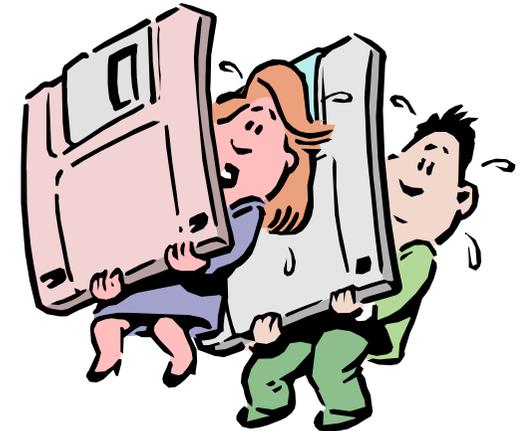
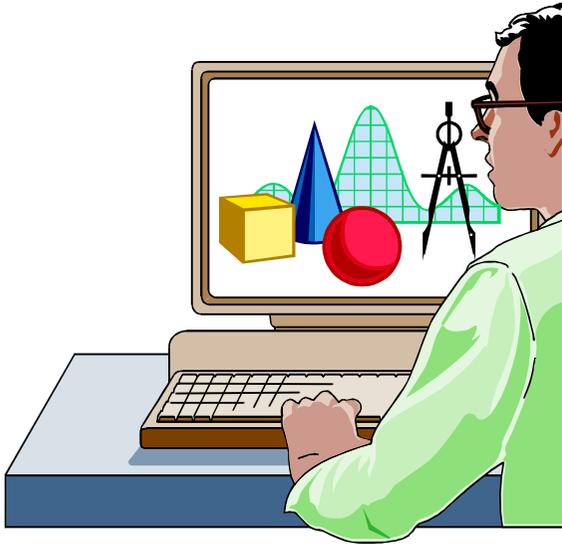


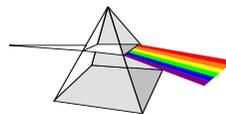


# AutoCAD

*Plus jamais ça !*



# Programmation AutoLISP Avancée



**Christian REB CAO/DAO Services**  
1 rue des Fleurs / 68230 ZIMMERBACH  
☎ 03 89 58 98 16 / ☎ 03 89 58 40 20  
✉ reb.c@wanadoo.fr  
Web: perso.wanadoo.fr / rebcao

Conseil - Formation - Développement

Votre Spécialiste AutoCAD

# Programme du module AutoLISP Programmation Avancée

## 1. Les chaînes de caractères

Concaténer, Afficher des chaînes de caractères  
Taille d'une chaîne de caractères  
Extraire une partie d'une chaîne de caractères  
Conversion MAJUSCULES/MINUSCULES d'une chaîne  
Conversion de différents types de données

## 2. Les itérations et les conditions

Les Itérations  
Les conditions

## 3. Les listes

Lire le nième élément de la liste  
Regrouper plusieurs listes en une liste unique  
Substituer un élément dans une liste

## Exercice N° 1

Insérer la date du jour sous 2 formats

## 4. Les paires pointées dans une liste

Construire une paire pointée  
Lire une paire pointée

## 5. Accès à la base de données du dessin

Les différents modes de sélection d'objets  
Nombre d'objets dans la sélection  
Traitement de la sélection

## 6. Les codes de groupe DXF

Les principaux codes

## Exercice N° 2 et 3

Changer le plan des objets, même PLAN que...  
Dessiner le trait d'axe d'un cercle ou d'un arc

## 7. Gestion de fichier

Ouvrir, Lire, Ecrire dans un fichier

## Exercice N° 4

Lire un fichier de point et les insérer

## 8. Les tables du dessin

Les différentes tables  
Lire une table

## Exercice N° 5 :

Trier une table dans un fichier





## Fonctions AutoLISP : Concatainer des chaînes de caractères

Une chaîne de caractères est constituée d'éléments de type alphanumérique (texte, numérique) notée entre guillemet. La fonction **STRCAT** permet de rassembler plusieurs chaînes de caractères en une seule.

( **STRCAT** "chaîne 1" "chaîne 2" VAR1 VAR2 )      *retourne une CHAINE unique*

*var1 et var2 sont obligatoirement des chaînes de caractères*

**exemple: afin d'éviter les fautes liées à la frappe directe créer un fichier CHAINE.LSP (à charger)**  
*(ne pas tenir compte des commentaires en italique dans le fichier LISP)*

```

( setq  C1  "BONJOUR"
        G1  "Monsieur"
        G2  "Madame"
        G3  "Mademoiselle"

        J1  ( STRCAT C1 " " G1 )      retourne "BONJOUR Monsieur"
        J2  ( STRCAT C1 " " G2 )      retourne "BONJOUR Madame"
        J3  ( STRCAT C1 " " G3 )      retourne "BONJOUR Mademoiselle"

        MD  ( STRCAT J2 " " J3 " et " J1 ".\nSoyez les bienvenus..." )

);setq

(princ)

```

┌ Espace fonctionnel



## Fonctions AutoLISP : Concatainer des chaînes de caractères (suite...)

( STRCAT "chaîne 1" "chaîne 2" VAR1 VAR2 )  
*var1 et var2 sont obligatoirement des chaînes de caractères*

charger le fichier : (load"chaîne")

*vérifier en mode direct au clavier :*

**!C1**                    *retourne "BONJOUR"*

**!J2**                    *retourne "BONJOUR Madame"*

**!MD**                    *retourne "BONJOUR Madame, BONJOUR Mademoiselle et BONJOUR Monsieur.\nSoyez les bienvenus..."*

**(prompt MD)**        *retourne BONJOUR Madame, BONJOUR Mademoiselle et BONJOUR Monsieur.  
Soyez les bienvenes...nil*





## Fonctions AutoLISP : Afficher des chaînes de caractères

Il existe plusieurs fonctions d'affichage pour les chaînes de caractères. Certaines de ces fonctions sont exclusivement réservées aux chaînes de caractères. Il s'agit des fonctions :

**PRIN1, PRINC, PRINT, PROMPT, WRITE-LINE**

### Format et correspondance des fonctions :

<b>PRIN1</b> (" ")	Affiche les chaînes à la suite en conservant les guillemets
<b>PRINC</b>	Affiche les chaînes à la suite sans afficher les guillemets
<b>PRINT</b> (" ")	Affiche les chaînes avec un saut de ligne en conservant les guillemets
<b>PROMPT</b> (*)	Affiche les chaînes à la suite sans afficher les guillemets
<b>WRITE-LINE</b> (*)	Affiche les chaînes avec un saut de ligne sans afficher les guillemets

*("") ces fonctions affichent toujours les chaîne de caractères en conservant les guillemets*

*(\*) ces fonctions n'acceptent que des chaînes de caractères*





# Fonctions AutoLISP : Afficher des chaînes de caractères

Créer le fichier **AFFICHE.LSP** dans lequel on utilise différentes fonctions d'affichage :

**fichier AFFICHE.LSP**

*(ne pas tenir compte des commentaires en italique dans le fichier LISP)*

```

(setq          SEPAR          " _et_" )

( prin1 G1 ) ( prin1 SEPAR ) ( prin1 G2 )   retourne  "Monsieur" et "Madame"

( princ G1 ) ( princ SEPAR ) ( princ G2 )   retourne  Monsieur et Madame

( print G1 ) ( print SEPAR ) ( print G2 )   retourne  "Monsieur"
                                                " et "
                                                "Madame"

( prompt G1 ) ( prompt SEPAR ) ( prompt G2 )   retourne  Monsieur et Madame

( write-line G1 ) ( write-line SEPAR ) ( write-line G2 )   retourne  Monsieur
                                                                    et
                                                                    Madame

```

 Espace fonctionnel





## Fonctions AutoLISP : Traitement des chaînes de caractères

Il est nécessaire de traiter les chaînes de caractères pour récupérer des informations du type : longueur, extraction, format. Il s'agit des fonctions suivantes :

### **STRLEN, SUBSTR, STRCASE**

#### **STRLEN**

**Longueur de la chaîne.**

*(strlen "CHAINE")*

(strlen G1)      *retourne 8*

(strlen MD)      *retourne 82*

#### **SUBSTR**

**Extraction d'une partie de la chaîne.**

*(substr CHAINE DEPART LONGUEUR)*

(substr J1 4 4)      *retourne "JOUR"*

(substr MD 25 12)      *retourne "Mademoiselle"*

#### **STRCASE**

**Convertit toute la chaîne en Majuscules ou Minuscules** *(strcase "Chaîne")* ou *(strcase "Chaîne" T)*  
*par défaut conversion en MAJUSCULES si T est précisé conversion en minuscules*

(strcase g2)      *retourne "MADAME"*

(strcase g2 T)      *retourne "madame"*





## Fonctions AutoLISP : Conversion des chaînes de caractères

Les fonctions suivantes permettent la conversion de **CHAÎNE** en **ENTIER**, en **REEL**, ou **ANGLE**.

