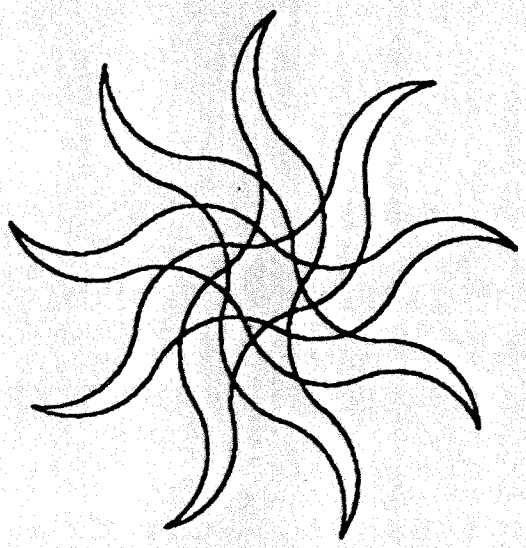
UNIVERSITE LOUIS PASTEUR
I.R.E.M.
10, rue du Général Zimmer
67084 STRASBOURG CEDEX

L O G O

1- LE LANGAGE



SEPTEMBRE 1985

I N T R O D U C T I O N

Ce fascicule a pour but de faciliter une première approche du langage **LOGO**. Il est présenté dans la version **MICRAL** (unique version LOGO disponible à l'époque sur un matériel agréé par l'Education Nationale).

Vous trouverez en annexe, une correspondance des primitives version **APPLE** & **TO7** nécessaire pour utiliser un manuel quelconque concernant le langage LOGO. Cette brochure n'a pas l'ambition de remplacer un tel manuel (cf. bibliographie), mais plutôt d'aider à la compréhension de certaines notions propres au langage LOGO.

Dominique GUIN

N.B. : Ce fascicule est suivi d'un compte-rendu d'une expérience LOGO en CM2. (Brochure IREM de Strasbourg)

Je remercie Marie Agnès EGRET de m'avoir signalé quelques erreurs avant la réédition de cette brochure.

P L A N

1ère Partie - Quelques fiches pour démarrer : (ces fiches étaient destinées à des élèves de CM2, ce qui explique le tutoiement...)

1. Silence on tourne
2. Déplacement de la tortue
3. Comment tracer un cercle ?
4. Repérage dans le plan
5. La tortue devient élève
6. Si tu veux modifier ta procédure...
7. Du carré au cercle ...
8. De la méthode !
9. Gérer son espace de travail
10. Utilisation des disquettes.

2ème Partie - Sortons du graphique !

1. Les objets LOGO : les mots
2. Les objets LOGO : les listes
3. Instruction de sortie : ECRIS
4. Instruction d'affectation : RELIE
5. La primitive CHOSE
6. Remarques sur l'instruction RELIE
7. Former d'autres mots, d'autres phrases
8. Former d'autres listes
9. Manipuler listes et mots
10. Comparons ...
11. Deux primitives utiles concernant les listes
12. Les prédicats manipulant listes et mots
13. Les structures de dialogue
14. Instructions conditionnelles
15. Opérations booléennes et prédicats
16. Créer des procédures

17. Les procédures paramétrées
18. Les procédures fonctions
19. Les variables locales
20. Récursivité
21. Liste de Propriétés

3ème Partie - Annexes

Exercices
Bibliographie
Liste des Primitives (Micral- Apple)

* * *
* *
*

1 - Silence, on tourne...

- Pour charger LOGO :

. Mettre la disquette LOGO dans le lecteur de gauche et taper B puis Ø. Appuyer sur la touche "RL VALIDE". (qui est appelée HOPLA dans la première partie destinée aux élèves).

. La date du jour est alors demandée. Elle n'est acceptée que si elle est correcte. Attendre encore un moment pour que LOGO soit chargé.

Les procédures graphiques sont alors disponibles (commandes de la tortue d'écran cf. 2)

- Pour obtenir les primitives de manipulations de listes et de mots, l'on tape :

  LEXI

- Si plus tard, on ne veut plus travailler avec mots et listes, on peut récupérer la place correspondante, en tapant :

  FINLEXI

- Pour obtenir le graphique, on tape

  TE

- Si l'on veut libérer la place occupée par les procédures graphiques, on tape :

  FINTE

- Si l'on veut utiliser les listes de propriétés (cf. Page 51), on tape PROPRIETES

- Pour libérer la place correspondante, on tape
  FINPROP

  ATTENTION :

  Cette version ne connaît que les nombres entiers compris entre -8191 et +8191 (contrairement à la version APPLE !).

- Pour quitter LOGO et se retrouver sous UTIL (l'utilitaire générale de EDEN), il faut taper AUREVOIR ou CTRL C suivi de la touche HOPLA.

- Pour recharger LOGO quand on est sous UTIL, taper Z (HOPLA). ATTENTION, toutes les procédures de l'espace de travail sont effacées.

2 - Déplacement de la tortue

1) Si tu veux passer au mode graphique, tu tapes TE (cf. fiche 4)

2) Si tu veux faire avancer la tortue, tu tapes
   AV nombre de pas HOPLA (HOPLA = RC = RETURN !)

3) Si tu veux faire reculer la tortue, tu tapes
   RE nombre de pas HOPLA

4) Si tu veux faire tourner la tortue vers la gauche, tu tapes
   TG nombre de degrés HOPLA

5) Si tu veux faire tourner la tortue vers la droite, tu tapes
   TD nombre de degrés HOPLA

6) Si tu veux supprimerle texte de l'écran, tu tapes
   VT HOPLA ou EFFECR

7) Si tu veux faire avancer la tortue sans qu'elle laisse une trace, tu tapes
   LC HOPLA

8) Si tu veux que la tortue laisse de nouveau une trace, tu tapes
   BC HOPLA

9) Si tu veux effacer, tu peux transformer le crayon en gomme, tu tapes
   GOMME HOPLA
   (si tu veux de nouveau une trace, tu tapes BC)

10) Si tu veux effacer l'écran graphique
    (les états de la tortue ne sont pas modifiés), tu tapes :
    NETTOIE HOPLA

3. Comment tracer un cercle ?

1) L'instruction REPETE permet de répéter plusieurs fois l'exécution d'une suite d'instructions

Ex : REPETE 4 [AV 2Ø TD 9Ø]

↑ Nombre de répétitions

↑ Suite d'instructions à répéter entre crochets

produira le tracé d'un carré de longueur 2Ø.

2) Comment tracer un polygone à 12 côtés ?

Pour tracer un polygone régulier, il faut effectuer une rotation totale de 360 degrés. Si nous voulons tracer un polygone régulier à 12 côtés, la rotation à chaque étape doit être de 36Ø/12 = 3Ø.

REPETE 12 [AV 1Ø TD 3Ø]

3) Comment tracer un cercle ?

Il suffit de tracer un polygone régulier ayant suffisamment de côtés pour donner l'illusion du cercle :

REPETE 3Ø [AV 5 TD 12] (par exemple)

Remarque

L'exécution commence à être longue, on a intérêt alors à cacher la tortue CT, on peut la faire réapparaitre grâce à MT.

4. Repérage dans le Plan

1) TE Initialise la tortue graphique en la plaçant à l'origine, c'est-à-dire au centre de l'écran, cap à 0, crayon baissé et tortue montrée.
   les déplacements par rapport à l'origine sont au maximum en abscisse de + 90, en ordonnée de + 61.

2) ORIGINE ramène la tortue à l'origine, fixe son cap à 0

3) CAP rend la valeur du cap de la tortue (en degrés)

   Exemple : ORIGINE
   ECRIS CAP
   ϕ

4) FIXECAP n donne à la tortue n comme nouvelle valeur du cap

5) FIXEXY n m déplace la tortue jusqu'au point d'abscisse n et d'ordonnée m.
   Si n ou m est négatif, mettre le nombre entre parenthèses.

