LOGO - MANUEL DE RÉFÉRENCE

Doris Avram - Tristan Savatier - Michèle Weidenfeld
et l'équipe de S.O.L.I.

Illustration de couverture : Nadine Monnier

TO TEK International
cedic nathan

Dans la même collection

Initiation au BASIC TO7/TO7-70 — *Christine et François-Marie Blondel.*
Le BASIC D.C.S. du TO7/TO7-70 et du MO5 — *Christine et François-Marie Blondel.*
Un ordinateur à la maison — *Jean Delcourt.*
Un ordinateur en fête — *Serge Pouts-Lajus.*
Un ordinateur et des jeux — *Jean-Pascal Duclos.*
Guide pratique de l'ordinateur personnel d'IBM — *Dalloz/Emery/Portefaix/Boisgontier/Salzman.*
LOGO, des ailes pour l'esprit — *Horacio C. Reggini.*
Premiers pas avec le ZX-SPECTRUM — *Ian Stewart/Robin Jones.*
Le langage machine de ZX-SPECTRUM — *Ian Stewart/Robin Jones.*
Plus loin avec le ZX-SPECTRUM — *Ian Stewart/Robin Jones.*
Jeux vidéo, jeux de demain — *Georges-Marie Becherraz/Alain Graber.*
Guide pratique de l'Oric-Atmos — *Michel Bussac/Robert Lagoutte.*
Des programmes pour votre Oric-Atmos — *Michel Piot.*
Premiers pas avec le Commodore 64 — *Ian Stewart/Robin Jones.*
Initiation à LOGO — *Doris Avram/Michèle Weidenfeld.*
LOGO, Manuel de référence — *Doris Avram/Tristan Savatier/Michèle Weidenfeld et l'équipe de S.O.L.I.*
Guide du MO5 — *André Deledicq.*
Faites vos jeux en assembleur sur TO7 et TO7-70 — *Michel Oury.*
Manuel de l'assembleur du 6809 et du TO7 et TO7-70 — *Michel Weissgerber.*
Manuel de l'assembleur du 6809 du MO5 — *Michel Weissgerber.*
Initiation au FORTH — *S.E.F.I.*
Guide pratique du vidéotex et du minitel — *Jean-Pierre Saboureau/Geneviève Bouché.*
Guide pratique de l'enseignement assisté par ordinateur — *Jean-Michel Lefèvre.*
La face cachée du TO7/TO7-70 — *Jean-Baptiste Touchard.*
Manuel technique du TO7/TO7-70 — *Michel Oury.*
Manuel technique du MO5 — *Michel Oury.*

Ce volume porte la référence
ISBN 2-7124-0533-1

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS

Ce manuel a été rédigé par :
Doris Avram, Tristan Savatier et Michèle Weidenfeld
sous la direction de :
Gérard Dahan
avec la participation de :
Sylvain Angeli et Marie Paule Deprund
avec l'aide de :
Anahid Derian, Evelina Rosenfeld et toute l'équipe de S.O.L.I. - Paris.

L'implantation du langage Logo a été réalisée sous la direction de :
Gérard Dahan
par :
Sylvain Angeli, avec la collaboration de Marie-Paule Deprund, Jean-Baptiste Touchard, Christian Jullien et l'équipe de S.O.L.I. — Paris.
Nous tenons à remercier l'équipe de Thomson-Simiv et plus particulièrement Bernard Dion et Tristan Savatier pour leur coopération et leurs précieux conseils.

Table

Préface

Le manuel de référence Logo vous sera utile dans votre activité de programmation.

Nous supposons que vous avez déjà une expérience de programmation en Logo, sinon, nous vous conseillons de commencer par le manuel d'initiation au langage Logo.

La première partie du manuel de référence contient une présentation concise de la structure du langage et de la "grammaire" Logo. Dans la deuxième partie, les primitives Logo (termes initiaux du langage) sont décrites en détail et illustrées par des exemples.

En annexe, vous trouverez des indications utiles sur les messages Logo, les codes ASCII, la liste alphabétique des primitives, leur présentation par chapitres, les procédures définies dans le manuel, etc.