### **ATOI, ASCII, ATOF, DISTOF, ANGTOF**

**Chaîne - ENTIER** ( **ATOI "chaîne"** )  
( atoi "451.236" ) *retourne 451*

**Chaîne - ENTIER** ( **ASCII "chaîne"** )  
( ascii g1 ) *retourne 77*

**Chaîne - REEL** ( **ATOF "chaîne"** )  
( atof "451.236" ) *retourne 451.236*

**Chaîne - REEL** ( **DISTOF "chaîne" <MODE>** )  
*MODE : 1 Scientifique / 2 Décimal / 3 Ingenierie / 4 Architecture / 5 Fractionnaire*  
( distof "145' 3/5\" 4 ) *retourne 1740.6*

**Chaîne - ANGLE** ( **ANGTOF "chaîne" <MODE>** )  
*MODE : 0 Degrés décimaux / 1 Degrés ' \" / 2 Grades / 3 Radians / 4 Géodésie*  
( angtof "45" 0 ) *retourne 0.785398 (radians)*





## Fonctions AutoLISP : Conversion des ENTIERS

Les fonctions suivantes permettent la conversion d' **ENTIER** en CHAINE ou REEL.

### **ITOA, CHR, FLOAT**

**Entier - CHAINE** ( **ITOA** <entier> )  
( itoa 451 ) *retourne "451"*

**Entier - CHAINE** ( **CHR** <entier> )  
( chr 66 ) *retourne "B"*

**Entier - REEL** ( **FLOAT** <entier> )  
( float 451 ) *retourne 451.0*





## Fonctions AutoLISP : Conversion des REELS

Les fonctions suivantes permettent la conversion de **REEL** en CHAINE, en ENTIER.

### **RTOS, FIX**

**Réel - CHAINE** ( **RTOS** <réel> <*MODE*> <précision> )  
*MODE : 1 Scientifique / 2 Décimal / 3 Ingenierie / 4 Architecture / 5 Fractionnaire*  
( rtos 451.23657 2 3 ) *retourne "451.237"*

**Réel - ENTIER** ( **FIX** <réel> )  
( fix 451.23657 ) *retourne 451*



## Fonctions AutoLISP : Conversion des ANGLES

La fonction suivante permet la conversion d' **ANGLE** (*exprimé en radian*) en CHAINE.

### **ANGTOS**

**Angle - CHAINE** ( **ANGTOS** <angle(radians)> <*MODE*> <précision> )  
*MODE : 0 Degrés / 1 Degrés ' " / 2 Grades / 3 Radians / 4 Géodésie*  
( angtos 1.5708 1 3 ) *retourne "90d0"*





## Fonctions AutoLISP : Les itérations : Boucles

Les fonctions ci-dessous utilisent le principe d'une boucle en programmation. *"TANT QUE, je fais..."*

### **WHILE, REPEAT**

**WHILE :** Evalue une expression tant qu'elle est vérifiée

```
(setq coef 1 total 231.75 )
```

```
(WHILE (< coef 25 )
```

```
    (setq total (* total coef))
```

```
    (prompt "\nRésultat : ") (princ total)
```

```
    (setq coef (1+ coef))
```

```
);while
```

**REPEAT :** Evalue n fois les fonctions ou arguments

```
( REPEAT 10
```

```
    ( calcul_coef )
```

```
    ( prompt "." )
```

```
);repeat
```





## Fonctions AutoLISP : Les itération : Boucles

La fonction ci-dessous utilise le principe de la boucle, en utilisant une liste d'arguments.

### **FOREACH**

**FOREACH :**      **Evalue la séquence pour chaque élément de la liste**

**soit la liste L1 contenant 7 éléments**

```
(setq txt "\n"  
      L1 (list "a" "b" "c" "d" "h" "k" "m")  
);setq
```

**la fonction FOREACH boucle tant que la liste n'est pas nil**

```
(foreach x L1  
  
        (print (ascii x))  
        (setq txt (strcat txt x))  
  
);foreach
```





## Fonctions AutoLISP : Les tests et conditions

Les fonctions ci-dessous testent une valeur et exécutent telle ou telle opération en fonction du résultat du test. Les fonctions sont les suivantes :

**IF, COND**

**IF :** Vérifie la condition, faire si vraie... Sinon faire...

```
( IF (= coef 0 )
```

```
  (progn
```

```
    (prompt "\nCalcul impossible avec une valeur nulle")
```

```
    (calcul 1)
```

```
  );progn si coef=0
```

```
  (progn
```

```
    (prompt "\nCalcul en cours...")
```

```
    (calcul coef)
```

```
  );progn si coef<>0
```

```
);if
```

**N.B. :** *la fonction **PROGN** est indispensable pour encadrer plusieurs fonctions à exécuter avec la fonction **IF**. Sans **PROGN** une seule fonction peut être exécutée si **VRAI**... ou si **FAUX**...*





## Fonctions AutoLISP : Les tests et conditions (suite...)

Les fonctions ci-dessous testent une valeur et exécutent telle ou telle opération en fonction du résultat du test. Les fonctions sont les suivantes :

IF, **COND**

**COND** : **Evalue et compare la valeur aux différentes conditions**

```
(setq rep (getstring "\nUnités : 1. Degré / 2. Grades / 3.Radians" )
```

```
(cond
```

```
( (= rep "1" ) (fonc_unite 1) )
```

```
( (= rep "2" ) (fonc_unite 2) )
```

```
( (= rep "3" ) (fonc_unite 3) (calcul 3) (affiche 3) )
```

```
( T (prompt "\nAucune valeur choisie!!!" ) )
```

```
);cond
```





## Fonctions AutoLISP : Les LISTES, manipulation et extraction

Les fonctions ci-dessous permettent la manipulation ou l'extraction de listes, il s'agit de :

**NTH, APPEND, MEMBER**

**NTH :** Retourne le nième élément de la liste (attention 0 = premier élément)

soit la liste L1 : (setq L1 ' (a b c d e f) )

(nth 3 L1)                      retourne D (4e élément)

(nth 6 L1)                      retourne nil (7e élément : inexistant)

**APPEND :** Regroupe les éléments de N listes en une liste unique

soit les listes L2 et L3 : (setq L2 ' ("a" "b" "c" "k" "e" "f") L3 ' (U3 V3 W3) )

(append L1 L2)                      retourne (A B C D E F "a" "b" "c" "k" "e" "f")

(append L1 L3)                      retourne (A B C D E F U3 V3 W3)

(append L1 L2 L3)                      retourne (A B C D E F "a" "b" "c" "k" "e" "f" U3 V3 W3)

**MEMBER :** Recherche une occurrence et retourne le reste de la liste  
(member <expression> <liste> )

(member "d" L2)                      retourne ("d" "e" "f")





## Fonctions AutoLISP : Les LISTES, modifier le contenu d'une liste

Il est souvent nécessaire de changer le contenu d'une liste par d'autres valeurs. Il s'agit de la fonction

### SUBST

**SUBST :** Echange le contenu d'un élément par une autre valeur

( subst <nouveau> <ancien> <liste> )

soit la liste L1 : (setq L1 ' ( a b c d e f ) )

(subst "coucou" (nth 2 L1) L1) retourne (A B "coucou" D E F)

soit la liste L2 : (setq L2 ' ( "a" "b" "c" "k" "e" "f" ) )

(setq L2 (subst "d" "k" L2)) retourne ("a" "b" "c" "d" "e" "f")





## AutoLISP travaux pratiques, Exercice N° 1 : Insérer la date du jour

L'exercice consiste à créer une fonction qui **insère la date du jour dans un dessin** en choisissant le format de la date.