5. La tortue devient élève

La tortue comprend déjà quelques mots. Tu peux lui apprendre d'autres qui feront ensuite partie de son "vocabulaire".

Si tu veux lui apprendre un nouveau mot, tu tapes

POUR ↵ nouveau mot HOPLA (HOPLA = RETURN ou RC)

Tu expliques ce qu'elle doit faire en tapant les ordres successifs. Quand tu as terminé, tu tapes FIN

Exemple :

```
POUR ↵ TOIT      HOPLA
TD   ↵  45       HOPLA
AV   ↵  5Ø       HOPLA
TD   ↵  9Ø       HOPLA
AV   ↵  5Ø       HOPLA
FIN              HOPLA
```

Tu vois apparaître sur l'écran

TOIT EST DEFINI

Le mot TOIT fait désormais partie du "vocabulaire" de la tortue : frappée sur le clavier, TOIT produira sur l'écran le tracé désiré.

Attention

les mot T et NIL ne peuvent pas être utilisés comme nom de procédure ou variable (mots réservés en LISP).

6. Si tu veux modifier la procédure...

1) Il faut rappeler le texte de la procédure. Pour cela, on tape

EDITE "nom de la procédure

En haut de l'écran apparaît

POUR nom de la procédure

texte de la procédure

FIN

2) Si l'on veut modifier la texte de la procédure, il faut placer le curseur à l'endroit où l'on désire modifier quelque chose.
Ceci est possible grâce aux 4 flèches.

Le caractère pointé est détruit par EFF CAR.
On sort de l'éditeur avec prise en compte des modifications effectuées par SORTIE, si l'on ne veut pas garder les modifications effectuées on sort par F3 ou CTRL C.

La touche VALIDE brise la ligne à la position du curseur.

7. Du carré au cercle...

1) Si nous reprenons les programmes de la fiche 3, ils ont beaucoup de points communs. Ne peut-on pas écrire un programme unique ?

```
POUR POLY  :ANGLE
REPETE 36Ø/  :ANGLE  [AV 5 TD  :ANGLE]
FIN
```

POLY 9Ø ⟵ exécution d'un carré

POLY 3Ø ⟵ exécution d'un polygone régulier à 12 côtés

POLY 12 ⟵ approximation d'un cercle

Voici donc une procédure paramétrée. A chaque appel de la procédure, par exemple :

POLY 90

1° 9Ø est placé dans la boîte ANGLE

c'est le contenu de la boîte ANGLE (noté : ANGLE)

ANGLE
9Ø

2° La procédure est exécutée

2) On peut améliorer le programme en introduisant un nouveau paramètre : la longueur du côté du polygone.

```
POUR POLYG : COTE : ANGLE
REPETE 36Ø/:ANGLE  [AV : COTE TD  :ANGLE]
FIN
```

POLYG 1Ø 3Ø

provoque le tracé d'un polygone régulier à 12 côtés de longueur 10.

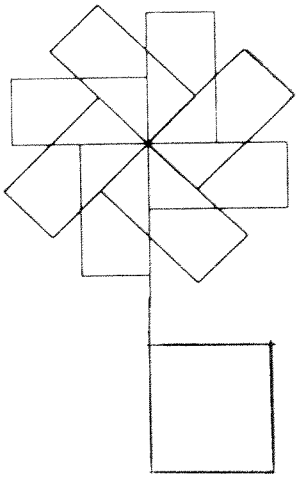
COTE	ANGLE
1Ø	3Ø

Z : Dans l'écriture des procédures paramétrées, ne pas oublier le :

8. De la méthode !

Le gros avantage des procédures c'est de permettre de décomposer les difficultés. Il faut au maximum structurer les projets.

Exemple : Pour réaliser le dessin suivant :



① POUR TRUC
ROSACE
TIGE
CARRE
FIN

② POUR ROSACE
REPETE 8 [RECTANGLE]
FIN

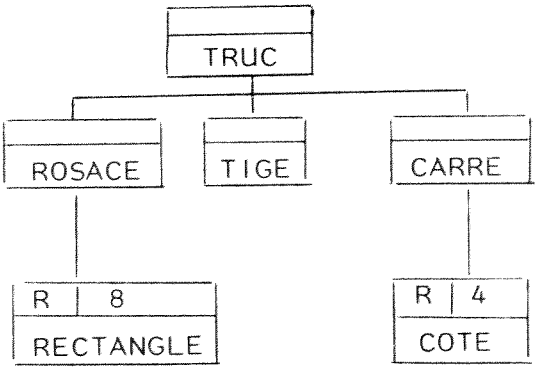
③ POUR RECTANGLE
TD 45
AV 2Ø TD 9Ø AV 1Ø TD 9Ø
AV 2Ø TD 9Ø AV 1Ø
FIN

④ POUR TIGE
TD 18Ø
AV 3Ø
FIN

⑤ POUR CARRE
REPETE 4 [COTE]
FIN

⑥ POUR COTE
AV 2Ø
TG 9Ø
FIN

On peut avoir une vue synthétique du projet grâce à la représentation suivante :



Exercice : réaliser le dessin de la couverture en utilisant la même méthode.

9. Gérer son espace de travail

- Si l'on veut imprimer une définition de procédure, on tape
  IMP "nom de la procédure

- Si l'on veut imprimer toutes les définitions des procédures présentes dans l'espace de travail, on tape
  IMPTOUT

- Si l'on veut imprimer les en-tête de toutes les procédures présentes dans l'espace de travail, on tape
  IMPT

Remarque :

Ces primitives sont bien utiles, quand on utilise l'imprimante (IMPRIMANTE ; on revient à l'écran par ECRAN)

- Si l'on veut effacer une procédure, on tape
  EFF "nom de la procédure

- Si l'on veut effacer plusieurs procédures, on tape
  EFF[nom1 nom2 ...]

- EFFTOUT efface toutes les procédures de l'espace de travail

  EFFTOUT n'efface pas les listes de propriétés (cf. Page 51), il faut écrire des procédures permettant de les effacer.

  Remarque :
  En fait, les noms de procédures sont gardés quelque part, et on arrive vite à un message d'erreur : zone des listes pleines (version MICRAL).

10. Utilisation des disquettes

- En tapant CATALOGUE l'écran affiche les titres du contenu de la disquette

- On peut ramener une procédure sauvée sur disquettes vers l'espace de travail en tapant :
  RAMENE "nom de sauvegarde

- On peut sauver une procédure sur une disquette en tapant
  SAUVE "nom de sauvegarde "nom de la procedure

- On peut sauver plusieurs procédures, en tapant SAUVE "nom de sauvegarde [proc1 proc2 ...]

- On peut sauver tout l'espace de travail en tapant
  SAUVETOUT " nom de sauvegarde

- FIXEDISQUE "a ou "b indique le lecteur sur lequel les prochaines entrées-sorties seront effectuées. (Cette primitive permet d'avoir une disquette langage dans le lecteur de gauche et une disquette fichier dans le lecteur de droite).

ATTENTION

- Un nom de sauvegarde ne peut avoir plus de 8 caractères.
- Il n'est pas possible, dans la version MICRAL, ce sauver des programmes comportant des séquences d'instructions dépassant la longueur d'une ligne (bien qu'ils soient exécutables...)

DEUXIEME PARTIE

1. LES OBJETS LOGO : LES MOTS.

1 _ Un MOT est une suite de caractères (lettres, chiffres, ponctuation) excepté les caractères séparateurs:

<, >, : , \, " , [,], (,) , * , / , + , -

et ESPACE.

Un nombre est un mot d'un type particulier: c'est un mot formé de chiffres uniquement.

