はじめての IJCAD カスタマイズ

マクロメニュー(DIESEL)からAutoLISPへの移行
本格カスタマイズへの入口

IJCAD 2015 対応版

2015-04-22

提供

インテリジャパン株式会社

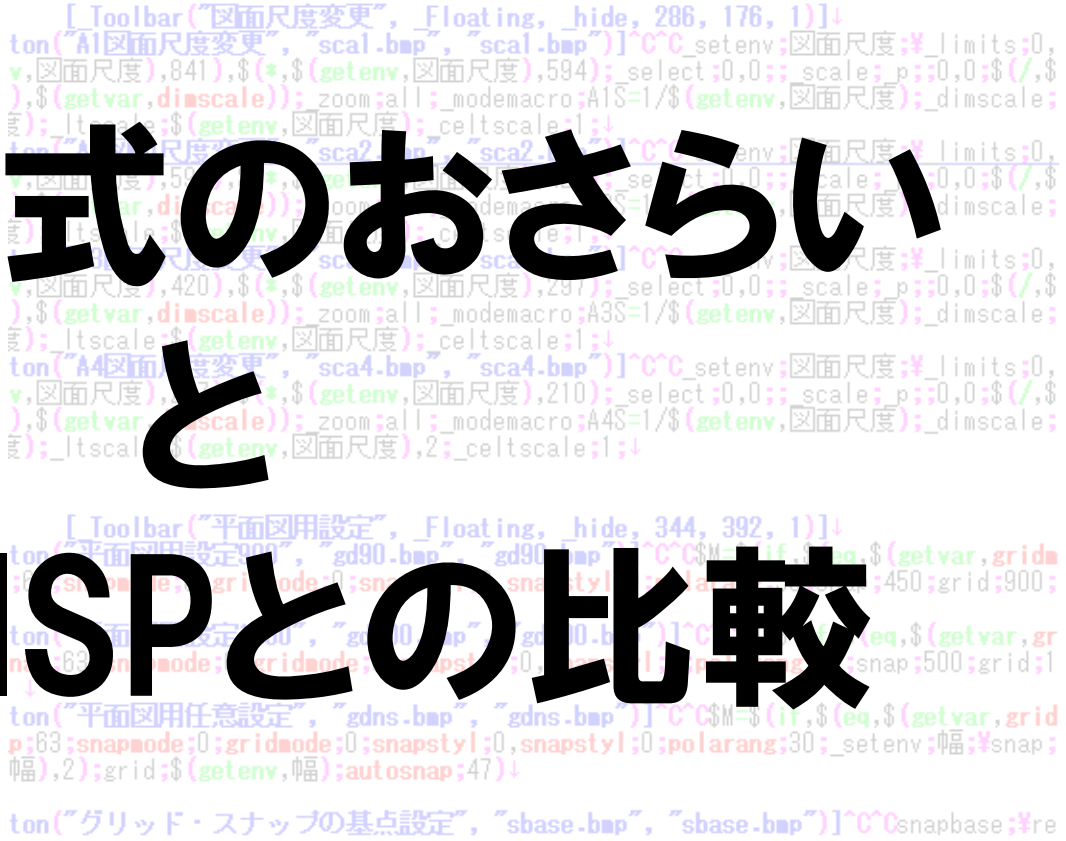
システムメトリックス株式会社

Gizmo Labs

内容

- DIESEL式のおさらいとAutoLISPとの比較
- DIESEL式からAutoLISP式への置き換え例
- AutoLISPの基礎
- AutoLISPの演習
- CADの設定を読み書きする（システム変数）
- DIESELメニューの置き換え実践

DIESEL式のおさらい
と
AutoLISPとの比較



DIESEL式のおさらい

- DIESEL(Direct Interpretively Evaluated String Expression Language)式は、
  文字列を受け取り結果として文字列を返す言語です。
- 20数個の関数が用意されています。
- 式はCADのメニューに記述します。
- 独自コマンドの追加などは出来ません。
- 書式の基本形は、

  最初　　$M=$(関数名,引数1[,引数2…])
  2つ目以降は　　$(関数名,引数1[,引数2…])　　でもOK.
  入れ子も可能　　$M=$(関数名,$(関数名,引数1[,引数2…])[,引数2…])
- 条件分岐は if による2分岐のみ出来ます。

AutoLISPとは・・・？

LISP は LISt Processing の略。
つまり、”リスト”を”処理”する事に長けたプログラム言語です。

- AutoCAD や IJCAD をカスタマイズするための、最もポピュラーな開発言語です。（他には、C++や.net ベースの言語などがあります。）
- 図形の情報を取得・変更したりする関数や、コマンドを実行する関数など、便利な関数が数百以上と非常に多く用意されています。
- AutoCAD/IJCAD とエディタだけあれば開発できます。
- 関数型言語に属し、コンパイルが不要で開発スピードが早いのが特徴。
- バージョンアップによる影響を受けにくいのも特徴。
- 頑張ると、ダイアログを使ったプログラムも作れます。