L'analyse sommaire du problème me permet d'identifier les objectifs suivants :

1. Je veux créer un fichier programme (AutoLISP) **DATEJ.LSP** et la commande **DATEJ**
2. Je veux insérer la date du jour dans mon dessin
3. Je veux que la date soit un bloc + attributs appelé **DATEJ**
4. Je veux choisir deux formats pour la date : 16/07/1994 ou 16 Juillet 1994
5. Je veux positionner librement ce bloc à l'écran
6. Je veux pouvoir entrer l'échelle de mon bloc

### Indications :

- La variable système contenant la DATE est : **CDATE**
- Vérifier son format avant toute programmation





# AutoLISP travaux pratiques, Exercice N° 1 : Les PHASES

Cette fonction comprend les phases suivantes :



1. Demander à l'utilisateur le format choisi ( JJ/MM/AAAA ou JJ Le\_Mois AAAA)

2. Demander à l'utilisateur l'échelle du bloc

3. Récupérer la DATE système avec CDATE



4. Extraire le Jour, le Mois, l'Année

5. Prévoir la conversion numérique du mois en chaîne de caractères (Mars, Juillet, Septembre, etc)

6. Insérer le bloc DATEJ à l'écran et choisir l'échelle





# Corrigé Exercice N° 1

```
;;; -----  
;;; Programme      DATEJ.LSP  
;;; Insertion de la date du jour en BLOC+ATTRIBUTS  
;;; -----  
  
;;; -----  
;;; Fonction principale  
;;;  
  
(defun c:datej ( / lmois jj mm aa datej rep echbloc nombloc)  
  
    ;;; lmois = liste en clair des mois de l'année  
    (setq lmois (list  
              "Janvier" "Février" "Mars" "Avril" "Mai" "Juin"  
              "Juillet" "Août" "Septembre" "Octobre" "Novembre" "Décembre"  
            ));list  
    datej (rtos (getvar "cdate") 2 0) ;;; conversion de la date en chaine  
    jj (substr datej 7 2) ;;; extraction du JOUR  
    mm (substr datej 5 2) ;;; extraction du MOIS  
    aa (substr datej 1 4) ;;; extraction de l'ANNEE  
  
    rep (getstring"\nFormat pour la date : 1 : 16/07/1994 / 2 : 16 Juillet 1994 <1> : ")  
    echbloc (getdist "\nEchelle du bloc <1> : ")  
    nombloc "datej"  
  
);setq
```

.../...





## Corrigé Exercice N° 1 (fin)

```
;;; -----  
;;; test la condition pour le format  
;;;  
  
(if (= rep "2") ;;; test le choix du format de la date  
    ;;; format de la date en clair (12 septembre 1994)  
    (setq datej (strcat jj " " (nth (- (atoi mm) 1) lmois) " " aa) )  
    ;;; format de la date simplifié ( 12/09/1994 )  
    (setq datej (strcat jj "/" mm "/" aa) )  
);if  
  
;;; -----  
;;; test la valeur de echelle  
;;;  
  
(if (= echbloc nil)  
    (setq echbloc 1)  
);if  
  
(setvar "cmdecho" 0)  
(prompt "\nPosition de la DATE")  
(command "inserer" nombloc pause echbloc echbloc 0 datej)  
  
(princ)  
);defun  
  
;;;***** FIN *****  
(prompt "\nTaper DATEJ pour insérer la DATE du jour en BLOC")  
(princ) ;;; évite d'afficher nil au chargement du programme
```





## Fonctions AutoLISP : Les LISTES, paires pointées

Grâce aux paires pointées dans une liste, il est possible de faire un accès direct à un élément d'une liste quelque soit sa taille.

### CONS

#### CONS :

Construit une paire pointée si aucun argument n'est une liste, sinon CONS ajoute <élément1> dans la liste indiquée à la place de <valeur>.

(cons <élément1> <valeur> )

créer et exécuter le fichier **CONS.LSP**

```
(setq L1 (list (cons 1 "BONJOUR" )
              (cons 2 "Monsieur" )
              (cons 3 "Madame" )
              (cons 4 "Mademoiselle" )
              (cons 100 31 )
              (cons 200 145 )
              (cons "AXE" 12 )
            );list
      );setq
```

Vérifier le résultat de la liste avec !L1





## Fonctions AutoLISP : Les LISTES, lire des paires pointées

La lecture des paires pointées dans une liste se fait à l'aide de la fonction :

**ASSOC**

**ASSOC :** Recherche un élément clé dans une liste d'association de listes

**(assoc <clé> <liste>)**

**Soit la liste L1 (load"cons.lsp") :**

**( (1. "BONJOUR") (2. "Monsieur") (3. "Madame") (4. "Mademoiselle") (100 31) (200 145) ("AXE" 12) )**

**(assoc 2 L1)**                      *retourne ( 2 . "Monsieur" )*

**(cdr (assoc 2 L1))**                *retourne "Monsieur"*

**(assoc 200 L1)**                    *retourne ( 200 145 )*

**(assoc "AXE" L1)**                 *retourne ( "AXE" 12 )*





## Fonctions AutoLISP : Sélectionner des objets dans un dessin (suite...)

Il est souvent nécessaire de sélectionner des objets dans un dessin pour modifier, construire, etc. La sélection peut être effectuée à l'aide de plusieurs fonctions qui sont :

ENTSEL, **SSGET**, ENTNEXT, ENTLAST

**SSGET** : Créer un jeu de sélection

(ssget <mode> <pt1> <pt2> <filtres>)

l'argument mode est optionnel, il définit le mode de sélection choisi  
(exemple : "P" précédent / "D" dernier / "F" fenêtre / "X" tous les objets)  
<filtres>, exemple : ' ( ( 0 . "line" ) ( 8 . "axe" ) ) = ligne sur le calque "axe"  
' ( ( 62 . 1 ) ) = objets de couleur 1 (rouge)

créer le fichier **SEL2.LSP**

```
(defun c:sel2 ()  
  (setq e2 (ssget))  
  (princ e2 )  
  (princ)  
);defun
```

### Sélectionner un objet avec SEL2 (SSGET)

retourne *par exemple* <Selection set: 12> Numéro du jeu de sélection



## Fonctions AutoLISP : Sélectionner des objets dans un dessin (suite...)

Il est souvent nécessaire de sélectionner des objets dans un dessin pour modifier, construire, etc. La sélection peut être effectuée à l'aide de plusieurs fonctions qui sont :

ENTSEL, SSGET, ENTNEXT, ENTLAST

**ENTNEXT :**      **Retourne le nom de la première entité du dessin, ou si un nom d'entité est fourni, retourne le nom d'entité suivant.**

**(entnext <nom d'entité> )**

(entnext)                      *retourne <Entity name: 600989a>*

(entnext (car e) )            *retourne <Entity name: 600970b>*

**ENTLAST :**      **Retourne le nom de la dernière entité du dessin,**

**(entlast)**

(entlast)                      *retourne <Entity name: 702560b>*





## Fonctions AutoLISP : Exploiter des sélections d'objets

Les fonctions ci-dessous permettent la recherche, la vérification sur certains objets.