Ex: "JEAN 75 "S17 "FRANCOIS.XAVIER sont des mots.

"FRANCOIS-XAVIER n'est pas un mot;

Pour préciser à LOGO qu'il s'agit d'un mot, on le fait précéder par des guillemets, sauf éventuellement s'il s'agit d'un nombre.

Mot vide: "

2. LES OBJETS LOGO : LES LISTES.

2 - Une LISTE est composée d'objets LOGO dont chacun peut être un mot ou une liste. Pour préciser à LOGO qu'il s'agit d'une liste, on l'entoure par des crochets.

Ex: [JEAN A EU 7]

[1 [1 2] [1 7 [1 72]]]

[[JEAN PAUL] A MAL AUX [DENTS]]

Cette dernière liste est formée de 5 objets dont 2 listes.

Liste vide: [ ]

3. INSTRUCTION DE SORTIE : ECRIS.

ECRIS objet (arité 1, arité variable avec parenthèses.)

Remarque : Si c'est un mot, il apparait sans guillemets, si c'est une liste, elle apparait sans les crochets extérieurs.

```
Ex :   ECRIS "A
       A
       ECRIS [ALAIN [A SOIF]]
       ALAIN [A SOIF]
       (ECRIS "C "A)
       C A
```

REMARQUE

On appelle arité d'une primitive le nombre d'arguments de cette primitive.

4. INSTRUCTION D'AFFECTATION : RELIE.

Un mot LOGO peut s'utiliser comme variable. Une variable LOGO est une boîte qui contient un objet LOGO. Cet objet est appelé la VALEUR du mot. (C'est donc un mot ou une liste et pas forcément un nombre).

RELIE mot objet (arité 2)

↑
nom de variable

met la valeur objet dans la boîte nom.

Ex : RELIE "OISEAU "PIGEON

- On donne un nom à une boîte OISEAU .
- Le mot PIGEON est placé dans cette boîte.

Représentation possible de RELIE :

OISEAU
↓
PIGEON

- OISEAU est lié à PIGEON
- OISEAU est une variable
- PIGEON est sa valeur

5. LA PRIMITIVE CHOSE.

Pour connaitre la valeur ou le contenu d'une boîte, on utilise :

CHOSE "OISEAU ou :OISEAU

Cette instruction "produit" la valeur de la variable OISEAU (ici PIGEON).

Remarque : CHOSE ou : devant un mot signifie que le mot doit être considéré comme une variable. Il produit la valeur de cette variable.

Représentation possible de : OISEAU

PIGEON

Ex : ECRIS CHOSE "OISEAU (ou ECRIS :OISEAU)

PIGEON

Remarque :

ECRIS "OISEAU

OISEAU

6- REMARQUES SUR L'INSTRUCTION RELIE.

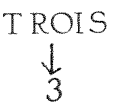
1 - On peut modifier ce qui se trouve dans la boîte, si l'on tape:

```
RELIE "OISEAU  "MOINEAU
```

La boîte OISEAU contient le mot MOINEAU au lieu de PIGEON.

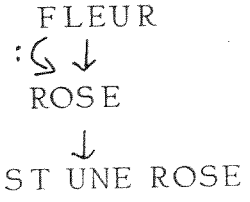
2 - Si l'on tape:

```
RELIE "TROIS 3
ECRIS : TROIS
3
ECRIS "TROIS
TROIS
```



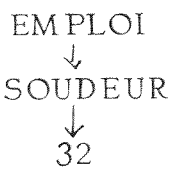
3 - Si l'on tape:

```
RELIE "FLEUR "ROSE
RELIE :FLEUR [C'EST UNE ROSE]
ECRIS :FLEUR
ROSE
ECRIS :ROSE
C'EST UNE ROSE
```



4 - Si l'on tape:

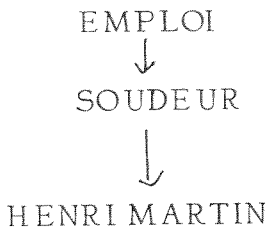
```
RELIE "EMPLOI "SOUDEUR
RELIE "SOUDEUR 32
ECRIS :EMPLOI
SOUDEUR
ECRIS :SOUDEUR
32
ECRIS CHOSE :EMPLOI
32
```



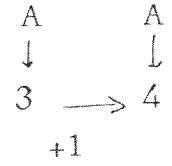

```
RELIE :EMPLOI [HENRI MARTIN]
```

A ce stade, si l'on veut faire le point:

```
ECRIS "EMPLOI ECRIS :EMPLOI
EMPLOI
SOUDEUR
(ECRIS [EMPLOI:] :EMPLOI)
EMPLOI: SOUDEUR
ECRIS CHOSE :EMPLOI
HENRI MARTIN
```



5 -
```
RELIE "A 3 RELIE "A :A+1
ECRIS :A
4
```



7- FORMER D'AUTRES MOTS, D'AUTRES PHRASES.

MOT mot 1 mot 2 (arité 2, arité variable avec parenthèses.)

Les données doivent être des mots.

MOT "produit" un mot qui est la concaténation des mots donnés.

Ex : MOT "BON "JOUR —produit→ BONJOUR

(MOT "BON) ⟶ BON

MOT " "BONJOUR ⟶ BONJOUR

(MOT "BON "J "OUR) ⟶ BONJOUR

PHRASE objet 1 objet 2 (arité 2, arité variable avec parenthèses.)

produit une liste formée de toutes les données.

Ex : PHRASE "A "BU ⟶ [A BU]

PHRASE [A] [BU] ⟶ [A BU]

PHRASE [IL AVAIT] "BU ⟶ [IL AVAIT BU]

(PHRASE "IL) ⟶ [IL]

PHRASE "IL [ ] ⟶ [IL]

ECRIS PHRASE "IL [ ]

IL

8-FORMER D'AUTRES LISTES.

LISTE objet 1 Objet 2 (arité 2, arité variable avec parenthèses.)

produit une liste dont les éléments sont chacune des données.

Ex : LISTE "A "BU ⟶ [A BU]

LISTE [A] [BU] ⟶ [[A] [BU]]

LISTE [IL AVAIT] "BU ⟶ [[IL AVAIT] BU]

LISTE "IL [ ] ⟶ [IL [ ]]

ECRIS LISTE [IL AVAIT BU] [ ]

[IL AVAIT BU] [ ]

COMPARONS LISTE ET PHRASE :

Argument 1	Argument 2	Valeur pour LISTE	Valeur pour PHRASE
"A	"B	[A BU]	[A BU]
[A]	[BU]	[[A] [BU]]	[A BU]
[IL AVAIT]	"BU	[[IL AVAIT] BU]	[IL AVAIT BU]
"IL	[ ]	[IL [ ]]	[IL]

ECRIS LISTE [IL AVAIT] "BU

[IL AVAIT] BU

ECRIS PHRASE [IL AVAIT] "BU

IL AVAIT BU

9 - MANIPULER LISTES ET MOTS.

Voici les principales primitives permettant d'agir sur les objets LOGO, mots ou listes non vides:

SP (Saufpremier) objet (arité 1)

SP "TROIS —produit→ "ROIS

SP [[A] BC D] ⟶ [BC D]

SD (Saufdernier) objet (arité 1)

SD "TROIS ⟶ "TROI

SD [[A] B C] ⟶ [[A] B]

PREM (premier) objet (arité 1)

PREM "TROIS ⟶ "T

PREM [[A] B C] ⟶ [A]

DER (Dernier) objet (arité 1)

DER "TROIS ⟶ "S

DER [[A] B C] ⟶ " C

2 - Voici les principales primitives permettant d'agir sur les LISTES.

METSD (Mets dernier) objet liste (arité 2)

produit une nouvelle liste formée de l'ancienne liste suivie de la première donnée.