Mise en route

Pour utiliser Logo avec votre ordinateur :

1. Mettez la cartouche Logo dans la trappe qui se trouve sur la gauche du clavier.

2. Allumez le téléviseur, connectez et allumez les périphériques si vous en avez, allumez l'ordinateur.

3. Si vous avez un MO5, Logo est alors à votre disposition. Sinon, choisissez [1] pour Logo.
En haut de l'écran apparaît :

```
LOGO 1.Ø (c) SOLI 1984
?_
```

Ce message[1] indique que Logo est chargé en mémoire. Le point d'interrogation (?) au début de ligne est le symbole d'invite Logo : il signifie que l'ordinateur attend vos instructions. Le trait qui clignote (—) est le curseur. Il indique la position du prochain caractère tapé.

4. Si vous voulez utiliser les lecteurs de disquettes, il faudra procéder comme suit :

MO5

Après l'étape 3, mettre la disquette DOS dans le lecteur 0 et faire .EFT (attention .EFT réinitialise Logo).

TO7 et TO7-70

Avant de choisir [1], à l'étape 3, mettre la disquette DOS dans le lecteur 0.

5. Si vous souhaitez démarrer Logo en ramenant automatiquement un fichier, reportez-vous à l'annexe C.

---

(1) Si vous n'obtenez pas ce message :
— éteignez tout ;
— vérifiez que chaque appareil est bien connecté ;
— répétez les instructions pour charger le langage.
Si vous n'obtenez toujours rien... consultez votre distributeur.

Le clavier

Le clavier comporte des touches alphanumériques comme une machine à écrire et certaines touches spécifiques à l'ordinateur que nous présentons brièvement :

ENTREE Conclut une ligne et provoque sa prise en compte.

Barre d'espacement. Déplace le curseur en laissant un espace (appelé le blanc).

← Déplace le curseur vers la gauche en effaçant un caractère.

CNT CONTROLE : utilisée simultanément avec une autre touche permet d'en modifier le sens.

CNT A pour [

CNT Z pour]

STOP Interrompt momentanément une exécution. Pour la reprendre, appuyer sur n'importe quelle touche du clavier.

Dans l'éditeur,

← déplace le curseur vers la gauche

→ déplace le curseur vers la droite

↑ déplace le curseur vers le haut

↓ déplace le curseur vers le bas.

Dans l'éditeur,

INS en combinaison avec une autre touche donne accès à différentes fonctions.

Dans l'éditeur,

EFF efface le caractère se trouvant à la place du curseur.

· Permet d'obtenir les caractères jaunes de certaines touches du clavier. Appuyer simultanément sur le point situé au milieu de cette touche et sur le caractère désiré. Attention ! Sur un MO5 cette touche est jaune, sans point dessus.

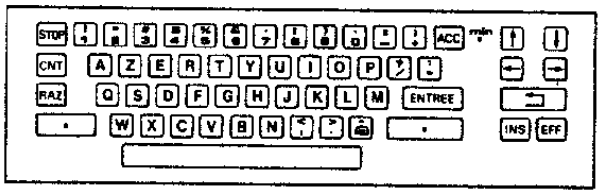
L'appui simultané de · et de ____ permet d'écrire en lettres minuscules. Pour remettre les majuscules, appuyez une deuxième fois sur ces deux mêmes touches.

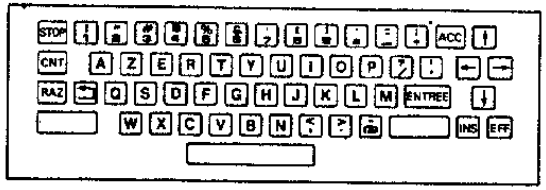
ACC Permet d'accentuer les lettres minuscules. Appuyer sur ACC , ensuite · et l'accent désiré, puis la lettre tapée se placera sous l'accent. Taper l'accent avant la lettre à accentuer. Pour le ç utiliser la touche ACC et ensuite deux fois sur la touche C . Sur le TO7-70 et le MO5 les minuscules accentuées les plus courantes s'obtiennent en apuyant sur :

ACC	puis	6	pour	é
ACC		7		è
ACC		8		ù
ACC		9		ç
ACC		Ø		à

Les instructions Logo doivent être en lettres majuscules. On peut utiliser les minuscules et les accents pour écrire des textes qui ne sont pas des instructions du langage.