**SSLENGTH, SSMEMB, SSNAME**

**SSLENGTH :** Nombre d'objets dans la sélection  
(**sslength** <nom du jeu de selection> )

**SSMEMB :** Test si le nom d'entité se trouve dans la sélection  
(**ssmemb** <nom ent> <nom du jeu de selection> )

*Utiliser la fonction SEL2 pour sélectionner des objets*

(**ssmemb** (car e) e2 ) *retourne <Entity name: 600989a>*

*Utiliser SEL1 pour sélectionner un objet qui n'a pas été sélectionné avec SEL2*

(**ssmemb** (car e) e2 ) *retourne nil*

**SSNAME :** Retourne le nom d'entité indexé ( indice 0 = première entité )  
(**ssname** <nom du jeu de selection> <indice>)

(**ssname** e2 0 ) *retourne <Entity name: 600989a>*

(**ssname** e2 (sslength e2) ) *retourne nil*

(**ssname** e2 (1- (sslength e2) ) ) *retourne <Entity name: 702560b>*





## Fonctions AutoLISP : Accès aux caractéristiques des objets sélectionnés

La finalité de toutes ces fonctions est d'accéder de façon séquentiel aux informations relatives aux objets sélectionnés. Il s'agit de la fonction :

### ENTGET

**ENTGET :**            **Retourne les définitions de l'objet sous forme de liste**

**(entget <nom d'entité> )**

*Soit un dessin dans lequel le premier objet est une ligne et le dernier objet un cercle.*

(entget (entlast) )            *retourne la liste de la dernière entité du dessin*  
((-1 . <Nom de l'entité: 22304f8>) (0 . "LINE") (5 . "1F") (67 . 0) (8 . "PIECE")  
(10 275.15 163.593 0.0) (11 280.107 166.776 0.0) (210 0.0 0.0 1.0))

(entget (entnext) )            *retourne la liste de la première entité du dessin*  
((-1 . <Nom de l'entité: 2230500>) (0 . "CIRCLE") (5 . "20") (67 . 0) (8 . "PIECE")  
(10 279.675 163.028 0.0) (40 . 1.65343) (210 0.0 0.0 1.0))

*Utiliser SEL1 pour localiser un objet précis :*

(entget (car e) )            *retourne la liste de l'objet sélectionné par SEL1*

*Exemple de liste de définition pour un texte :*

((-1 . <Entity name: 60001d9e>) (0 . "TEXT") (8 . "CADRE") (6 . "CONTINUOUS") (62 . 6) (5 . "10452F") (10 -3014.72 -6231.53 0.0)  
(40 . 300.0) (1 . "Formation") (50 . 0.0) (41 . 0.9) (51 . 0.2617) (7 . "COMP") (71 . 0) (72 . 1) (11 -1909.01 -6231.53 0.0) (210 0.0 0.0 1.0) (73 . 0) )





## Fonctions AutoLISP : Les CODES de groupe DXF

AutoCAD utilise un code pour chaque caractéristiques d'objet (hauteur, calque, type de ligne, coordonnées XYZ, etc). On retrouve ces codes dans la liste résultant d'un **ENTGET**... Ces codes sont utilisés également dans la définition des fichiers DXF. Voici quelques exemples des codes les plus courants :

<b>-1</b>	NOM de l'ENTITE
<b>0</b>	TYPE d'ENTITE
<b>2</b>	NOM d'un ELEMENT (nom de bloc, étiquette d'attribut, etc)
<b>5</b>	Code de maintien de l'objet
<b>6</b>	Type de ligne de l'objet (uniquement si différent de DUCALQUE)
<b>8</b>	NOM de CALQUE (PLAN)
<b>10</b>	Point PRINCIPAL (départ ligne, pt insertion Texte/Bloc, etc)
<b>40-48</b>	Réel (Facteur d'échelle, Hauteur texte, rayon, etc)
<b>50-58</b>	Angles
<b>62</b>	Numéro de COULEUR (uniquement si différent de DUCALQUE)
<b>67</b>	ESPACE de l'entité (absent ou 0 = OBJET / 1 = PAPIER)
<b>100</b>	Marqueur de sous-classe de données
<b>999</b>	Commentaires
<b>1000</b>	Chaîne ascii de XDATA (255 octets)





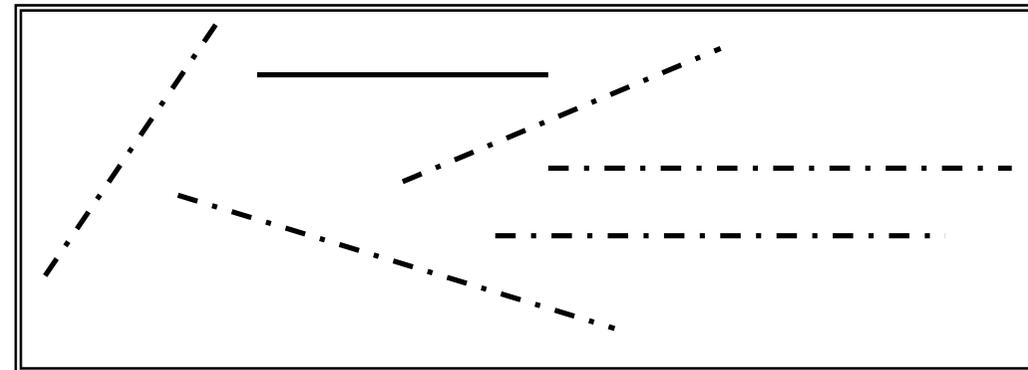
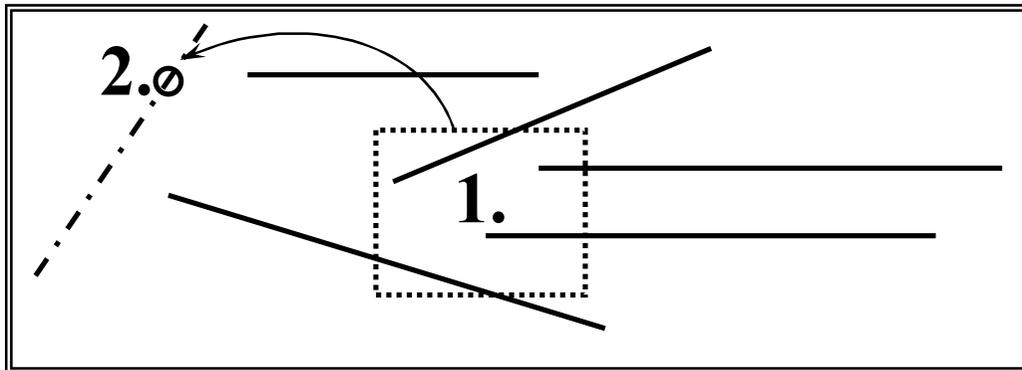
## AutoLISP TP, Exercice N° 2 : Changer le CALQUE par sélection

L'exercice consiste à créer une fonction qui **modifie le calque des objets sélectionnés en choisissant un objet possédant cette caractéristique : Même CALQUE que...** (*cette commande est apparue en version 2000 seulement...*)

L'analyse sommaire du problème me permet d'identifier les objectifs suivants :

1. Je veux créer un fichier programme (AutoLISP) **MEMCALQ.LSP** et la commande **MEMCA**
2. Je veux sélectionner des objets pour changer leur CALQUE à partir d'un objet possédant ce calque
3. Je veux afficher le choix des objets à changer
4. Je veux demander de sélectionner l'objet de référence
5. Je veux afficher le NOM du nouveau calque pour ces objets
6. Je veux récupérer le NOM du calque
7. Je veux utiliser la commande standard **CHPROP**

### Sélection de l'objet de référence (calque AXE)





## AutoLISP travaux pratiques, Exercice N° 2 : Les PHASES

Cette fonction comprend les phases suivantes :



1. Demander à l'utilisateur de sélectionner les objets à changer
2. Demander à l'utilisateur de sélectionner l'objet de référence pour le calque
3. Récupérer le nom du calque de la phase 2



4. Changer le calque des objets sélectionnés à la phase 1 avec la commande CHPROP
5. Afficher le nom du nouveau calque pour les objets de la phase 1





## Corrigé Exercice N° 2

```
;;; -----  
;;; Programme      MEMCALQ.LSP  
;;; Changer de CALQUE...même CALQUE que...tel objet (sélection)  
;;; -----  
  
;;; -----  
;;; Fonction principale  
;;;  
  
(defun c:memca (/ ent e)  
  
    ;;; -----  
    ;;; sélection des objets à changer  
    ;;;  
  
    (graphscr)                ;;; bascule en écran graphique  
  
    (prompt "\nChoix des objets à changer de CALQUE... ")  
  
    (setq ent (ssget) )      ;;; sélection des objets à changer
```

.../...