DIESEL式 と AutoLISPの比較

- **機能の比較**

内容	DIESEL関数	AutoLISP関数
四則計算	△（引数10個まで）	○
コマンド実行	○	○
独自コマンド作成	×	○
ユーザ入力	△（コマンド入力待ちのみ）	○
図形の生成・更新・削除	△（コマンド利用のみ）	○
条件分岐	△（ifの2分岐のみ）	○
ループ処理	×	○
複雑な処理のプログラム	×	○
エラー処理	×	○
ダイアログ利用	×	○

DIESEL式 と AutoLISPの比較

- **関数の対比** 1

	DIESEL関数	AutoLISP関数	内容
四則演算	＋	＋	(加算)
	－	－	(減算)
	＊	＊	(乗算)
	／	／	(除算)
比較演算子	=	=	(...に等しい)
	!=	/=	(...に等しくない)
	<	<	(...より小さい)
	<=	<=	(...より小さいか等しい)
	>	>	(...より大きい)
	>=	>=	(...より大きいか等しい)

DIESEL式 と AutoLISPの比較 2

- **関数の対比 2**

	DIESEL関数	AutoLISP関数	内容
論理演算子	and	and	論理積（xxxかつxxx）
	or	or	論理和（xxxまたはxxx）
	xor	xor	排他的論理和。(or のうち、重複を除く)
	if	if	条件分岐（もしxxxならAそれ以外ならB）
文字列操作関数	linelen	なし	ステータス領域の最大表示文字数を返します
	strlen	strlen	文字列の半角での文字数を返します
	substr	substr	文字列の一部を切り出します
	upper	upper	大文字に変換します
	eq	eq	文字列同士が等しいかどうかの比較
	eval	eval	文字列の評価

DIESEL式 と AutoLISPの比較 3

- **関数の対比** 3

	DIESEL関数	AutoLISP関数	内容
文字列関数	edtime	なし	指定した書式で日時を返します
	nth	nth	引数うちn番目のデータを取り出します
	index	なし	カンマ区切りの文字列から指定位置の文字列を取り出します
その他	fix	fix	小数点以下を切り捨てます
	getenv	getenv	環境変数、レジストリの値を読み出します
	getvar	getvar	システム変数値を返します
	rtos	rtos	指定した形式・精度の実数に編集します
	angtos	angtos	指定形式で角度を返します

DIESEL式 と AutoLISPの比較 4

- 共通点
  - どちらもメニューに記述できます。
  - 前述の関数の対比表に示したとおり、同じ関数名のものが多くあります。
- 書式も 少し似ています

  DIESEL式：　**$M=$**（関数名,引数1[,引数2…]）

  LISP式：　（関数名　引数1 [引数2 …]）

  * 赤字の部分が無いことがわかります。

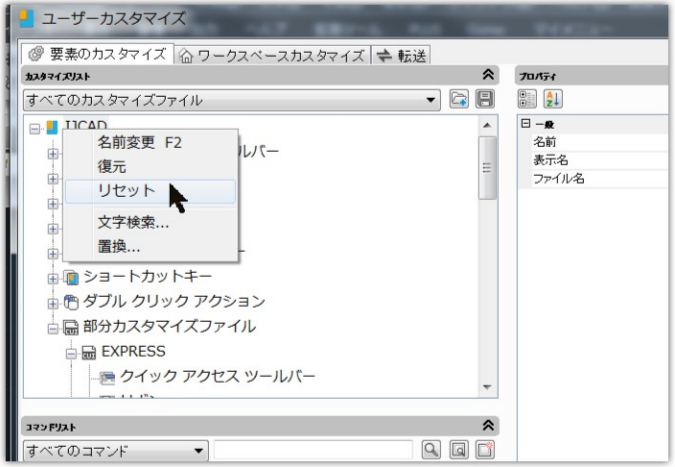
- 以上のように、DIESELの置き換えという点に絞ると、似ている部分が多いため敷居は高くありません。この講習では、上記のDIESEL式 にある関数の他に、いくつかのLISP関数利用に絞ってあります。まずは DIESEL式を置き換えを行ってみてから、ステップアップしていくと良いでしょう。
- その他のAutoLISP関数について知りたい場合は、最後のページで紹介しているドキュメント等を参考にして下さい。

DIESEL式からAutoLISP式への置き換え例

```
01. (setq a 1 b 1)
02. (if (= a b)
03.   (progn
04.     (princ "¥nA = B ")
05.     (setq a (+ a 10) b (- b 10))
06.   )
07. )
08. A = B
09. ''',-9'''
```

メニューファイルについて

- IJCADでは、AutoCADのメニューファイル（.mnu, .mns,.cui.,cuix）を読み込むことが出来ます。
  - 但し、標準メニューの内容は似ていますが、AutoCAD と IJCAD で異なるため、メニュー全体を書きだしした AutoCAD のメニューファイルを IJCAD にそのままの状態で読み込ませる事はおすすめできません。
    新規に IJCAD 上で作成してしまう方がいいです。
  - 独自のツールバーを作成していて、その部分だけを書きだしたメニューファイルであれば、特に編集せずに読み込める事が多いです。
  - 万が一、メニューの構成などが壊れてしまっても、IJCAD の初期状態に戻すことができます。



DIESEL式 から AutoLISPの置き換え例 1

- 以下のDIESELマクロメニューをAutoLISPの式に置き換えてみます。

円文字に文字を入れるメニュー

```
^C^C_Circle;¥$M=$(*,$(getvar,dimscale),5);_Change;L;;P;C;2;;_TEXT;J;M;@0,0;
```