Clavier TO7 et TO7-70





Clavier MO5

Rappel

les crochets [et]
sont obtenus par
CNT-A et CNT-Z

Première partie
La structure du langage Logo

1. Ce que comporte le langage Logo

Définitions

a) Les objets Logo

Tout comme une suite de caractères forme une unité appelée le mot, une suite de mots ou de listes forme une unité appelée la liste. Ces unités sont appelées : **objets Logo.** Un objet Logo est donc soit un mot, soit une liste.

Les mots

Un mot Logo est une suite de caractères (lettres, chiffres ou signes du clavier).

On peut donner un sens à un mot, c'est là votre initiative !

Les listes

Une liste est une unité composée d'une suite de mots ou d'autres listes.

b) Les noms et les choses

Un nom est un mot qui sert à désigner un objet Logo. L'objet Logo désigné par un nom est appelé sa "chose".

c) Les procédures

Une procédure est une action ou une série d'actions pouvant être associées à un mot qui devient alors le nom de la procédure. Par abus de langage, on appelle parfois ce nom procédure.

Exemples

— *Exemples de mots :*

```
?ECRIS "AZ23Ka
AZ23Ka
```

Le mot ECRIS est une procédure ; le mot AZ23Ka est sa donnée.

```
?ECRIS "BONJOUR
BONJOUR
```

Le mot ECRIS est une procédure ; le mot BONJOUR est sa donnée.

— Les nombres sont des mots particuliers.

```
?ECRIS 2578
2578
```

— *Exemples de listes :*

```
?ECRIS [ A B C ]
A B C
```

```
?ECRIS [BONJOUR [AVANCE 30] D2 R2]
BONJOUR [AVANCE 30] D2 R2
```

— Un **mot** devient un **nom** lorsqu'on lui affecte un objet Logo.

```
?DONNE "BONJOUR "SALUT
```

Le mot DONNE est une procédure ; les mots BONJOUR et SALUT sont ses données. La procédure DONNE permet de désigner le mot SALUT par le mot BONJOUR. (On DONNE le nom BONJOUR au mot SALUT.) BONJOUR sera non seulement la suite des lettres B,O,N,J,O,U,R mais aussi un nom désignant l'objet Logo SALUT, et le mot SALUT sera la chose du nom BONJOUR.

— *Exemple de procédure :*

```
?POUR BONJOUR
>ECRIS "SALUT
>FIN
VOUS VENEZ DE DEFINIR BONJOUR
```

La procédure POUR va associer au mot BONJOUR l'action ECRIS "SALUT. Le mot BONJOUR est maintenant devenu une procédure.

```
?BONJOUR
SALUT
```

Le langage Logo vous permet de construire vos programmes. Un programme écrit en Logo est constitué de "briques" pouvant être assemblées de différentes manières, mais toujours en obéissant à certaines règles. Ces règles simples et peu nombreuses constituent la "grammaire du langage".

2. Procédures primitives et procédures utilisateur

Certaines procédures sont toujours comprises par Logo car elles sont connues du système au départ. Ce sont des "procédures primitives", que l'on appelle plus simplement des **primitives,** et qui constituent le vocabulaire initial de Logo.

Si vous tapez par exemple :

```
?NETTOIE
```

Les traces de la tortue disparaîtront de l'écran sans qu'elle bouge. Vous n'avez pas défini NETTOIE mais Logo sait déjà ce que cela veut dire.

Il y a aussi les procédures utilisateur, celles que vous définissez vous-mêmes en utilisant la primitive POUR. Nous les appelons plus simplement des **procédures.** Vous en trouverez beaucoup d'exemples dans les pages qui vont suivre.

2.1 Définir et faire exécuter une procédure

Exemple de définition de procédure :

```
POUR BONJOUR
ECRIS "SALUT
FIN
```

La première et la dernière ligne ont leurs propres règles. La première est appelée la "ligne de titre de la procédure". Elle doit toujours commencer par POUR suivi du nom de la procédure. Notez que le nom de la procédure ne peut être précédé ni de guillemet ("), ni de deux-points (:).