## Corrigé Exercice N° 2 (fin)

```
;;; -----  
;;; boucle si jeu de sélection vide  
;;;  
  
(while (= e nil)  
  
  (setq e (car (entsel "\nchoisir l'objet qui possède le CALQUE...")))  
  
  ;;; -----  
  ;;; test si un objet est sélectionné  
  ;;;  
  (if (/= e nil)  
    (progn  
      (setvar "cmdecho" 0)      ;;; supprime les messages des commandes AutoCAD  
      ;;; change les propriétés des objets sélectionnés  
      (command "chprop" ent "" "ca" (cdr (assoc 8 (entget e))) "")  
      ;;; affiche le nom du nouveau calque des objets  
      (prompt (strcat "\nNouveau CALQUE des objets ---> " (cdr(assoc 8 (entget e))) ) )  
    );progn  
  );if  
);while  
  
(princ)      ;;; évite affichage de nil en fin de fonction  
);defun  
  
;;;***** fin programme *****  
(prompt "\nTaper MEMCA pour changer le CALQUE des objets par sélection...")  
(princ)
```





## AutoLISP TP, Exercice N° 3 : Dessiner un trait d'axe

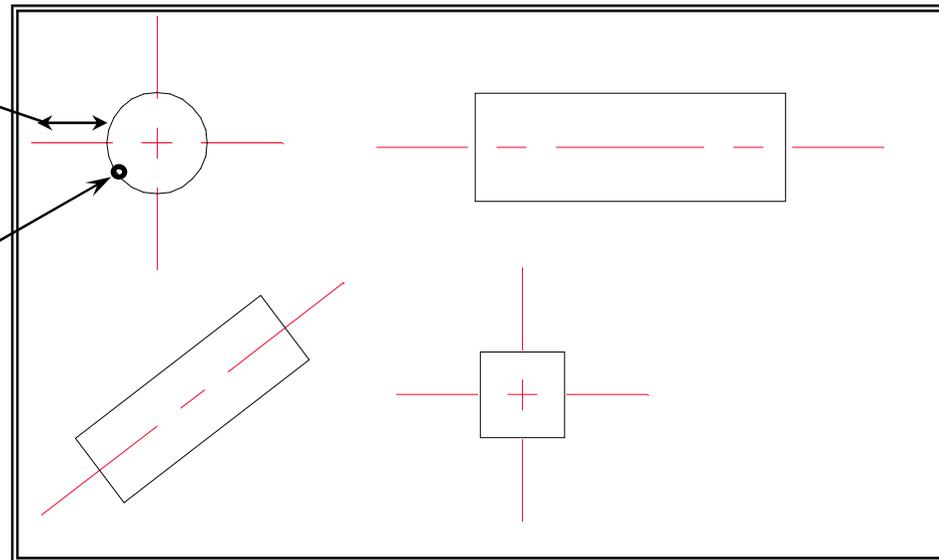
L'exercice consiste à créer une fonction qui **dessine un trait d'axe sur un cercle ou un arc**.

L'analyse sommaire du problème me permet d'identifier les objectifs suivants :

1. Je veux créer un fichier programme (AutoLISP) **AXE.LSP** et la commande **AXE**
2. Je veux sélectionner un cercle ou un arc
3. Je veux choisir la valeur du prolongement de mon axe
4. Je veux gérer automatiquement le **CALQUE** de mon axe

Prolongement de l'axe

Sélectionner le cercle





## AutoLISP travaux pratiques, Exercice N° 3 : Les PHASES

Cette fonction comprend les phases suivantes :



1. Demander à l'utilisateur de sélectionner un arc ou un cercle
2. Pendant la phase 2, boucler tant que l'utilisateur n'a pas choisi un arc ou un cercle, ou stoppé la commande



3. Demander à l'utilisateur la valeur du prolongement de l'axe
4. Calculer les points extrêmes de l'axe
5. Créer le calque AXE avec les caractéristiques Type ligne AXES, Couleur Rouge.
6. Dessiner le trait d'axe sur le calque AXE ( CHPROP Dernier )





## Corrigé Exercice N° 3

```
;;; -----  
;;; Programme   AXE.LSP  
;;; Construction automatique des AXES  
;;; -----  
  
;;; -----  
;;; initialise variables globales  
  
(setq  axe_prol      10      ;;; valeur par défaut du prolongement de l'axe  
       lg_axe       20      ;;; valeur par défaut de la longueur de l'axe  
) ;setq  
  
;;; //////////////////////////////////////  
;;; fonction PRINCIPALE  
  
(defun c:axe( / val sel pl_axe tl_axe cl_axe )  
  
  (graphscr)      ;;; bascule en écran GRAPHIQUE  
  
  (setq          pl_axe "axe"   ;;; nom du calque pour les axes  
          tl_axe "axes"   ;;; type de ligne pour le calque axe  
          cl_axe 1 ;;; couleur du calque axe 1 = rouge  
) ;setq
```

.../...





## Corrigé Exercice N° 3 (suite...)

```
;;; affiche message valeur par défaut pour le prolongement de l'axe
(prompt "\nProlongement de l'axe <" ) (princ (rtos axe_prol 2 2) ) (prompt"> : ")

(setq val (getdist) ) ;;; prolongement de l'axe saisie par l'utilisateur

(if (/= val nil)
    (setq axe_prol val) ;;; affecte la nouvelle valeur si l'utilisateur
                        ;;; a changé la valeur par défaut
);if

;;; -----
;;; sélection de l'ARC ou du CERCLE
;;;
(setq sel nil) ;;; met à nil la sélection SEL pour entrer dans la boucle

(while (= sel nil) ;;; boucle tant que rien n'a été sélectionné

    (setq sel (entsel "\nSélectionner un CERCLE ou un ARC : ") )

    (if (/= sel nil)
        (setq val (entget (car sel) ) ) ;;; test la sélection
    );if
);while
```

.../...





## Corrigé Exercice N° 3 (suite...)

```
;;; -----  
;;; test s'il s'agit d'un ARC ou CERCLE  
;;;  
(if (or (= (cdr (assoc 0 val)) "CIRCLE" )  
        (= (cdr (assoc 0 val)) "ARC" )  
      ) ;or  
    (progn  
      ;;; calcul des points de construction  
      ;;;  
      (setq pt_axe (cdr (assoc 10 val))           ;;; centre CERCLE-ARC  
            lg_axe (* (cdr (assoc 40 val)) 2)    ;;; diamètre CERCLE-ARC  
            pt1_axe (list                        ;;; extrémité N° 1 de l'axe  
                    (- (car pt_axe) (/ lg_axe 2) axe_prol )  
                    (cadr pt_axe)  
                    ) ;list  
            pt2_axe (list                        ;;; extrémité N° 2 de l'axe  
                    (+ (car pt_axe) (/ lg_axe 2) axe_prol )  
                    (cadr pt_axe)  
                    ) ;list  
            ) ;setq  
      .../...
```





## Corrigé Exercice N° 3 (suite...)

```
(sv_param)      ;;; sauvegarde les paramètres
(init_param)    ;;; initialise les paramètres
(test_pl_axe)   ;;; test si le calque existe

;;; dessine le trait d'axe sur le calque AXE
(command "ligne" pt1_axe pt2_axe "")      ;;; dessine la ligne d'axe
(command "reseau" "d" "" "p" pt_axe 2 "90" "o") ;;; ATTENTION à l'angle

(old_param)     ;;; restitue les anciens paramètres

);progn si oui

(prompt "\n\nAucun CERCLE ou ARC sélectionné...")

);if

(princ)

);defun axe
```

.../...