METSD "AURORE [ ] ⟶ [AURORE]

METSD "A [B C D] ⟶ [B C D A]

```
ECRIS DER METSD "A [BCD]
A
ECRIS METSD "S [ ]
S
ECRIS METSD " [MANGE]
MANGE
```

METSP (Mets premier) objet liste (arité 2)

produit une nouvelle liste formée de l'ancienne précédée de la première donnée.

```
Ex :  ECRIS METSP [LE UN] [BOL VERRE]
      [LE UN] BOL VERRE
```

10 - COMPARONS ...

primitive	argument 1	argument 2	valeur
METSD	"VACHE	"CHEVAL	erreur
LISTE	"VACHE	"CHEVAL	[VACHE CHEVAL]
PHRASE	"VACHE	"CHEVAL	[VACHE CHEVAL]
METSP	"LOGO	[EST SUPER]	[LOGO EST SUPER]
LISTE	"LOGO	[EST SUPER]	[LOGO [EST SUPER]]
METSD	"LOGO	[EST SUPER]	[EST SUPER LOGO]
PHRASE	"LOGO	[EST SUPER]	[LOGO EST SUPER]
METSP	[LE CHAT]	[REGARDE MEDOR]	[[LE CHAT] REGARDE MEDOR]
LISTE	[LE CHAT]	[REGARDE MEDOR]	[[LE CHAT] [REGARDE MEDOR]]
METSD	[LE CHAT]	[REGARDE MEDOR]	[REGARDE MEDOR [LE CHAT]]
PHRASE	[LE CHAT]	[REGARDE MEDOR]	[LE CHAT REGARDE MEDOR]
METSD	"ORDINATEUR	[ ]	[ORDINATEUR]
LISTE	"ORDINATEUR	[ ]	[ORDINATEUR [ ]]
PHRASE	"ORDINATEUR	[ ]	[ORDINATEUR]

II- DEUX PRIMITIVES UTILES CONCERNANT LES LISTES

ITEM n objet (arité 2)

(n nombre)

- Si l'objet est un mot, produit le n^{ième} caractère.
- Si l'objet est une liste, produit le n^{ième} élément.

Remarque: l'objet doit être non vide.

Ex : RELIE "ANIMAUX [CHIEN CHAT GIRAFE]

ECRIS ITEM 3 :ANIMAUX

GIRAFE

ITEM ϕ "TRUC ——— mot vide

ITEM 8 [A B] ——— liste vide

ITEM 2 [A [B [C]]] ——— [B [C]]

COMPTE objet (arité 1)

- Si l'objet est un mot, produit le nombre de caractères.
- Si l'objet est une liste, produit le nombre d'éléments.

Ex : COMPTE " ——→ ϕ

COMPTE "ABC ——→ 3

COMPTE [ ] ——→ ϕ

COMPTE [A B C] ——→ 3

COMPTE [[VOYEZ [LE]] GEANT] ——→ 2

ECRIS COMPTE [A B C]

3

12-LES PREDICATS MANIPULANT LISTES ET MOTS.

VIDE? objet (arité 1)

produit "VRAI si la donnée est le mot vide ou la liste vide.

```
Ex : ECRIS VIDE? 3
     FAUX
     ECRIS VIDE?[] ECRIS VIDE? "
     VRAI
     VRAI
```

MOT? objet (arité 1)

produit "VRAI si la donnée est un mot, "FAUX sinon.

```
Ex : ECRIS MOT? [UNE FLEUR]
     FAUX
     RELIE "FLEUR "ROSE
     ECRIS MOT? 23  ECRIS MOT? :FLEUR
     VRAI
     VRAI
```

LISTE? objet (arité 1)

produit "VRAI si la donnée est une liste, "FAUX sinon.

```
Ex : LISTE? 3        ------>  "FAUX
     LISTE? [ ]      ------>  "VRAI
     LISTE? "        ------>  "FAUX
     LISTE? [A B C]  ------>  "VRAI
```

CHOSE? mot (arité 1)

produit "VRAI si le mot a une valeur, "FAUX sinon.

Ex : ECRIS CHOSE? "A

FAUX

RELIE "A "FLEUR

ECRIS CHOSE? "A ECRIS CHOSE "A

VRAI

FLEUR

MEMBRE? objet liste (arité 2)

produit "VRAI si la donnée est un élément de la liste,

"FAUX sinon.

Ex : MEMBRE? 3 [2 5 [3] 6] ⟶ "FAUX

MEMBRE? 3 [2 5 3 6] ⟶ "VRAI

MEMBRE? [2 5] [2 5 3 6] ⟶ "FAUX

EGAL? objet 1 objet 2 (ou objet 1 = objet 2) (arité 2)

produit "VRAI si les 2 objets sont identiques, sinon "FAUX.

Ex : ECRIS [LA] = "LA

FAUX

ECRIS EGAL? [ ] "

FAUX

RELIE "M "CHOIX ECRIS CHOSE "M = "CHOIX

VRAI

13 - LES STRUCTURES DE DIALOGUE :

Comment entrer à partir du clavier un objet LOGO?

1 - ENTRER UNE LISTE LOGO

LISLISTE (arité 0)

attend que l'utilisateur tape une ligne et produit cette ligne sous forme de liste.

Ex : 1- ECRIS LISLISTE — quand cette instruction est lue, arrêt du micro, jusqu'à la frappe au clavier.

BONJOUR ⟵ on tape au clavier

BONJOUR — exécution de l'instruction

2- RELIE "A LISLISTE — arrêt du micro

BONJOUR ⟵ on tape au clavier

exécution de l'instruction

ECRIS :A

BONJOUR

A ↓ BONJOUR

3- POUR HELLO

ECRIS [QUEL EST VOTRE NOM?]

ECRIS PHRASE "BONJOUR LISLISTE

FIN

HELLO — exécution de la procédure

QUEL EST VOTRE NOM?

ZOE ⟵ on tape au clavier

BONJOUR ZOE

4 - POUR REMPLISL :QUESTION :TIROIR

ECRIS :QUESTION

RELIE :TIROIR LISLISTE

(ECRIS [TU T'APPELLES] CHOSE :TIROIR)

FIN

REMPLISL [DONNE TON IDENTITE?] "IDENTITE

(appel de la procédure)

DONNE TON IDENTITE?

JEAN DUPONT ← on tape au clavier

TU T'APPELLES JEAN DUPONT

TIROIR
↓
"IDENTITE
↓
LISLISTE : JEAN DUPONT

2 - ENTRER UN MOT LOGO

Si l'on désire entrer un mot LOGO (en particulier un nombre) :

POUR REMPLIS :QUESTION :TIROIR

ECRIS :QUESTION

RELIE :TIROIR PREM LISLISTE

(ECRIS CHOSE :TIROIR)

FIN

REMPLIS [DONNE LA LONGUEUR?] "LONGUEUR

DONNE LA LONGUEUR ← exécution de la procédure

30 ← on tape au clavier

30

LONGUEUR
↓
PREM LISLISTE : 30

LISCAR (arité 0)

attend que l'utilisateur tape un caractère et produit ce caractère.

```
Ex :    RELIE "REP LISCAR
        X      <------  on tape au clavier
        ECRIS :REP                          REP
                                             |
        X                        LISCAR :    X
```

14 - INSTRUCTIONS CONDITIONNELLES :

1 - SI condition ALORS liste d'instructions

(SINON liste d'instructions)

↑ facultatif

Ex : SI 1<2 ALORS [ECRIS "OK] SINON [ECRIS "KO]

OK

POUR DEVINETTE

ECRIS [QUELLE EST LA CAPITALE DU LIECH-

TENSTEIN?]