La dernière ligne doit toujours contenir uniquement FIN.

Quand on utilise une procédure, on dit qu'on **"appelle"** cette procédure.

BONJOUR exprime la demande à Logo d'exécuter ECRIS "SALUT (ECRIS est une procédure primitive que la procédure BONJOUR appelle).

Il y a une autre façon d'appeler une procédure : en tapant directement le nom de la procédure quand Logo est au niveau supérieur (ce qu'on appelle aussi le mode direct, c'est-à-dire lorsque le symbole d'invite

(?) apparaît en début de ligne). Nous avons déjà vu un exemple avec NETTOIE.

Voici un autre exemple :

```
?ECRIS "SALUT
SALUT

?BONJOUR
SALUT
```

Si Logo cherche et ne trouve pas la procédure associée à un mot il affiche un message. Supposez, par exemple, que vous n'avez pas défini de procédure CARRE (ou que vous ne l'avez pas ramenée dans la mémoire de votre ordinateur) et que vous tapiez :

```
?CARRE
COMMENT FAIRE CARRE
```

Dans la définition d'une procédure, vous pouvez bien entendu faire appel à une autre procédure que vous avez précédemment définie.

```
POUR BONJOURTOUS
BONJOUR
ECRIS [ET BONJOUR A TOUS]
FIN

?BONJOURTOUS
SALUT
ET BONJOUR A TOUS
```

2.2 Procédures avec données

Une donnée est une information dont doit disposer une procédure pour réaliser son action. Cette information devra être un objet Logo (mot ou liste).

Certaines procédures ont besoin de données.

Par exemple :

```
?ECRIS "SALUT
SALUT
```

Voilà ce qui arriverait si Logo ne trouvait pas la donnée nécessaire :

```
?ECRIS
PAS ASSEZ DE DONNEES POUR ECRIS
```

Le nombre de données nécessaires et leur nature dépend de la procédure.

```
?AVANCE 45
```


AVANCE est une primitive qui attend une seule donnée : un nombre.

```
?REPETE 4 [AVANCE 50 TD 90]
```

REPETE attend deux données : un nombre et une liste.

Les procédures que vous définissez peuvent aussi avoir des données. Dans ce cas, lors de la définition de la procédure, les noms des données (chacun précédé de deux-points), doivent figurer dans le titre, après le nom de la procédure. N'importe quel mot peut servir de nom de donnée.

Vous préciserez la ou les donnée(s) au moment de l'appel de la procédure.

Par exemple :

```
POUR SUPERBIENVENUE :PRENOM
ECRIS "SALUT
ECRIS :PRENOM
ECRIS [BONNE JOURNEE]
FIN
```

La ligne de titre indique à Logo que la procédure SUPERBIENVENUE a une seule donnée dont le nom est PRENOM. Le corps de la procédure contient trois appels à la procédure ECRIS. Le second de ces appels utilise le nom de donnée PRENOM. Voici un exemple de demande d'exécution de la procédure SUPERBIENVENUE au niveau supérieur.

```
?SUPERBIENVENUE "JEAN
SALUT
JEAN
BONNE JOURNEE
```

Ici la donnée de SUPERBIENVENUE est JEAN; Logo la prend comme la chose de PRENOM quand il exécute la procédure. Donc ECRIS :PRENOM est exécuté comme ECRIS "JEAN.

3. Commandes et opérations

Certaines procédures rendent un objet Logo quand elles sont exécutées. Celui-ci doit toujours être la donnée d'une autre procédure, sinon un message Logo apparaît.

Par exemple :

```
?SOMME 31 28
QUE FAIRE DE 59

?ECRIS SOMME 31 28
59
```

Il y a donc deux sortes de procédures en Logo. Celles qui rendent toujours un objet Logo après leur exécution sont appelées **opérations ;** celles qui ne rendent jamais d'objet Logo après leur exécution sont appelées **commandes.** Cette distinction est si importante que dans la suite de ce manuel il vous sera indiqué à chaque fois si une primitive est une commande ou une opération.

L'objet Logo rendu par une opération est appelé **résultat** de l'opération.