## Corrigé Exercice N° 3 (suite...)

```
;;; -----  
;;; test si PLAN AXE existe  
  
(defun test_pl_axe()  
  
  (if (= (tblsearch "layer" "axe") nil) ;;; test si le calque AXE existe...  
      (command "calque" "e" pl_axe "tl" tl_axe pl_axe "co" cl_axe pl_axe "")  
      (command "calque" "e" pl_axe ""))  
  );if  
  
);defun  
  
;;; -----  
;;; sauve paramètres courants du dessin  
  
(defun sv_param()  
  
  (setq old_osmode (getvar "osmode") ;;; sauvegarde les accrochages objets  
        old_clayer (getvar "clayer") ;;; sauvegarde le calque courant  
        old_cmdecho (getvar "cmdecho") ;;; sauvegarde l'état de CMDECHO  
  );setq  
  
);defun
```

.../...





## Corrigé Exercice N° 3 (FIN)

```
;;; -----  
;;; restaure les paramètres courants du dessin  
  
(defun old_param()  
  
    (setvar "osmode"      old_osmode )  
    (setvar "clayer"      old_clayer )  
    (setvar "cmdecho"     old_cmdecho)  
  
);defun  
  
;;; -----  
;;; initialise les nouveaux paramètres  
  
(defun init_param()  
  
    (setvar "osmode"      0 )  
    (setvar "cmdecho"     0 )  
  
);defun  
  
;;; ***** fin programme *****  
(prompt "\n\nTaper AXE pour construire un AXE ")  
(princ)
```





## Fonctions AutoLISP : Gestion de fichiers externes

Ces fonctions nous permettront de créer, de lire, d'écrire ou d'ajouter des données dans un fichier externe à AutoCAD. (généralement des fichiers textes ASCII).

**OPEN, READ-LINE, CLOSE, PRIN1, PRINC, PRINT, WRITE-LINE**

**OPEN :** Ouverture d'un fichier. Il faut l'assigner à une variable (SETQ) L'assignation correspond à un pointeur utilisé par les fonctions

(open <nom du fichier> <mode> )

**Mode :**

"R" Ouverture en lecture

"W" Ouverture en écriture : création (écrasement si existe)

"A" Ouverture en ajout : création ou ajout en fin de fichier

**Taper la commande : SH DIR \ >TOTO.TXT**

(setq f1 (open "toto.txt" "r" ))      retourne *par exemple* <File: #96eb0>

(setq f2 (open "toto-bis.txt" "r" ))      retourne *nil* (fichier inexistant)





## Fonctions AutoLISP : Gestion de fichiers externes

Ces fonctions nous permettront de créer, de lire, d'écrire ou d'ajouter des données dans un fichier externe à AutoCAD. (généralement des fichiers textes ASCII).

OPEN, **READ-LINE**, CLOSE, PRIN1, PRINC, PRINT, WRITE-LINE

**READ-LINE :** Lit une chaîne de caractère se trouvant dans le fichier ouvert et passe à l'enregistrement suivant. (lecture séquentielle)

(read-line <descripteur fichier> )

*Enregistrement 1* (read-line f1) retourne "" (Enregistrement N°1)

*Enregistrement 2* (read-line f1) retourne " Le volume dans le lecteur C s'appelle..."

*Enregistrement 3* (read-line f1) retourne " Le numéro de série du volume est 1D3B-5664"

*En fin de fichier* (read-line f1) retourne nil

**CLOSE :** Ferme le fichier indiqué par le descripteur (f1). Très important, un fichier ouvert doit toujours être refermé en fin d'utilisation.

(close f1) retourne nil, sinon retourne Erreur : File not open





## Fonctions AutoLISP : Gestion de fichiers externes

Ces fonctions nous permettront de créer, de lire, d'écrire ou d'ajouter des données dans un fichier externe à AutoCAD. (généralement des fichiers textes ASCII).

OPEN, READ-LINE, CLOSE, PRIN1, PRINC, PRINT, WRITE-LINE

**Les fonctions ci-dessous permettent toutes d'écrire dans un fichier**

( <La\_Fonction> <élément\_à\_écrire> <descripteur\_fichier> )

<b><u>Taper</u></b> :	(setq f1 (open "toto.txt" "a" ))	<b>f1 est ouvert en mode ajout</b>
	(prin1 "Fonction PRIN1" f1 )	<b>ajoute la chaîne avec guillemets</b>
	(princ "Fonction PRINC" f1 )	<b>ajoute la chaîne sans guillemets + saut de ligne</b>
	(print "Fonction PRINT" f1 )	<b>ajoute la chaîne avec guillemets + saut de ligne</b>
	(write-line "Fonction WRITE-LINE" f1 )	<b>ajoute la chaîne sans guillemets + saut de ligne</b>





## Fonctions AutoLISP : Gestion de fichiers : Rechercher un fichier

Une fonction de recherche permet de localiser très facilement un fichier ( nom . extension ) en parcourant les répertoires de la variable support ACAD.

### **FINDFILE**

**FINDFILE :** Recherche un nom de fichier en tenant compte des chemins de la variable ACAD

( findfile <unité:\chemin\NOMduFICHIER> )

(findfile "memcalq.lsp")      *retourne* "C:\\FORMTION\\ACAD13\\MEMEPLAN.LSP"

(findfile "memcalq")      *retourne* nil (ce fichier n'existe pas !)

(findfile "acad.mnu")      *retourne* "C:\\ACADR13\\WIN\\SUPPORT\\ACAD.MNU"  
(réponse en fonction de la version d'AutoCAD)





## AutoLISP TP, Exercice N° 4 : Dessin automatique à partir d'un fichier

L'exercice consiste à créer une fonction qui **lit un fichier structuré afin de dessiner automatiquement.**

L'analyse sommaire du problème me permet d'identifier les objectifs suivants :

1. Je veux créer un fichier programme (AutoLISP) **PTCHARGE.LSP** et la commande **PTCH**
2. Je veux dessiner automatiquement des symboles codifiés en lisant le fichier choisi
3. Je veux prendre en compte la codification de mon fichier de point
4. Je veux, en fonction du code, insérer les symboles suivants :

1 = Point type (bloc)



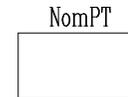
10 = SYMB010 (bloc)



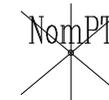
20 = SYMB020 (bloc)



30 = SYMB030 (bloc)



5. Je veux insérer SYMB000 si le point n'est pas codifié :



6. Je veux pouvoir choisir l'échelle de mes symboles
7. Je veux que le nom du Point soit afficher dans mes symboles sous forme d'attribut
8. Je veux entrer le NOM de mon fichier de points
9. Je veux garder toute liberté pour le calque des symboles





## AutoLISP travaux pratiques, Exercice N° 4 : Les PHASES

Cette fonction comprend les phases suivantes :



1. Etudier la structure du fichier de point (exemple : les lignes précédées de 0000 sont des commentaires)
2. Demander à l'utilisateur le nom du fichier de point à lire (NOM + Extension)
3. Demander à l'utilisateur l'échelle des symboles
4. Ouvrir le fichier en lecture
5. Lire chaque enregistrement
6. Insérer en X,Y le bon symbole en fonction de la codification