SI LISLISTE = [VADUZ] ALORS [ECRIS "CORRECT]

SINON [ECRIS "FAUX]

FIN

DEVINETTE ⟵ exécution

QUELLE EST LA CAPITALE DU LIECHTENSTEIN?

SAINT QUENTIN

FAUX

2 - plus pédagogique :

TESTE condition

SIVRAI liste d'instructions 1

SIFAUX liste d'instructions 2

Si la condition est vraie éxécution de la liste 1, sinon éxécution de la liste 2.

Ex :

```
POUR JEU
ECRIS [QUI EST LE MEILLEUR?]
TESTE LISLISTE =[MOI]
SIVRAI [ECRIS [JUSTE]]
SIFAUX [ECRIS [VRAIMENT?]]
FIN
```

```
JEU                éxécution
QUI EST LE MEILLEUR?
JULES        <----  on tape au clavier
VRAIMENT?
```

Remarque : en version MICRAL, il faut souvent ajouter des parenthèses pour que l'évaluation de la condition se fasse correctement, exemple :

```
POUR REMARQUE
RELIE "RADI 789
TESTE (DER :RADI) = "9
SIVRAI [ECRIS [JUSTE]]
SIFAUX [ECRIS [VRAIMENT ?]]
FIN

REMARQUE <---- exécution
JUSTE
```

Attention, pour qu'un test d'égalité rende la valeur VRAI, les deux membres doivent être de même type, c'est-à-dire, deux mots ou deux listes, exemple :

DER :RADI = [9] prend toujours la valeur FAUX puisque le premier membre est un mot, et le second, une liste.

15- OPERATIONS BOOLEENNES ET PREDICATS :

ET booleen 1 booléen 2

OU booléen 1 booléen 2

NON booléen

produisent les valeurs logiques de ces expressions.

(LOGO connait 2 valeurs booléénnes : "VRAI et "FAUX)

Ex : ECRIS "VRAI = OU "VRAI "FAUX

↑

VRAI (attention aux guillemets!)

ECRIS OU MOT? "OUI 3>2

VRAI

ECRIS NON MEMBRE? "4 [5 6]

VRAI

PREDICATS : >, <, =, ≠

16-CREER DES PROCEDURES...

Jusqu'à présent, nous avons analysé les rôles de quelques primitives et nous en avons observé les effets dans des exemples écrits en mode IMMEDIAT.

LOGO possède cette "ambiguité intéressante" de mêler intimement le mode IMMEDIAT et le mode création de PROCEDURE. En effet, l'équivalent de l'écriture d'un programme n'est autre que la création par l'utilisateur d'une nouvelle PRIMITIVE ... Cette nouvelle primitive peut s'adjoindre :

- aux primitives prévues par le constructeur;
- aux primitives déjà crées par l'utilisateur.

Chaque primitive peut être définie par l'emploi des primitives de ces deux natures. Un programme, si sophistiqué soit-il, n'est autre qu'une PRIMITIVE faisant appel à un lot plus ou moins considérable de primitives permettant de "régler" tel ou tel sous-problème. Nul doute que la création de procédures LOGO renforce l'idée de PROGRAMMATION DESCENDANTE, ce dernier concept traduisant la volonté de décomposer un problème complexe en tâches plus élémentaires et ainsi de suite.

Voici un exemple très simple : celui de la conjugaison d'un verbe régulier du premier groupe à la seconde personne du singulier du passé simple.

Ainsi, si l'utilisateur entre SOUPER au clavier, à l'écran doit apparaitre TU SOUPAS.

VOICI une décomposition possible de ce problème :

CONJ
- Entrer un verbe à l'infinitif dans la boîte VERBE
- RADICAL
  - Ecriture d'une procédure qui détermine le radical et le place dans la boîte RAD
- Ecriture de la conjugaison souhaitée.

```
POUR CONJ
ECRIS [DONNE LE VERBE A L'INFINITIF?]
RELIE "VERBE PREM LISLISTE
RADICAL
(ECRIS "TU MOT : RAD "AS)
FIN

POUR RADICAL
RELIE "RAD SD SD :VERBE
FIN
```

17. LES PROCEDURES PARAMETREES :

La procédure CONJ peut être rédigée différemment en autorisant l'entrée du verbe (paramètre) en même temps que le lancement de la procédure :

```
POUR CONJU :VERBE
RADICAL
(ECRIS "TU MOT : RAD "AS)
FIN
CONJU "CHANTER        ← éxécution
TU CHANTAS
```

Cet ordre a pour effet:

1 - Placer le verbe CHANTER dans la boîte VERBE

VERBE
CHANTER

2 - Exécuter la procédure

A chaque fois qu'une procédure paramétrée estéxécutée, il y a SUBSTITUTION de la valeur, ici, CHANTER, au nom du paramètre, ici VERBE. L'argument CHANTER est placé dans la boîte VERBE.

Ex : POUR POLY :COTE :ANGLE

REPETE 36Ø/ :ANGLE [AV :COTE TD :ANGLE]

FIN .

POLY 5Ø 9Ø ← éxécution d'un carré de longueur 50

Les valeurs qu'on choisit comme arguments, par exemple ici 50 et 90, sont placées dans les boîtes indiquées: la première dans la boîte COTE, la deuxième dans la boîte ANGLE

COTE	ANGLE
5Ø	9Ø

Remarque : Si les boîtes des arguments contiennent déjà des objets, lorsqu'on éxécute la procédure, ces objets sont retirés, mais mis de coté. Ils sont remis à la fin de l'éxécution de la procédure. Autrement dit, la procédure emprunte la boîte et lui rend sa condition d'origine, une fois qu'elle a terminé: la variable est LOCALE dans la procédure ou elle se trouve.

Ex :1 - POUR ANIMAL :SON

SI :SON ="MIAOU ALORS [ECRIS "CHAT]

SI :SON = "OUAH ALORS [ECRIS "CHIEN]

SINON[ECRIS [JE NE SAIS PAS]]

FIN

RELIE "SON "GRONDEMENT

ECRIS :SON

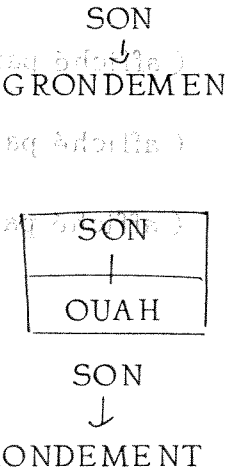
GRONDEMENT

ANIMAL "OUAH

CHIEN

ECRIS :SON

GRONDEMENT



2 - Deux procédures différentes font référence à des noms dans des "bibliothèques " différentes, ce qui implique que les procédures peuvent utiliser les mêmes noms pour différents usages sans aucun conflit.

Ex : POUR DEMO :X

```
ECRIS :X
CHANGER :X
ECRIS :X
FIN
```

DEMO 1 ← éxécution

1 (affiché par DEMO)

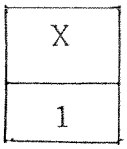
2 (affiché par CHANGER)

1 (affiché par DEMO)

POUR CHANGER :X

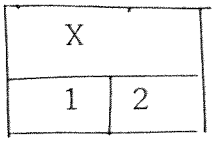
```
RELIE "X :X+1
ECRIS :X
FIN
```

DEMO



Appel ↓

CHANGER



Comparer à :

```
RELIE "X 1
CHANGER :X
2
ECRIS :X
1
```

3 - Un exercice de permutation :

```
POUR PERM1 :A :B
RELIE "C :A
RELIE "A :B
RELIE "B :C
(ECRIS :A :B)
FIN
RELIE "A 3 RELIE "B 5
(ECRIS :A :B)
3 5
PERM1 :A :B
5 3
(ECRIS :A :B)
3 5
```

A	B
↓	↓
3	5

éxécution de la procédure

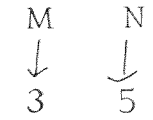
A	B
3	5

A	B	C
3	5	3
5	3	3

Si l'on veut écrire une procédure qui nous retourne la permutation des contenus de A et de B , nous utiliserons des variables LIBRES :

```
POUR PERM :A :B
RELIE "C CHOSE :A
RELIE :A CHOSE :B
RELIE :B :C
FIN
```

```
RELIE "M 3   RELIE "N 5
( ECRIS :M :N )
3 5
PERM "M "N
( ECRIS :M :N )
5 3
```



éxécution de la procédure

A	B
M	N

M (:A)	N (:B)	C
3	5	3
5	3	3

REMARQUE :

POUR PERM2 :A :B

```
RELIE "C :A
RELIE "A :B
RELIE "B :C
(ECRIS [A :] :A [B :] :B)
FIN
```

```
PERM2 5 6
A : 5 B : 6
(ECRIS :A)
```

éxécution de la procédure

IL N'y a pas de valeur pour A.