Par exemple, ECRIS et DONNE sont des commandes, HASARD et SOMME sont des opérations.

Nous avons déjà vu l'exemple ECRIS SOMME 31 28, en voici un autre.

```
?ECRIS HASARD 2
```

Logo va afficher un nombre entier positif inférieur à 2 : soit 1 soit 0.

HASARD 2 apparaît comme la donnée de ECRIS. Lorsque HASARD 2 est exécutée, son résultat est communiqué à ECRIS.

Si vous essayez d'utiliser une commande comme donnée d'une procédure vous obtiendrez ceci :

```
?ECRIS DONNE "X 25
PAS ASSEZ DE DONNEES POUR ECRIS
```

On obtient ce message parce que DONNE est une commande.

DONNE exécute son action, mais ne rend rien à ECRIS qui attend une donnée.

Jusqu'à maintenant, nous n'avons considéré que les procédures primitives de Logo, mais toutes les procédures que vous définirez vous-même se comporteront soit comme des commandes, soit comme des opérations lors de l'exécution. Par exemple, la procédure SUPERBIENVENUE est une commande. (Elle ne rend jamais de résultat.)

Exemple d'une opération :

```
POUR PILE-OU-FACE
SI EGAL? HASARD 2 0 [RENDS "PILE] [RENDS "FACE]
FIN
```

Cette procédure rend le mot PILE si HASARD 2 rend 0, et le mot FACE si HASARD 2 rend 1. C'est donc une opération définie par l'utilisateur.

```
?PILE-OU-FACE
QUE FAIRE DE PILE
ou
QUE FAIRE DE FACE
```

D'un autre côté, nous avons :

```
?ECRIS PILE-OU-FACE
PILE
ou
FACE
```

Pour construire vos propres opérations, vous devrez toujours utiliser la commande RENDS. Pour plus de détails voir la deuxième partie de ce manuel.

4. Les noms dans les procédures

Reprenons la procédure SUPERBIENVENUE :PRENOM. Lors de l'appel de cette procédure, un objet Logo va être donné au mot PRENOM pendant toute la durée de l'exécution de SUPERBIEN-VENUE "JEAN, c'est-à-dire jusqu'à réapparition du symbole d'invite. PRENOM est le nom donné à la chose JEAN.

Après exécution de la procédure, le mot PRENOM se retrouve dans l'état où il était avant l'appel de la procédure SUPERBIENVENUE.

Exemples :

```
POUR BONJOUR :PRENOM
ECRIS :PRENOM
FIN
```

Pendant l'exécution de BONJOUR "JEAN, PRENOM est le nom de l'objet Logo JEAN.
Après l'exécution, PRENOM n'a plus de chose.

Si auparavant, on a fait DONNE "PRENOM "FRANÇOISE ceci ne change rien à l'exécution de BONJOUR "JEAN mais après l'exécution PRENOM redevient le nom désignant FRANÇOISE.

```
POUR COUCOU :PRENOM
SALUT :PRENOM
ECRIS [AU REVOIR] ECRIS :PRENOM
FIN
POUR SALUT :X
ECRIS "BONJOUR ECRIS :X
FIN
```

COUCOU appelle SALUT. Si on demande l'exécution de COUCOU "JEAN, au moment de l'appel de SALUT, X devient le nom de l'objet Logo JEAN. Dès la fin de l'exécution de SALUT, c'est-à-dire après l'affichage de BONJOUR JEAN, X n'a plus pour "chose" JEAN, par contre PRENOM a toujours JEAN pour chose.

Un exemple un peu plus complexe :

```
POUR ECRIRE-CHOSE :X
ECRIS CHOSE :X
FIN
?DONNE "Y 6
```

Nous donnons au mot Y le mot 6

```
?ECRIRE-CHOSE "Y
6
```

Pendant l'exécution de la procédure, X devient le nom désignant le mot Y.
:X est donc Y (la chose de X) et CHOSE :X est donc 6 (la chose de Y).

```
?DONNE "X 4
```