AutoLISPへ置き換えたメニュー

```
^C^C_Circle;¥;(* (getvar "dimscale") 5);_Change;L;;P;C;2;;_TEXT;J;M;@0,0;
```

- 赤字のDIESEL式の部分のみLISP式に変更します。
- 上記のように、AutoCAD LTのメニューをIJCADに移植する場合にはこのような変換方法で実現することができます。

DIESEL式 から AutoLISPの置き換え例 2

- 以下のDIESELマクロメニューをAutoLISPの式に置き換えてみます。

画層名：壁に、幅50で二重線を作図するメニュー

```
^C^C$M=$(if,$(eq,$(getvar,clayer),壁),,-layer;m;壁;c;253;;lw;0.13;;;)_dline;c;n;w;50
```

AutoLISPに置き換えたメニュー

```
^C^C(if (/= (getvar "clayer") "壁") (command "-layer" "m" "壁" "c" "253" "" "lw" "0.13"
"" ""))_dline;c;n;w;50
```

- if文内のコマンドはLISPの（command）関数を使います。
  - （command）関数では、コマンドの各ステップを ”-layer” といった形で ”（ダブルクォート）で囲って記述します。
  - Enter はメニューマクロでは ; ですがLISPのコマンド文では ”” で記述します。
  - 入力待ちにしたい場合は pause を記述します。

  ex. (command "_line" pause pause "") ; ← これで単線分になります。

  ex. (command "_line" ¥¥ ¥¥ "") ; ← ¥¥ でもOKです。

DIESEL式 から AutoLISPの置き換え例 3

- 以下のDIESELマクロメニューをAutoLISPの式に置き換えてみます。

文字の画層に2㎜で印刷される文字を作成するメニュー

```
^C^C$M=$(if,$(eq,$(getvar,clayer),文字),,_-layer;m;文字;c;92;;lw;0.13;;;)^C^C_-style;tex
t1;simplex,bigfont;;;;;;;_dtext;¥$M=$(*,$(getvar,LTSCALE),2.0);0;
```

長いので、バラして考えてみましょう。

```
^C^C$M=$(if,$(eq,$(getvar,clayer),文字),,_-layer;m;文字;c;92;;lw;0.13;;;)     ← レイヤを調整してる部分
^C^C_-style;text1;simplex,bigfont;;;;;;;                                    ← 文字を調整してる部分
_dtext;¥$M=$(*,$(getvar,LTSCALE),2.0);0;                                    ← 文字を記入している部分
```

これをLISP式に置き換えしてみます。

```
^C^C(if (/= (getvar "clayer") "文字")(command "_-layer" "m" "文字" "c" "92" "" "lw" "0.13" "" ""))
                                                                  ↑ レイヤを調整してる部分
^C^C_-style;text1;simplex,bigfont;;;;;;;                          ← 文字を調整してる部分
(command "_dtext" pause (* (getvar "LTSCALE") 2.0) "0")           ← 文字を記入している部分
```

繋げ直して、以下のようになります。

```
^C^C(if (/= (getvar "clayer") "文字")(command "_-layer" "m" "文字" "c" "92" "" "lw" "0.13" "" ""))^C^C_
-style;text1;simplex,bigfont;;;;;;;;(command "_dtext" pause (* (getvar "LTSCALE") 2.0) "0")
```

このように、分けて考えれば長い DIESEL式のものも難しくありません。

LISPの基礎

```
(setq a 1 b 1)
(if (= a b)
  (progn
    (princ "¥nA = B ")
    (setq a (+ a 10) b (- b 10))
  )
)
A = B
'''-9'''
```

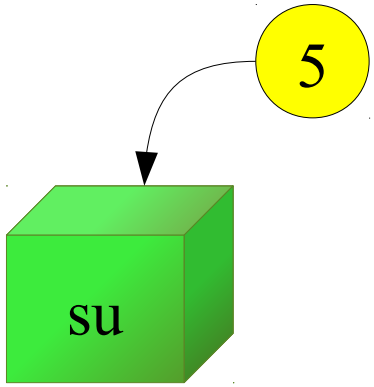
LISP式の記述

- LISP式の基本構文は、（関数 引数［引数 ...］）です。
- 関数の中には複数の引数を指定するものもあります。
  - (+1 2 3)　　(if (評価式) (A式) (B式))
  - (setq シンボル 値)
- 全て文字列として扱われるDIESEL式とは異なり、LISP式は色々なデータタイプがあります。
  - データタイプの例： 整数 、実数、文字列、リスト、選択セット、...etc
- LISPは、シンボルを使用してデータを保持します。
  - (setq str1 "これは文字列です") ；str1に文字を割り当てた例
- 次の文字はシンボル名に使用できません 。
  - ((左カッコ)、) (右カッコ)、. (ピリオド)、' (アポストロフィ)、" (ダブルクオート)、; (セミコロン)、数字のみの名前

LISPのデータタイプ例

整数	1　0　-1
実数	0.12345　　0.0　　-0.123　　2.01011E+007
文字列	“文字列”
シンボル	T　nil　pt1　str
リスト	(1.0 1.0 0.0)　　("this" "that" "the other") ((0 . 2) (1 . “a”))
選択セット	<Selection set: 1>
図形名	<Entity name: 27f0540>
ファイルディスクリプタ	#<file "c:¥¥myinfo.dat">