18- PROCEDURES FONCTIONS :

Toute la puissance d'une procédure provient du fait qu'elle peut devenir un des éléments de base d'une définition de procédure beaucoup plus complexe. Il faut arriver à ce qu 'une procédure non seulement effectue une opération mathématique quelconque, mais rends aussi le résultat de cette opération accessible à d'autres procédures. On y parvient grâce à :

RENDS objet ou résultat d'opération (arité 1)
provoque la sortie de la procédure et ramène le résultat à la procédure appelante.

Ex :

```
POUR CARRE :X
RENDS :X*:X
FIN

ECRIS CARRE 3
9
3Ø.25
```

```
POUR MOYENNE :X :Y
RENDS (:X+:Y)/2
FIN

ECRIS CARRE (MOYENNE 5 6 )
```

```
ECRIS MOYENNE ( CARRE 5 ) ( CARRE 6 )
3Ø.5
```

```
POUR ABS :X
SI :X<Ø ALORS [RENDS -(:X)]
SINON [RENDS :X]
FIN
```

On pourra utiliser l'opération ABS dans un programme quelconque.

PROCEDURE AVEC RESULTAT BOOLEEN :

```
POUR TEST
ECRIS [TAPE 3]
SI LISLISTE = [3] ALORS [RENDS "VRAI]
FIN

TEST
TAPE 3
3 ←—— On tape au clavier

ECRIS TEST

VRAI
```

19. VARIABLES LOCALES.

LOCAL nom 1 nom 2 (arité variable)

ne peut être utilisé que dans une procédure. Les données sont des noms de variable qui seront déclarées locales à la procédure.

Ex : 1 - POUR PERM :A :B

```
LOCAL "C
RELIE "C CHOSE :A
RELIE :A CHOSE :B
RELIE :B :C
FIN
```

2 - POUR OUINON :QUESTION

```
LOCAL "REPONSE
ECRIS :QUESTION
RELIE "REPONSE PREM LISLISTE
SI :REPONSE = "OUI ALORS [RENDS "VRAI]
SINON [RENDS "FAUX]
FIN
```

POUR SALUER

ECRIS [QUEL EST VOTRE NOM COMPLET?]

RELIE "REPONSE LISLISTE

SI OUINON [AIMEZ-VOUS VOTRE NOM?] ALORS

[ECRIS [C'EST BIEN]]SINON [ECRIS [C'EST DOMMAGE]]

(ECRIS HEUREUX DE VOUS CONNAITRE, :REPONSE)

FIN

SALUER ← éxécution

QUEL EST VOTRE NOM COMPLET? REPONSE ↓

HERMAN NIXON ← on tape au clavier HERMAN NIXON

AIMEZ-VOUS VOTRE NOM? QUESTION ↓ AIMEZ-VOUS VOTRE NOM?

NON on tape au clavier REPONSE ↓ NON

(sortie de OUINON)

C'EST DOMMAGE REPONSE ↓ HERMAN NIXON

HEUREUX DE VOUS CONNAITRE, HERMAN NIXON

Si REPONSE n'était pas déclaré comme variable locale dans la procédure OUINON, on aurait perdu la valeur HERMAN NIXON de REPONSE dans SALUER.

20. RECURSIVITE

Définition

Une procéfure récursive est une procédure qui s'appelle elle-même.

1. Des procédures qui se répètent à l'infini

Exemples

① POUR COMPTEUR: Nombre
ECRIS : NOMBRE
COMPTEUR : NOMBRE - 1 ⟵ Appel récursif
FIN

Appel de la procédure : COMPTEUR 10

NOMBRE
10

début d'exécution → 10
Appel récursif → COMPTEUR : NOMBRE - 1
de COMPTEUR

NOMBRE
9

suite de l'exécution → 9
Appel récursif → COMPTEUR : NOMBRE - 1
de COMPTEUR

NOMBRE
8

suite de l'exécution → 8
etc...

Donc l'exécution de COMPTEUR 10 affiche 10 puis 9, 8, 7,..., 1, 0, - 1, ... etc... jusqu'à ce qu'on arrête l'exécution par la commande CTRL B.

② POUR POLYSPI : COTE : ANGLE
AV : COTE
TD : ANGLE
POLYSPI : COTE + 3 : ANGLE ⟵ Appel récursif
FIN

Appel de la procédure : POLYSPI 50 90

début d'exécution ⟶

Appel récursif ⟶ POLYSPI : COTE + 3 : ANGLE

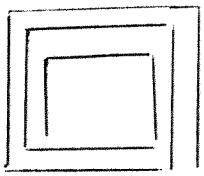
COTE	ANGLE
53	90

Suite de l'exécution ⟶

Appel récursif ⟶ POLYSPI : COTE + 3 : ANGLE

COTE	ANGLE
56	90

etc... on obtient un dessin de spirale carrée



dont on arrête l'exécution par la commande

CTRL B

2. COMMANDES CONDITIONNELLES

Il est possible et même très fortement conseillé de réécrire ces 2 procédures, en ajoutant une instruction conditionnelle de manière à contrôler exactement l'exécution.

```
POUR COMPTEUR : NOMBRE
SI : NOMBRE = 0  ALORS  [STOP]
ECRIS : NOMBRE
COMPTEUR : NOMBRE - 1
FIN
```

En réponse à la commande COMPTEUR 3, l'ordinateur affichera 3, 2, 1.

```
POUR POLYSPI : COTE : ANGLE
SI : COTE > 150 ALORS [STOP]
AV : COTE
TD : ANGLE
POLYSPI : COTE + 3 : ANGLE
FIN
```

l'exécution de l'instruction POLYSPI 50 90 se traduira par un dessin d'une spirale carrée jusqu'à ce que la valeur de COTE dépasse 150.

Voici un dernier exemple de procédure récursive

```
POUR INV :M
TESTE VIDE? :M
SIVRAI [RENDS "]
SIFAUX [RENDS MOT DER :M INV SD :M]
FIN
ECRIS INV "1234
4321
ECRIS PHRASE INV "ELIME [BOIT TROP]
EMILE BOIT TROP
```

Remarque : Ne pas oublier les tests d'arrêt, sinon la procédure se répète à l'infini...

LISTE DE PROPRIETES :

A tout mot LOGO, est associé une liste de propriétés (P - liste). A chaque propriété, est associée une valeur. La P - liste est de la forme suivante :

[prop 1 [val 1] prop 2 [val 2] ..... prop N [val N]]

Les primitives suivantes permettent de manipuler les listes de propriétés.

