに AutoCAD に図形を変更してもらいましょう。

コマンド ： **(entmod mod)**
((-1 . <図形名: 2560548>) (0 . "LINE") (5 . "29") (100 . "AcDbEntity")(67 . 0) (8 . "0") (100 . "AcDbLine") (10 7.0 3.0 0.0)(11 7.0 4.0 0.0) (210 0.0 0.0 1.0))

直線の始点が変わりましたか？ 動作としてはストレッチに似てますね。

●今回使った変数は ax bx var ent mod でした、結構たくさん使いましたね。きれい好きな人は、お掃除忘れずにね。

すごーい、かな？

これで LISP を使った図形の修正方法のお話しはお終いです。「LISP で図形が修正できる」これだけで、いろんなことができそうな気がしてきませんか？ その他にも、図形を描く為に必要最低限の情報でリストを作って「これで図形を描いてね」と AutoCAD にお願いするなんてこともできます。それだけじゃありません、もっといろんなことが LISP でできるんです。（もちろん、できないこともありますが (^^;)
でも、それはまた別のときのお話しにしましょうね。

最終回？

実は、この講座は今回が事実上の最終回なんです。
次回は、御挨拶と無駄話だけとなってます。

残念ですが、最終回

何といっても初級ですから

今回の初級講座では、コマンドラインから直接入力できる範囲に限定して話をしてみたのですが、いかがだったでしょうか？

そろそろ、毎回コマンドラインから入力するのがおっくうになったんじゃないですか？もちろん現在使っている LISP プログラムは毎回コマンドラインから入力なんかしてないですよね。でも、その説明は初級の域を越えるという判断であえて避けてみました。何といっても初級ですから (^^;

いまのレベルは．．

現在の初級講座終了時点でのレベルは「ようやくプログラムが書き始められるかな？」くらいです。時間があれば「庭の小径」が書けるようになるまで説明したかったのですが、それは中級講座ということにしましょう（いまのところ予定はありませんけど (^^;)

最後に

長いことつまらない話しにお付き合いいただきまして、本当にありがとうございました。既に「こんなこと解っとるわい」という人には無駄ですけど、これから LISP を始める人のきっかけになればいいなと思って始めたことですのでどうかご了承下さいませ。それでは、またどこかでお会いしましょう (^^)/~~~