変数に値を代入する setq



- 変数とは、値を入れておく箱のようなものです。
- setq は、変数に値を代入するための関数です。
- 以下は、変数 su に 5 を代入します。
  (setq su 5)
- 値を確認するには ! を使います。
  !su　　> 5
- 変数に文字を代入するときは、文字をダブルコーテーションで囲みます。
  (setq moji "ABC")
- 計算などに使用できます。(関数の引数として、変数を利用。)
  (+ su 2)　　> 7

setenv と setq の違い

- setenv は、環境変数としてPC（のレジストリ）に保存されます。
- setq は、変数としてメモリに保持されます。

- このような違いから、Setenv は、CADソフトを起動し直しても覚えていますが、setq はCADを終了、あるいは不要になった段階で開放されます。

コメント

- コメントを記入することで、分かりやすいプログラムにしておき、メンテナンス性を高めます。
  LISPプログラム中のコメントは ;（セミコロン）の後に記入します。

```
; コメント
(setq p1 (list 10 20)) ; コメント
```

  - ;（セミコロン）から行末までがコメントになります。
  - 但し、メニューマクロ内では、;（セミコロン）は、Enter になるので、コメントは書けません。

```
(setq a 1 b 1)
(if (= a b)
 (progn
  (princ "¥nA = B ")
  (setq a (+ a 10) b (- b 10))
 )
)
A = B
'''-g'''
```

LISPの演習

```
(setq al '((name box) (width 3) (size 4.7263) (depth 5)))
((NAME BOX) (WIDTH 3) (SIZE 4.7263) (DEPTH 5))
(assoc 'size al)
(SIZE 4.7263)

(assoc 'weight al)
nil

(setq bigl '((a b c)(d)(e f g)))
(assoc 'e bigl)
(E F G)

(setq elist (entget (car (entsel "図形選択:"))))
(princ (strcat "¥n選択図形は、" (cdr (assoc 0 elist))))
選択図形は、*選んだ図形タイプ名*
```

コマンドライン上で実行してみる 1

- IJCAD のコマンドラインから、プログラムを1行づつ実行してみます。
- 足し算 6+3 をするとき、コマンドラインに次のように記述します。
  (+ 6 3)
  9　　← 計算結果
- LISPは開きカッコ（ ではじまり、閉じカッコ ）で終わります。
  - ＋ は最初に書きます。（前置記法といいます。）
  - 各シンボルの間はひとつ以上の半角スペースを入れます。
  - すべて半角で記述します。
- ー（減算）、＊（乗算）、／（除算）、の場合も同じです。
  6−3 →(− 6 3)　　6−3−1 →(− 6 3 1)
  3×2 →(＊ 3 2)　　3×2×4 →(＊ 3 2 4)
  8÷2 →(／ 8 2)　　8÷2÷4 →(／ 8 2 4)

コマンドライン上で実行してみる 2

- 以下のようなカッコの複数ある計算式の場合、
  (3 + 2) *(5 - 2)
- LISPでは次のように記述します。
  (* (+ 3 2) (- 5 2))
  - 構文は、カッコの一番内側かつ左側から処理されます。
- 開いたカッコは必ず閉じる必要があります。つまり（と）の数は、必ず同じになります。
- カッコの記述を間違った例
  (* (+ 3 2) (- 5 2)　；カッコが足りない
  (* (+ 3 2 (- 5 2)))　；カッコの数は合ってるが (3 + 2) *(5 2) の計算だと閉じる位置が違っている

コマンドライン上で実行してみる 3

- princ 関数は、画面にメッセージを表示します。
  (princ “こんにちは”)
- 次のプログラムを実行すると、メッセージが2回表示されます。

```
(defun C:test02 ()
        (princ “こんにちは”)
)
```

- プログラムの最後に（princ）を追加すると、不要なメッセージが表示されなくなります。

```
(defun C:test02 ()
        (princ ”こんにちは”)
        (princ)
)
```

コマンドライン上で実行してみる 4

- LISPではユーザの操作から情報を取得して値を保持・利用することが出来ます。次のように記述してEnterしてみてください。
  (getpoint)
- 何も表示されませんが、クリックすると座標のリストが返ってきます。
  (-40900.0　30500.0　0.000000) ←（X座標点 Y座標点 Z座標点）です。
  Getpoint 関数はユーザに点を問い合わせる関数です。
- もちろん変数に値を保持する事もできます。
  (setq pt1 (getpoint))
  !pt1 >> (-40900.0　30500.0　0.000000)
- 尚、次のように記述して実行すると、”点を指示：”がメッセージとして表示されます。
  (getpoint “¥n点を指示：”) ； ¥nは改行の意味です。

ファイルから実行してみる 1

- LISPプログラムは、テキストファイルに記述したプログラムから実行出来ます。テキストファイルは通常テキストエディタで記述します。
- テキストエディタとは、文章やプログラムを作成するためのソフトです。
- Windows に標準搭載のテキストエディタはメモ帳で、メモ帳での開発も可能です。
- 市販・フリー・シェアなどのソフトで、メモ帳より便利なテキストエディタがあります。
  - 秀丸、NotePad++、EmEditor、さくらエディタ、TeraPad、など。
  - **対応するカッコの強調表示**や**対応するカッコに移動**する事のできるエディタがお勧めです。