DPROP (donne propriété) mot 1 mot 2 liste (arité 3)
↑ nom de la propriété
↑ valeur de la propriété

Ex : DPROP "PIN "FAMILLE "CONIF
(P - liste de PIN : FAMILLE [CONIF])

PLISTE mot (arité 1)
produit la liste de propriétés associée au mot :

```
Ex :   ECRIS PLISTE "PIN
       FAMILLE [CONIF]
1 -    DPROP "PIN "FRUIT [POMME DE PIN]
       ECRIS PLISTE "PIN
       FRUIT [POMME DE PIN] FAMILLE [CONIF]
```

2 - DPROP "PIN "FAMILLE [CONIFERE]

ECRIS PLISTE "PIN

FRUIT [POMME DE PIN] FAMILLE [CONIFERE]

Remarques pour DPROP :

1 - rajoute la nouvelle propriété en tête de la liste

2 - si la propriété existait déjà, sa valeur est remplacée par la nouvelle valeur.

RPROP (ramène propriété) mot 1 mot 2 (arité 2)
↑
nom de la propriété

ex : ECRIS RPROP "PIN "FAMILLE

CONIFERE

APROP (annule propriété) mot 1 mot 2 (arité 2)
↑
nom de la propriété

ex : APROP "PIN "FRUIT

ECRIS PLISTE "PIN

FAMILLE [CONIFERE]

Remarque : EFFTOUT n'efface pas les listes de propriétés. Il faut écrire des procédures utilisant APROP pour effeacer les P- listes.

EXERCICES.

1 - On se donne un groupe sujet (GS), un groupe verbe (GV) et un adverbe : on veut faire écrire la phrase.

Ex : GS ↓ LE PARAPLUIE — GV ↓ MANQUE — AD ↓ CRUELLEMENT

2 - Faire écrire la phrase :

LE CHIEN MANGE LE CHAT

en utilisant uniquement les variables A, R et P où :

A ↓ LE — R ↓ CHIEN — P ↓ MANGE ↓ CHAT

3 - En utilisant des variables GS , GV , GO , faire écrire :

[IL] [A MANGE] [LA POMME]

4 - On entre un verbe à l'infinitif au clavier (par exemple MANGER), on veut faire écrire la 2ème personne du présent :

TU MANGES

5 - On entre au clavier une phrase d'au moins 3 mots. Faire écrire le 3ème mot, le nombre de mots, et chaque mot de la phrase.

6 - Faire la moyenne de 6 nombres en entrant successivement chaque nombre au clavier.

7 - Même question ,en entrant une liste non vide de nombres.

8 - On entre au clavier un mot ou une phrase Faire écrire :

- Si l'on n'a rien tapé
- Si c'est un mot ou une phrase

9 - ECRIRE une procédure qui calcule la longueur d'une liste éventuellement vide.

10 - Ecrire une procédure qui retourne si un mot est un palindrome.

11 - Ecrire une procédure qui épelle un mot.

12 - ECrire une procédure qui écrit tous les mots d'une liste.

13 - Ecrire une procédure qui enlève la première et la dernière lettre d'un mot.

14 - Ecrire une procédure qui retourne si un nombre est pair.

15 - Ecrire une procédure qui retourne si un nombre est inférieur ou égal à un autre.

16 - Ecrire une procédure qui incrémente le contenu numérique d'une variable de 1.

17 - Ecrire une procédure qui retourne si un caractère donné apparait dans un mot donné.

18 - Ecrire une procédure pour coder un mot (on translate chaque caractère de 3 lettres : A-->D, B --->E, etc...)

19 - Ecrire une procédure du type :

TANTQUE :CONDITION :LISTE.D.INSTRUCTIONS.

20 - Ecrire une procédure qui calcule l'occurence d'un élément d'une liste.

21 - DESSIN D'UN ARBRE BINAIRE.

22 - ETABLIR UNE MINI BASE DE DONNEES.

- D O C U M E N T A T I O N -

- Seymour PAPERT : LE JAILLISSEMENT DE L'ESPRIT - Ed Flammarion 1981.

- Gérard BOSSUET : L'ORDINATEUR A L'ECOLE - Ed P.U.F. 1982 .

- Harold ABELSON : LE LOGO SUR APPLE - Ed Cedic/Nathan 1984

- Revue EDUCATION ET INFORMATIQUE.

- Bande VHS "Préparation à LOGO" Albert HUBER
(Ecole Normale Strasbourg - IREM Strasbourg)

- LOGO - MANUEL DE REFERENCE - Ed Cédic/Nathan 1984

- INITIATION A LOGO - Ed Cédic/Nathan 1984

- Michel BOURBION : L'ALTERNATIVE LOGO - Ed. Armand Colin 1985

- Brochure I.R.E.M. Strasbourg : LOGO N° 2 - une expérimentation en CM 2.

- WEIDENFELD - BRUILLARD - PEROLAT : Aller plus loin en LOGO- Eyrolles 1985.

- MYX André : Plus loin avec LOGO - Cédic-Nathan 1985.

AIDE-MEMOIRE DES PRIMITIVES LOGO

(Brochure N° 1 LOGO - I.R.E.M. de Strasbourg)

MICRAL 80-22 G **APPLE II** M 0 5 /T07

APROP nom prop	52	ANNULEPROP nom prop	
Annule la propriété prop du nom			
AV n	5	AV n	AV n
Déplace la tortue de n pas vers l'avant			
BC	5	BC	BC
Abaisse le crayon			
CT	6	CT	CT
Rend la tortue invisible			
CAP	7	CAP	CAP
Retourne le cap de la tortue			
CARAC n		CAR n	CAR n
Retourne le caractère dont le code ASCII est n			
CATALOGUE	13	CATALOGUE	CATALOGUE
Affiche les noms de tous les fichiers de la disquette			
CHOSE mot	18	CHOSE mot	CHOSE mot
Retourne le contenu de l'objet désigné par mot			

CHOSE ? mot	28		

Retourne vrai si le mot a un contenu ou une valeur

CODE car		ASCII car	ASII car

Retourne le code ASCII du caractère

COMPTE liste ou mot	26	COMPTE liste	COMPTE liste

Retourne le nombre d'éléments dans la liste (le nbre de caractères pour un mot en version MICRAL)

COPIEDEF nouvmot		COPIEDEF nouvmot mot	

Copie la définition de mot dans nouvmot

DEFINIS mot liste		DEFINIS mot liste	

Donne liste comme définition de mot

DEFINI ? mot		DEFINIP mot	

Retourne vrai si mot est le nom d'une procédure, sinon faux

DER objet	23	DERNIER objet	DER Objet

Retourne le dernier élément d'objet

DPROP mot prop objet	51	DPROP mot prop objet	

Donne à mot la propriété prop avec la valeur objet

ECRAN	12	IMPRIMANTE numéro de port	SORTIE numéro de port

Dirige toutes les impressions vers l'écran

ECRIS objet	16	ECRIS (EC) Objet	ECRIS Objet

Ecris objet (sans [ ] et guillemets) et passe à la ligne suivante

EDITE mot	9	EDITE (ED) mot	ED liste

Edite la procédure de nom mot

EGAL ? objet 1 objet 2	28	EGALP objet 1 objet 2	EGAL ? ob 1 ob 2
Retourne vrai si les 2 entrées sont identiques			
EFF mot ou liste	12	EFFACE (EF) mot ou liste	EFP
Efface les procédures nommées de l'espace du travail			
EFFTOUT	12	EFTOUT	EFT
Efface toutes les procédures de l'espace de travail			
ET pred 1 pred 2		ET pred 1 pred 2	ET pred 1 pred 2
Retourne vrai si les 2 entrées sont vraies, sinon faux			
FAIS liste d'instructions		EXECUTE liste d'instructions	EXEC liste d'instructions
Exécute la liste d'instructions			
FIN	8	FIN	FIN
Indique la fin d'une procédure			
FINLEXI	4		
Libère la place occupée par le travail sur listes et mots			
FINTE	4		
Libère la place occupée par les procédures graphiques			
FIXECAP n	7	FCAP n	FCAP n
Donne au cap dela tortue la valeur n degrés			
FIXECURSEUR pos		FCURSEUR pos	FCURS pos
Place le curseur à la position pos : [numéro de ligne numéro de colonne] (en mode texte)			
FIXEDISQUE "a ou "b	13	FDISQUE numéro d'accès	FLECTEUR numéro d'accès
indique le lecteur sur lequel s'effectuent entrées-sorties			
FIXEXY nombre 1 nombre 2	7	FPOS pos	FPOS pos
Place la tortue à l'abscisse nombre 1, à l'ordonnée nombre 2 ou à la position pos : [abscisse ordonnée]			