# TP Exercice N° 4 : Fichier de point

```

0000 champs : X      Y      Z      NomPT      CodePT
0000
0000 code :      1 Point      10 SYMB010
0000              20 SYMB020      30 SYMB030
0000              AUTRES CODES = SYMB000
0000
875008.93331 162191.18536 169.5685225 12 1
875276.50315 162546.53319 169.6107234 321 1
875439.73821 162703.71386 171.4037025 15 1
875132.6144 162214.43936 170.9530742 17 20
875201.23992 162342.76936 169.9252345 19 20
875172.48716 162188.13456 166.8346925 20 10
875226.96854 162183.39709 167.4369961 31 10
875283.80236 162204.01297 167.4202303 40 20
875325.76726 162235.00673 166.4092093 52 30
875381.09808 162262.71275 166.4410259 67 1
875424.61447 162316.76615 167.6803037 37 1
875375.27709 162406.1489 167.6808267 89 50
875510.32425 162406.76619 167.0046504 201 1
875604.46433 162537.51975 166.9398876 202 1
875206.11494 162407.91703 167.6683245 304 1
875254.75909 162479.31302 167.8075099 78 1
875351.8863 162494.83917 167.6480004 22 1
875543.36384 162646.07472 174.044043 315 20
875633.34033 162571.38512 167.3138401 316 20
874978.34062 162198.83265 168.4881345 317 20
874977.27387 162200.19273 167.1428015 318 10
874996.31777 162176.90847 170.7123722 319 10

```

```

874984.06795 162185.96267 169.512936 320 10
874985.24993 162187.95242 169.434724 321 10
874986.92336 162189.05076 169.428729 255 20
874981.28042 162188.80922 168.72225 322 10
874979.84348 162188.95161 169.0149847 323 10
874978.13027 162191.87695 168.8198625 324 10
874978.27103 162198.9658 168.4560761 325 10
875003.8114 162213.12619 168.5841446 326 20
875002.67534 162214.98363 167.1827377 327 10
875009.52333 162211.1796 168.845949 328 10
875012.41614 162209.82594 169.0000311 329 20
875012.98708 162209.31808 168.626895 330 20
875014.06821 162208.61521 169.2801607 331 30
875021.81755 162214.07437 169.3095782 332 20
875023.89934 162214.57505 169.3031403 333 20
875025.52381 162215.65033 169.2998244 334 20
875015.6399 162206.30454 169.4387778 335 1
875019.3037 162200.7514 169.632857 336 1
875020.30028 162199.28781 169.5665181 337 1
875026.1813 162208.02844 169.6299823 338 1
875026.1813 162208.02844 175.86 339 1
875020.78952 162198.09017 169.5386266 340
875021.16221 162197.67694 169.1558962 341
875022.65151 162195.29593 170.6277831 342 1
875017.77437 162190.57862 170.5744973 343 20

```





## Corrigé Exercice N° 4

```
;;; -----  
;;; Programme      PTCHARGE.LSP  
;;; chargement d'un fichier de point au format :  
;;;  
;;;      X      Y      Z      NomPT      CodePT  
;;; -----  
  
;;; -----  
;;; fonction principale  
;;;  
  
(defun c:ptcharge( / fichier_pt ech_symb )  
  
    ;;; -----  
    ;;; Choix du fichier de points  
    ;;;  
    (setq fichier_pt (getfiled "Choix du fichier de POINTS" "." "txt;xyz" 0))
```

.../...





## Corrigé Exercice N° 4 (suite...)

```
;;; -----  
;;; test si fichier non nul, puis saisie de l'échelle  
(if fichier_pt  
  (progn  
    (setq ech_symb (getdist "\nEchelle des symboles <1> : ") )  
    (if (= ech_symb nil)  
        (setq ech_symb 1)      ;;; affecte l'échelle 1 aux blocs  
    ) ;if  
    (charge_pt fichier_pt)      ;;; exécute la fonction CHARGE_PT  
  ) ;progn  
) ;if  
(princ)  
) ;defun  
  
;;; -----  
;;; lecture du fichier  
;;;  
(defun charge_pt ( fichier_pt / f1 enr pt_x pt_y pt_z pt_n pt_c )  
  
  ;;; -----  
  ;;;  
  (setq      f1 (open fichier_pt "r")      ;;; ouvre le fichier en lecture  
            enr (read-line f1)            ;;; lit le 1er enregistrement  
  ) ;setq
```

.../...





## Corrigé Exercice N° 4 (suite...)

```
(sv_param)      ;;; sauvegarde les paramètres courants (fonction SV_PARAM)
(init_param)    ;;; initialise les paramètres courants (fonction INIT_PARAM)

(while enr      ;;; entre dans la boucle tant que ENRegistrement existe

  (if (/= (substr enr 1 3) "000")                ;;; ignore les lignes 000
    (progn
      (setq enr (read(strcat "(" enr ")))        ;;; transforme en LISTE
            pt_x (car enr)                       ;;; Coordonnées X de PT
            pt_y (cadr enr)                      ;;; Coordonnées Y de PT
            pt_z (caddr enr)                    ;;; Coordonnées Z de PT
            pt_n (caddr enr)                    ;;; Nom de PT
            pt_c (caddr (cdr enr))              ;;; Code de PT
      ) ;setq
      (dessin_code pt_x pt_y pt_z pt_n pt_c )    ;;; insère le symbole
    ) ;progn
  ) ;if

  (setq enr (read-line fl) )
) ;while

(old_param)    ;;; restitue les paramètres sauvegardés
) ;defun charge_pt
```

.../...





## Corrigé Exercice N° 4 (suite...)

```
;;; -----  
;;; dessine chaque point  
;;;  
(defun dessin_code( x y z nom code / nombloc )  
  
    (if (= nom nil)                ;;; test si NOM est nil  
        (setq nom "NOM ?")  
    );if  
  
    (if (= code nil)              ;;; test si code est nil  
        (setq code 0)  
    );if  
  
    ;;; -----  
    ;;; formate le CODE en caractères "00." ou "0.."  
    ;;;  
    (if (< code 10)  
        (setq code (strcat "00" (itoa code) ) )  
        (if (< code 100)  
            (setq code (strcat "0" (itoa code) ) )  
        );if  
    );if  
  
    (setq nombloc (strcat "SYMB" code) )           ;;; définit le nom du symbole  
  
    .../...
```





## Corrigé Exercice N° 4 (suite...)

```
;;; -----  
;;; test si le fichier de symbole existe (Wbloc)  
;;;  
(if (= (findfile (strcat nombloc ".dwg" ) ) nil)  
      (setq nombloc "symb000")  
);if  
;;; insere le bloc + ATTRIBUT  
(command "inserer" nombloc (list x y z) ech_symb "" 0 nom)  
  
);defun dessin_code  
  
;;; -----  
;;; sauve paramètres courants  
;;;  
  
(defun sv_param()  
  
      (setq old_osmode (getvar "osmode")      ;;; sauve les accrochages  
            old_blipmode (getvar "blipmode")  ;;; sauve l'état des marques  
            old_clayer (getvar "clayer")      ;;; calque courant  
            old_attdia (getvar "attdia")      ;;; case de dialogue des attributs  
            old_cmdecho (getvar "cmdecho")    ;;; messages des commandes  
);setq  
);defun  
  
.../...
```





## Corrigé Exercice N° 4 (FIN)

```
;;; -----  
;;; restaure les paramètres courants  
(defun old_param()  
  (setvar "osmode"      old_osmode)    ;;; restitue les anciens paramètres  
  (setvar "blipmode"    old_blipmode)  
  (setvar "clayer"      old_clayer)  
  (setvar "attdia"      old_attdia)  
  (setvar "cmdecho"     old_cmdecho)  
  (setq  old_osmode     nil             ;;; annule les variables  
        old_blipmode   nil  
        old_clayer     nil  
        old_attdia     nil  
        old_cmdecho    nil)  
  ) ;setq  
);defun  
;;; -----  
;;; initialise les nouveaux paramètres  
(defun init_param()  
  (setvar "osmode"      0)  
  (setvar "blipmode"    0)  
  (setvar "attdia"      0)  
  (setvar "cmdecho"     0)  
);defun  
;;; ***** fin programme *****  
(prompt "\nTaper PTCHARGE pour charger un fichier de POINTS (.TXT)...")  
(princ)
```