ファイルから実行してみる 2

- エクスプローラのフォルダオプションの表示タブで、「登録されている拡張子は表示しない」のチェックを外しておきます。
- C：ドライブに LISP という名前のフォルダを作成します。
- メモ帳を起動します。
  「スタート」ー「すべてのプログラム」ー「アクセサリ」ー「メモ帳」
- 次のプログラムを入力します。

```
(defun C:study01()
        (setq p1 (list 10 10))
        (setq p2 (list 50 10))
        (setq p3 (list 30 30))
        (command "line" p1 p2 p3 "C")
)
```

- ファイル名：study01.lspで、ファイルを保存します。
  拡張子をlspにしないと、mystudy01.txtになってしまいます。

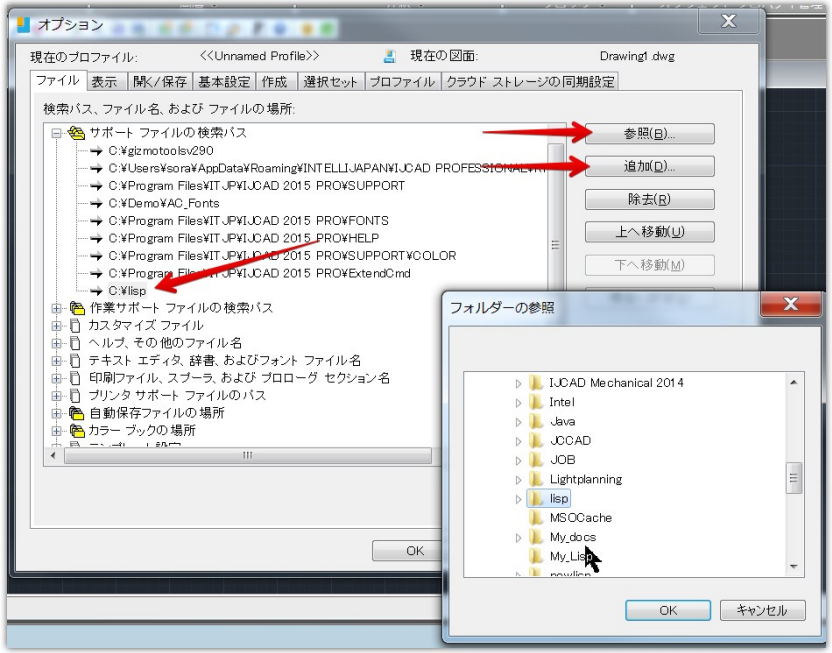
ファイルから実行してみる 3

- CAD にプログラムをロード（読み込み）します。
  フォルダの区切りは、￥ではなく、/（スラッシュ）を使います。
  (load "C:/lisp/study01.lsp")
  ¥マークを使用する場合は次のようになります (load "C:¥¥lisp¥¥test01.lsp")
- プログラムを実行します。
  study01
- 実行できたでしょうか？

- 尚、Defun は、関数を新たに定義する関数ですが、C:xxx のように
  関数名が C: から始まる場合、通常のCADコマンドのように利用可能になります。

ファイルから実行してみる 4

- ユーザー定義パスを設定すると、ロード時のフォルダ指定が不要になります。
  (load "study01.lsp") でOK。
- 設定方法
  (1) 右クリック → オプションから「ファイル」タブを表示して
  (3) 「サポートファイルの検索パス」をクリックし、[追加]ボタン、参照ボタンをクリックし、C:¥lisp を選択して追加



ファイルから実行してみる 5

- さらに、毎回プログラムをロードするのは面倒なので、ロードの手間を減らす自動ロードを試してみましょう。
- IJCADの場合、gcad.lsp というファイルがあると、起動した時と図面を開いたときに、自動的にロードするようになっています。
- ですので、gcad.lspに開発したプログラムをロードする処理を書いておくと、毎回プログラムをロードしなくてもよくなります。
  注）自動ロードの処理内で command 関数は使ってはいけません。

- その他に、メニューに書いてしまうという方法も。

  例えば、my_prog.lsp というファイルで my_cmd1 というコマンドを定義している場合、
  メニューマクロに

  ^C^C(load "my_prog.lsp")^C_my_cmd1;
  というように記述しても動作します。

メッセージを表示してみる

- LISPからメッセージを表示するには、いくつかの方法があります。

- メッセージダイアログを表示する：alert 関数
  明確に注意や警告を促したい場合などで使用します。
  (alert “メッセージ”)

- コマンドラインに表示する：princ 関数/ print関数 / prin1関数

```
(defun C:study02()
  (princ “こんにちは”)
)
```

  - プログラムの最後に (princ) を追加すると、不要なメッセージが表示されなくなります。

```
(defun C:test020()
  (princ "Hello")
  (princ)
)
```

LISPからコマンドを実行する 1

- command関数は、線分や円などのコマンドを実行します。
- コマンドラインから、実行する順序通り、記述します。
- コマンドラインから線分コマンドを実行します。

```
: line
始点: 10,20
A=角度/L=長さ/<終点>: 50,80
A=角度/L=長さ/F=フォロー/U=取消し/<終点>:
```