Nous donnons au mot X le mot 4.

```
?ECRIRE-CHOSE "X
X
```

Avant l'exécution de la procédure, le mot X est le nom désignant le mot 4.
Pendant l'exécution, X devient le nom désignant le mot X donc de lui-même.
:X est donc X et la chose de X est X.
Après l'exécution, X retrouve l'état où il était avant cette exécution.

```
?ECRIS CHOSE "X
4
```

5. La ligne Logo

Une ligne Logo est une suite d'instructions (une ou plusieurs) conclue par l'appui de la touche [ENTREE]. En mode direct une ligne Logo peut contenir jusqu'à 248 caractères. Par contre, dans l'éditeur elle est illimitée (voir Chapitre 11). Une ligne Logo peut s'étendre sur plusieurs lignes d'écran.

Par exemple :

```
?DONNE "LISTENOMS [ANNE BERNARD CHARLES!
DENISE ERIC FRANCOISE GILLES HELENE]
```

Les lignes Logo peuvent être complexes et nous allons montrer comment les analyser en utilisant nos connaissances sur la différence entre commandes et opérations pour les découper en groupes significatifs.

Voici quelques indications pour vous guider :

1. Dès que vous voyez un nom de procédure, assurez-vous que vous savez :
a) combien de données elle attend
b) s'il s'agit d'une commande ou d'une opération.

2. Le premier mot d'une ligne Logo est toujours une commande.

3. Une opération est toujours placée comme donnée d'une autre procédure.

4. Une donnée peut être une valeur explicite (mot précédé de guillemets ou liste entre crochets), le résultat d'une opération, ou la chose d'un nom (mot précédé de deux points).

5. Lorsque toutes les données nécessaires à une commande ont été répertoriées la procédure suivante doit être une commande.

Notez bien que toutes les données d'une procédure doivent apparaître sur la même ligne Logo.

Voici un exemple de ligne Logo complexe :

```
REPETE :N [DONNE "I SOMME :I 1 ECRIS CHOSE MOT :TOTO :I] TAPE
"BONJOUR
```


Analysons-la :

REPETE est une commande à deux données : un nombre et une liste d'instructions. TAPE est une commande à une donnée : "BONJOUR (le mot BONJOUR).

Voici le premier découpage.

Groupe 1 : REPETE et ses données

Groupe 2 : TAPE et sa donnée

Considérons les deux données de REPETE, nous voyons d'abord :N (la chose de N) qui est la première donnée, puis une liste d'instructions donc, la seconde donnée.

Groupe 1.a : :N
Groupe 1.b : la liste d'instructions

Pour le prochain découpage, nous considérerons la liste d'instructions, celle-ci sera analysée comme une ligne Logo. DONNE est une commande à deux données. ECRIS est une commande à une donnée. Ce sont les seules commandes de la liste, nous avons donc :

Groupe 1.b.1 : DONNE et ses données
Groupe 1.b.2 : ECRIS et ses données

La première donnée de DONNE est "I (c'est-à-dire le mot I). La seconde est SOMME :I 1, c'est-à-dire le résultat de l'opération SOMME qui elle-même a deux données :I (la chose de I), et 1. Nous avons donc :