GOMME	5	GC	

Transforme le crayon de la tortue en gomme à effacer

HASARD		HASARD n	HASARD n

Rend un nombre aléatoire compris entre 0 et 99

Retourne un entier aléatoire non négatif inférieur à n

IMP mot	12	IM mot	IM mot

Imprime la définition de la procédure de nom mot

IMPT	12	IMTS	IMTS

Imprime les lignes titres des procédures de l'espace de travail

IMPTOUT	12	IMTOUT	IMTOUT

Imprime les définitions des procédures de l'espace de travail

IMPRIMANTE	12	IMPRIMANTE numéro de port	SORTIE numéro de port

Oriente l'écriture vers l'imprimante au lieu de l'écran

ITEM n liste ou mot	26	ITEM n liste	ITEM n liste

Retourne le $n^{ième}$ élément du mot ou de la liste

LC	5	LC	LC

Lève le crayon

LEXI	4		

Pour obtenir les primitives de manipulations de listes et de mots

LISCAR	31	LISCAR	LISCAR

Retourne le caractère tapé par l'usager (attend si nécessaire)

LISLISTE	29	LISLISTE (LL)	LL

Retourne la ligne tapée par l'usager (attend si nécessaire)

LISTE objet 1 objet 2	22	LISTE objet 1 objet 2	Liste objet 1 objet 2

Retourne la liste formée de ses entrées

LISTE ? objet	27	LISTEP objet	LISTE ? objet

Retourne vrai si objet est une liste,
faux sinon

LOCAL mot	46	LOCAL mot	

Rend la variable de nom mot locale

MEMBRE ? objet liste	28	MEMBREP objet liste	MEMBRE ? Objet liste

Retourne vrai si objet est un élément
de la liste

METSD objet liste	24	METF objet liste	MD Objet Liste

Retourne la liste formée en plaçant objet
comme dernier élément de liste

METSP objet liste	24	METD objet liste	MP Objet liste

Retourne la liste formée en plaçant objet
comme premier élément de liste

MONTRE objet		MONTRE objet	

Ecrit objet et passe à la ligne suivante

MOT mot 1 mot 2	21	MOT mot 1 mot 2	MOT mot 1 mot 2

Retourne le mot concaténation des
2 entrées

MOT ? objet	27	MOTP Objet	MOT ? Objet

Retourne vrai si objet est un mot,
faux sinon

MT	6	MT	MT

Rend la tortue visible

NETTOIE		NETTOIE	NETTOIE

Efface l'écran graphique sans modifier la
position de la tortue

NOMBRE ? objet		NOMBREP objet	NOMBRE ? objet
Retourne vrai si objet est un nombre, faux sinon			
NON prédicat	34	NON prédicat	NON prédicat
Retourne vrai si prédicat est faux, faux si prédicat est vrai			
ORIGINE	7	CENTRE	ORIGINE
Place la tortue à [0 0] et fixe le cap à 0			
OU predicat 1 prédicat 2	34	OU prédicat 1 predicat 2	OU pred 1 pred 2
Retourne faux si les 2 entrées sont fausses, vrai sinon.			
PHRASE objet 1 objet 2	21	PHRASE (PH) objet 1 objet 2	PHRASE Objet 1 Objet 2
Retourne la liste formée de objet 1 et objet 2			
PLISTE mot	51	PLISTE mot	
Retourne la liste de propriétés de mot			
PLUSGRAND ? mot 1 mot 2 PLUSPETIT ? mot 1 mot 2			
Ces opérations effectuent des comparaisons numériques ou alphanumériques suivant le type des données.			
POUR mot	8	POUR mot	POUR mot
Commence la définition de la procédure de nom mot			
PREM objet	23	PREMIER objet	PREM objet
Retourne le premier élément d'objet			
PRIMITIVES			
Affiche sur l'écran la liste des primitives			
PRIMITIVE ? mot		PRIMITIVEP mot	PRIM? mot
Retourne vrai si mot est une primitive, faux sinon			

PRODUIT a b		PRODUIT a b	PROD a b
Retourne le produit de ses entrées			
QUOTIENT a b		QUOTIENT a b	QUOT a b
Retourne la partie entière de $\frac{a}{b}$			
RAMENE mot	13	RAMENE mot	RAMENE mot
Charge dans l'espace de travail les définitions contenues dans le fichier			
RE n	5	RE n	RE n
Déplace la tortue de n pas vers l'arrière			
RECYCLE		RECYCLE	RECYCLE
Effectue une récupération de la mémoire disponible			
RELIE mot objet	17	DONNE mot objet	DONNE mot objet
Donne la valeur objet à mot			
RENDS objet	44	RETOURNE, RT objet	RENDS objet
Retourne le contrôle à la procédure appelante avec objet comme produit			
REPETE n liste	6	REPETE n liste	REPETE n liste
Exécute liste n fois			
RPROP mot prop	52	RPROP mot prop	
Retourne la propriété prop de mot			
SAUVE mot objet	13		SAUVE mot objet
Sauve dans le fichier de nom mot, les procédures dont le nom figure dans objet			
SAUVETOUT	13	SAUVE mot	SAUVED mot
Sauve dans le fichier de nom mot, toutes les procédures présentes dans l'espace de travail			

SD objet	23	SD objet	SD objet

Retourne tout objet sauf son dernier élément

SI pred alors liste 1(sinon liste 2)	32	SI pred liste 1 (liste 2)	SI pred liste 1 (liste 2)

Si pred est vrai, exécute liste 1 ; sinon liste 2 (s'il y a lieu)

SIFAUX liste	33	SIFAUX,SIF liste	

Exécute liste si le plus récent TESTE (TEST version APPLE) était faux

SIVRAI liste	33	SIVRAI, SIV liste	

Exécute liste si le plus récent TESTE (TEST version APPLE) était vrai

SOMME a b		SOMME a b	SOMME a b

Retourne la somme de ses entrées

SP objet	23	SP objet	SP objet

Retourne tout objet sauf son premier élément

STOP	49	STOP	STOP

Arrête la procédure et retourne le contrôle à la procédure appelante

TAPE objet		TAPE objet	TAPE Objet

Ecrit objet sans crochets sans passer à la ligne suivante

TD n	5	DR n	TD n

Fait tourner la tortue de n degrés vers la droite

TE	7		

Passage au mode graphique (MICRAL) Initialise la tortue à la position [0,0] avec cap 0, crayon baissé et tortue montrée

TESTE predicat	33	TEST predicat	

Vérifie si prédicat est vrai ou faux

TEXTE mot		TEXTE mot	

Retourne la définition de la procédure mot sous forme de liste

TG n	5	GA n	TG n

Fait tourner la tortue de n degrés vers la gauche

VIDE ? objet	27	VIDEP objet	VIDE ? objet

Retourne vrai si obj est le mot de la liste vide ; sinon, retourne faux

VISIBLE ?		VISIBLEP	VISIBLE ?

Retourne vrai si la tortue est visible, faux si elle ne l'est pas

VT		VT	VT

Vide l'écran de texte