- comnmand関数で表現すると

```
(command "line" "10,20" "50,80" "")
```

- 最後に空Enterがあることを注意します。

LISPからコマンドを実行する 2

■ **線分を作図するプログラム。**

```
(defun C:study03 ()
      (setq p1 (list 10 20))
      (setq p2 (list 50 80))
      (command "line" p1 p2 "")
      (princ)
)
```

- 点の座標に変数を利用しています。
- 座標点リストは、(X Y [Z]) のリストとして定義します。

Command 実行の練習問題

- 円（circle）を作図するプログラムを開発します。
- 中心点は、100,200 半径は 75 とします。
- コマンド名・ファイル名は、study04.lsp とします。
- まずは、circleコマンドをコマンドラインで実行し、コマンドの順番をメモしましょう。
- 半径・中心点の値は変数に入れます。中心点はlistです。

簡単なデバッグの方法 1

- プログラムが正常に動作しないとき、コマンドラインに着目します。
- 括弧の数が正しくないとき、セミコロンの数が正しくないファイルをロードしようとすると、**エラーが出て、ロードできません。**

```
(defun C:test ()
        (princ “Hello) )
        (princ)
)
```

エラー：不正なリスト

簡単なデバッグ方法 2

- **関数名や文法が間違っているとき。**
  →ロードはできますが、実行時にエラーが出ます。

```
(defun C:test ()
        (prink “Hello”)
        (princ)
)
```

- エラー: 関数定義がありません:PRINK
  →エラーの内容が表示されます

処理の制御：条件分岐 if

- if関数を使うことで、条件により処理される内容を制御できます。
- if の使い方

  a=1のときは su に3を代入し、a≠1のときは0を代入する。

```
(if (= a 1)
    (setq b 3)   ←条件が正しい（nil 以外の）とき実行
    (setq b 0)   ←条件が間違っている（nil の）とき実行
)
```

- 条件式

(= a b)	等しい	(eq a b)	同一
(/= a b)	等しくない	(equal a b)	評価結果が同一
(> a b)	より大きい	(>= a b)	より大きいか等しい
(< a b)	より小さい	(<= a b)	より小さいか等しい

ブロック化 progn

- **Progn の使い方**
  if の条件式で実行できるプログラムは1行だけです。
  複数行実行したとき、progn で行をブロック化します。

```
(if (= a 1)
    (progn                    ← (= a 1) が nil以外のとき
        (setq su1 3)
        (setq su2 6)
    )
    (progn                    ← (= a 1) が nilのとき
        (setq su1 0)
        (setq su2 1)
    )
)
```

字下げ（インデント）

- プログラムを見やすくするため、書き出し位置をずらします。
- 一般的に、スペース2つ～4つかタブを使います。

■見にくいプログラム

```
(if (= a 1)

(progn
(setq b 3)
(setq c 4)
)

(progn
(setq b 0)
(setq c 1)
)
)
```

■見やすいプログラム

```
(if (= a 1)
    (progn
        (setq b 3)
        (setq c 4)
    )
    (progn
        (setq b 0)
        (setq c 1)

    )
)
```

■見やすいプログラム2

```
(if (= a 1)
    (progn
        (setq b 3)
        (setq c 4) )
    (progn
        (setq b 0)
        (setq c 1) ))
```

ifの練習問題

- 朝か昼かを聞いて、あいさつするプログラムを開発します。
- [1]=朝 [2]=昼:
  おはよう　　← 朝のとき
  こんにちは　　← 昼のとき
  と表示するコマンド。

- コマンド名・ファイル名は、study05 とします。
- ユーザーの入力は、getint 関数 を使います。
  (getint "今はいつごろですか？ [1]=朝 [2]=昼 : "))
- メッセージの表示は princ 関数を使います。
- 入力=1のとき、「おはよう」を表示し、それ以外のとき、「こんにちは」を表示します。

```
ACADVER          "17.1i" (読込み専用)
ACISOUTVER         120
AFLAGS           0
ANGBASE           0.0
ANGDIR           0
APBOX            0
APERTURE           10
AREA           0.0 (読込み専用)
ATTDIA           0
ATTMODE          1
ATTREQ           1
AUDIOICON        1
AUDIOICONCOLOR
AUDIOICONSCALE     1.0
AUDITCTL         0
AUNITS           0
AUPREC           4
AUTOMENULOAD       1
AUTOSNAP         0
AXISMODE         0
AXISUNIT         X=   0.0  Y=   0.0  Z=   0.0
BACKZ          0.0 (読込み専用)
BASEFILE         "C:¥Program Files¥IJCAD 7 Pro¥Templates¥icad.dwt
BINDTYPE         0
BKGCOLOR           256
BLIPMODE         0
CAMERADISPLAY      0
CAMERAHEIGHT       0.0
CDATE          20100315.1 (読込み専用)
CECOLOR          "ByLayer"
CELTSCALE          1.0
CELTYPE          "BYLAYER"
CELWEIGHT          -1
CEPSNTYPE          0 (読込み専用)
```