## Fonctions AutoLISP : Accès aux tables du dessin courant

AutoCAD gère des tables pour les éléments suivants : Blocs, Calques, Styles de texte, Styles de cotation, Types de ligne, Vues, Fenêtres multiples, et SCU. Il est possible d'accéder à ces tables pour lire ou rechercher un élément. Il s'agit des fonctions :

**TBLNEXT, TBLSEARCH**

**TBLNEXT :** Affiche le nom d'un élément d'une table

(tblnext <nom de la table> <premier T> )

Nom des tables :

<b>BLOCK</b>	blocs	<b>STYLE</b>	les styles de texte
<b>DIMSTYLE</b>	styles de cotation	<b>VIEW</b>	les vues
<b>LAYER</b>	les calques	<b>VPORT</b>	les fenêtres multiples
<b>LTYPE</b>	les types de lignes	<b>UCS</b>	les SCU

(tblnext "layer" T) *retourne le nom du 1<sup>er</sup> PLAN de la liste (toujours le PLAN 0)*

(tblnext "layer") *retourne le nom du PLAN suivant (créé en N°2)*

(tblnext "layer") *retourne le nom du PLAN suivant (créé en N°3)... etc*





## Fonctions AutoLISP : Accès aux tables du dessin courant

AutoCAD gère des tables pour les éléments suivants : Blocs, Calques, Styles de texte, Styles de cotation, Types de ligne, Vues, Fenêtres multiples, et SCU. Il est possible d'accéder à ces tables pour lire ou rechercher un élément. Il s'agit des fonctions :

TBLNEXT, TBLSEARCH

**TBLSEARCH :** Recherche l'existence d'un élément d'un tableau

(tblsearch <nom du tableau> <nom de l'élément> )

(tblsearch "layer" "0" )      *retourne la liste des caractéristiques de l'élément*

(tblsearch "layer" "cache" )      *retourne la liste des caractéristiques du calque CACHE*

(tblsearch "layer" "toto" )      *retourne nil si le plan n'existe pas*





## Fonctions ADS : Utilisation de fonctions ADS en LISP

En plus de toutes les fonctions AutoLISP existantes, il est possible à tout moment d'utiliser certaines fonctions ADS (langage C). Il en existe une très intéressante qui effectue le tri des données, il s'agit de la fonction :

### ACAD\_STRLSORT

**ACAD\_STRLSORT :** Classe alphabétiquement une liste de chaîne de caractères

( acad\_strlsort <liste alphanumérique> )

*Soit la liste alphanumérique (chaîne de caractères) L1 :*

*(setq L1 ' ( "x" "v" "a" "745.25" "b" "z" "121" "89" )*

*(acad\_strlsort L1)           retourne ( "121" "745.25" "89" "a" "b" "v" "x" "z" )*





## AutoLISP TP, Exercice N° 5 : Trier une table

L'exercice consiste à créer une fonction qui **lit les tables du dessins pour trier alphabétiquement les éléments et envoyer le résultat dans un fichier.**

L'analyse sommaire du problème me permet d'identifier les objectifs suivants :

1. Je veux créer un fichier programme (AutoLISP) **IDXTABLE.LSP** et la commande **IDX**
2. Je veux trier une table dans un fichier
3. Je veux choisir le nom de la table à trier
4. Je veux désigner le fichier à créer





## AutoLISP travaux pratiques, Exercice N° 5 : Les PHASES

Cette fonction comprend les phases suivantes :



1. Proposer à l'utilisateur les tables disponibles
2. Demander à l'utilisateur le nom du fichier à créer
3. Lire la table choisie par l'utilisateur



4. Trier la table
5. Créer le fichier trié
6. Fermeture du fichier





## Corrigé Exercice N° 5

```
;;; -----  
;;; Programme      IDXTABLE.LSP  
;;; Tri une table AutoCAD dans un fichier  
;;; -----  
  
;;; -----  
;;; fonction principale  
;;;  
(defun c:idx( / rep )  
  
  (prompt "\n\t *** Tables AutoCAD à trier dans un fichier ***")  
  (setq rep (getint "\nTable: 1.BLOCS 2.PLANS 3.StyleTXT 4.StyleCOT 5.VUES 6.TypeLIGNE : "))  
  
  (cond  
    ;;; choix de la table  
    ( (= rep 1) (index_table "block"      "BLOCS"      ))  
    ( (= rep 2) (index_table "layer"     "PLANS"      ))  
    ( (= rep 3) (index_table "style"     "Style des TEXTES" ))  
    ( (= rep 4) (index_table "dimstyle"  "Style de COTATION" ))  
    ( (= rep 5) (index_table "view"      "VUES"      ))  
    ( (= rep 6) (index_table "ltype"     "Type de LIGNES"  ))  
    ( T        (prompt "\nAucune table sélectionnée...") )  
  );cond  
  
(princ)  
  
);defun c:idx
```

.../...





## Corrigé Exercice N° 5 (suite...)

```
;;; -----  
;;; lecture et tri de la table  
;;;  
(defun index_table( nomtable nomfr / msg_fic nom_fic nomtable elt_tb num lst_tb )  
  
  (setq elt_tb (tblnext nomtable T)   ;;; lit le 1er élément de la TABLE  
        num    1                       ;;; compteur de boucle  
  
        msg_fic (strcat "Nom du fichier à créer pour la table " nomfr)  
        nomfic (getfiled msg_fic "." "txt" 1)  
  );setq  
  
  (prompt (strcat "\nLecture de la table : " nomfr ))  
  
  (while (and elt_tb  
             nomfic  
           ) ;and  
  
        (setq lst_tb (cons(cdr(assoc 2 elt_tb)) lst_tb) ;;; construit une liste  
              elt_tb (tblnext nomtable)                ;;; lit l'élément suivant  
              num    (1+ num)                          ;;; compteur de boucle  
        ) ;setq  
  );while
```

.../...





## Corrigé Exercice N° 5 (suite...)

```
(if (> num 1)
  (progn
    (setq lst_tb (acad_strlsort lst_tb) )      ;;; tri la liste
    (ecrit_fic nomfic lst_tb nomfr )          ;;; appel création fichier
    (prompt "... création réussie")
  );progn

  (prompt "... Table vide, le fichier est vide...!!!...")
);if

);defun index_table

;;; -----
;;; création du fichier trié
;;;

(defun ecrit_fic ( nomfic lst_tb titre / elt f1 )

  (setq  elt      (car lst_tb)                ;;; lecture du premier élt de liste
        lst_tb   (cdr lst_tb)                ;;; recré la liste
        f1       (open nomfic "w")          ;;; création du fichier (écrasé si existe)
  );setq
```

.../...





## Corrigé Exercice N° 5 (FIN)

```
(write-line "*****" f1)
(write-line "" f1)
(write-line (strcat "      Liste de la table des : " titre) f1)
(write-line "" f1)
(write-line (strcat "      Dessin : " (getvar "dwgname") ) f1)
(write-line "" f1)
(write-line "*****" f1)
(write-line "" f1)
(write-line "" f1)

(while elt
  (write-line (strcat "      " elt) f1)
  (setq elt (car lst_tb) ;; lecture du premier elt de liste
        lst_tb (cdr lst_tb) ;; recré la liste
  );setq
);while

(close f1) ;; fermeture du fichier
);defun escrit_fic

;;; ***** fin programme *****
;;;
(prompt "\nTaper IDX pour trier une table dans un fichier...")
(princ)
```