CADの設定を読み書きする システム変数

システム変数とは

- システム変数は、現在のCADの状態や既定値などが保存されている設定値です。（とても沢山あります。）
- どんなシステム変数があるか調べるには、Setvar コマンドを使用します。

```
: setvar
    ?=一覧/変数名 <FILEDIA>: ?
    表示するシステム変数 <*>: *
     3DDWFPREC            2
     ACADLSPASDOC         0
     ACADPREFIX           "C:¥Users¥hoge¥Documents¥;C:¥Program Files¥IJCAD 7 Pro¥Fonts¥;"
      （読込み専用）
     ACADVER              "17.1i"  （読込み専用）
     ...
```

システム変数をLISPで使用する

■ LISPではシステム変数を以下の関数から利用します。

システム変数の値を取得する　　：getvar 関数
システム変数の値を変更・更新する　：setvar 関数

例

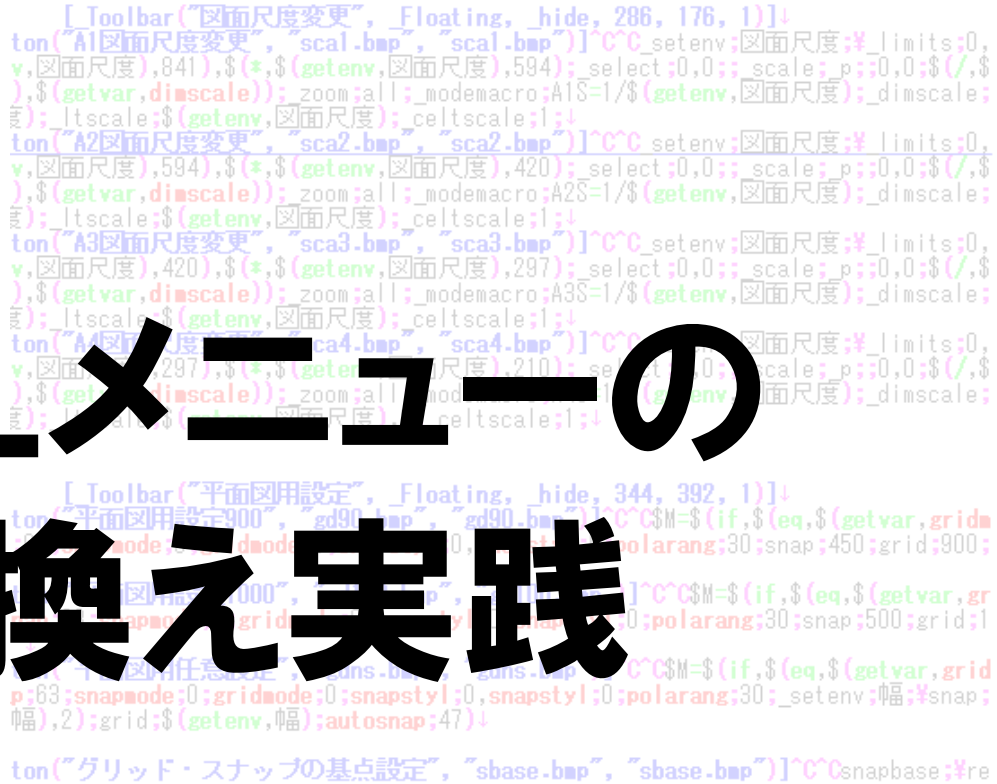
```
; 現在のエンティティスナップモード
(getvar "OSMODE")
"37"

(setvar "OSMODE" 0)    ← 実行するとESNAPが NON になります。
```

システム変数の注意点

- システム変数には、図面ファイルに保存されるもの、システム（PC）に保存されるもの、保存されないものがあります。

- IJCAD の場合
  変数はあるものの、機能していないシステム変数がたまにあります。（AutoCAD 互換のために用意されているもので、アップデートで対応したりすることもあります。）



DIESELメニューの置き換え実践

置き換え練習問題 -1

■ カーソルサイズを変更する式を変えてみましょう

```
^C^C$M=$(if,$(=,$(getvar,cursorsize),100),cursorsize;5,cursorsize;100);
```

■ 回答例

```
^C^C(if (= (getvar "cursorsize") 100) (setvar "cursorsize" 5) (setvar "cursorsize" 100))
```

別の回答例

```
^C^C(setvar "cursorsize" (if (= (getvar "cursorsize") 100) 5 100))
```

置き換え練習問題 -2

■ 回転複写する式を変えてみましょう

```
^C^C_select;$M=$(if,$(getvar,CMDACTIVE),¥,)_rotate;p;;¥c;
```

- **このメニューマクロの場合、IJCAD では単純に DIESEL式 を LISP式に書き換えただけでは動きません。**
- **このような場合は、置き換える事をやめてコマンドの流れを見て書き直しします。手順としては、図形を選択してコピーし、元の図形を回転するという命令になります。**
- **図形の選択は（ssget）関数を使います。**

- **回答例**

```
^C^C(setq ss1 (ssget))(command "_copy" ss1 "" "@" "@" "" "_rotate" ss1 "" "")
```

置き換え練習問題 -3

■ 補助線なし長さ寸法の式を変えてみましょう

- **メニューマクロ**

```
^C^C-layer;m;SUNPO;c;8;SUNPO;;dimSE1;ON;dimSE2;ON;dimlinear;¥¥¥_dimcontinue
```

- **回答例**

```
^C^C-layer;m;SUNPO;c;8;SUNPO;;dimSE1;ON;dimSE2;ON;dimlinear;¥¥¥_dimcontinue
```

置き換え練習問題 -4

■ ± 寸法公差記入の式を変えてみましょう

- **メニューマクロ**

```
^C^C^C_dimtp;_non;¥_dimoverride;_dimtp;;_dimtm;$m=$(getvar,dimtp);
dimtol;on;_dimtfac;0.7;_dimtolj;1;_dimtdec;4;_dimtzin;8;;¥;
```

- **回答例**

```
^C^C^C_dimtp;¥(command "_dimoverride" "dimtp" """dimtm" (getvar "dimtp") "dimtol"
"on" "dimtfac" "0.7" "dimtolj" "1" "dimtdec" "4" "dimtzin" "8" "" pause)
```

参考
（おまけ）

時間を取得する LISP関数の例

- DIESEL式の editime がLISPに無いので、かわりに利用できる時間を取得する関数を作成します。
  以下は現在の時間を取得して文字列として返すLISP関数です。
  自作の関数には接頭子を付ける事が推奨されていますので、ここでは “my:” を付けています。

```
; 今日の年月日を取得して返す関数　(my:today) →　“2010-10-10”
(defun my:Today ( / d yr mo day)
     (setq d (rtos (getvar ”CDATE”) 2 6)
           year (substr d 1 4)   month (substr d 5 2)  day (substr d 7 2))
     (strcat year ”-” month ”-” day)) ;_my:today
; 今の時間を時:分:秒で時間を返す　(my:time) →　“01:23:45”
(defun my:Time ( / d hr m s)
     (setq d (rtos (getvar ”CDATE”) 2 6)
           hr (substr d 10 2)  m (substr d 12 2)   s (substr d 14 2))
     (strcat hr ”:” m ”:” s)) ;_my_time
```

時間を取得する LISP関数の例

- 前ページの続きです。現在の時間から算出して曜日の文字列を返すLISP関数です。

```
; 今日の曜日を計算して返す関数　(my:weekday) →　“水”
(defun my:weekday (/ w)
      (setq w (rem (fix (getvar ”date”)) 7)))
      (cond
        ((= w 0) (princ ”月”))  ((= w 1) (princ ”火”))
        ((= w 2) (princ ”水”))  ((= w 3) (princ ”木”))
        ((= w 4) (princ ”金”))  ((= w 5) (princ ”土”))  ((= w 6) (princ ”日”))
)) ;_my:weekday
```

; 上記3つの関数を利用して editime **と似たような結果を返す関数を作ります。**

- ; (my:edtime) → ”2010-11-12(**金曜日**),15:33:5”

```
(defun my:edtime (/)
   (strcat (my:today) ”(” (my:weekday) ”曜日),”(my:time)))
```

本講座で使用したLISP関数まとめ

DIESEL式と同じ関数名のものは、同じ用途。以下はそれ以外のLISP関数についてです。

- Command 関数　… CADのコマンドを実行します。
- Setq 関数 … 変数シンボルに値を代入します。
- Getint 関数 … ユーザが整数を入力するまで待機(一時停止)し、入力された数値を返します。
- Getpoint 関数 … ユーザが点を入力するまで待機(一時停止)し、入力された点を返します。
- Ssget 関数 … ユーザが図形を選択するまで待機し、選択セット返します。
- Princ 関数 … 文字列を返します。
- Defun 関数 … 関数を定義します。
- Load 関数 … LISPプログラムをロードします。
- Strcat 関数 … 引数の文字列を結合した文字列を返します。
- Cond 関数 … 条件にあった処理を実行する、多分岐条件関数としての機能を果たします。
- Rem 関数 … 引数の余剰（余り）を返します。

初心者が陥りやすい LISP トラブル

- LISPファイルに全角スペースが混ざってしまいうまく動かない。
  ＞ 文字列内以外で全角文字は使えません。

- カッコの閉じ忘れがあり、うまく動かない。
  ＞ 対応するカッコが確認出来るエディタを使いましょう。

- 関数は読み込めるが場合によってうまく動かない。
  ＞ グローバルな変数を多く利用していると他の関数で使用している変数と
  バッティングしてしまうことがあるかもしれません。
  できるだけローカル変数を使うようにしましょう。

- Command 関数がうまく動かない。
  ＞ Command 関数に記述する内容が長くなってうまく動かない場合は、
  コマンド関数の処理を分割すると解決しやすくなる事があります。
  ex. (command "_line" pause pause "" "_circle" pause 100)
  ↓
  (command "_line" pause pause "")
  (command "_circle" pause 100)

参考：LISP関連情報

- オンライン
  - AutoLISP 関数のドキュメントとサンプル
    http://wiki.gz-labs.net/
- 書籍
  - AutoCAD AutoLISP 実践ガイド
    著者：T.ボスフィールド　　発売：(株)星雲社
  - AutoLISP 徹底活用ガイド
    著者：落合重紀　　発行：(株)翔泳社
  - AutoLispの初歩
    著者：岡島正夫　　発行：東海大学出版会