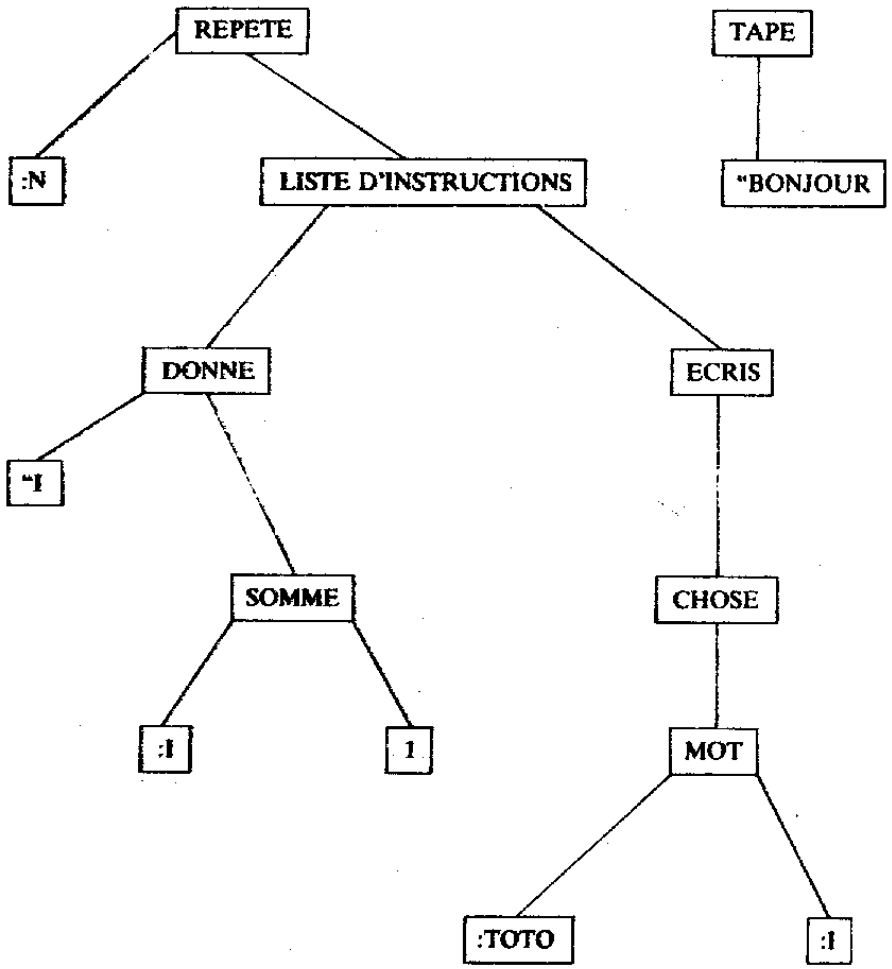
Groupe 1.b.1.a. : "I
Groupe 1.b.1.b. : SOMME et ses données

Nous allons maintenant considérer le groupe ECRIS. La donnée de ECRIS est CHOSE MOT :TOTO :I, c'est-à-dire le résultat de l'opération CHOSE.

CHOSE est une primitive ; il s'agit d'une opération à une seule donnée, MOT :TOTO :I, qui sera le résultat de l'opération MOT.

MOT est une opération à deux données qui sont :TOTO et :I, c'est-à-dire les choses de TOTO et de I. Nous avons donc terminé le découpage de la ligne Logo.

Le diagramme suivant résume ce que nous avons fait :



6. Symboles spéciaux

Dans une ligne Logo certains caractères ont une fonction particulière :

Les séparateurs

— Espace — permet de séparer des mots.

— Crochets [] — indiquent le début (crochet ouvrant) et la fin d'une liste (crochet fermant).

— Parenthèses () — indiquent le début et la fin d'un groupement dont on demande à Logo l'évaluation autonome prioritaire (voir Chapitre 5).

— Le dollar $ — permet de ne plus considérer comme tels les caractères séparateurs.
Doit précéder le caractère.
ex. ?ECRIS "SAINT$ MICHEL
SAINT MICHEL

Les symboles d'opération

Ces signes ne sont pas des séparateurs ; pour leur écriture, ils doivent donc être précédés et suivis par des séparateurs ; ils sont infixés, c'est-à-dire qu'on les écrit entre les deux nombres sur lesquels porte l'opération (voir Chapitre 7).

/ division

* multiplication

− symbole de la soustraction, du nombre négatif ou le moins unaire. ex. : si :X est 2, − :X rend −2.

+ addition

> supérieur

< inférieur

= égal

Autres symboles

Guillemet (") Utilisé immédiatement devant un mot, le guillemet indique que ce mot est explicitement la donnée d'une procédure. Les nombres n'ont pas besoin d'être précédés de guillemets.

Deux points (:) Utilisés immédiatement devant un mot, les deux points indiquent que ce mot devra être un nom et rendent la chose de ce nom (l'objet Logo désigné par ce nom).

ECRIS :X est équivalent à ECRIS CHOSE "X

Deuxième partie

Les primitives du langage Logo

Toutes les primitives du langage Logo sont présentées et décrites exhaustivement dans les différents chapitres qui suivent. Elles sont regroupées par domaines d'application.

Chaque primitive est traitée de la façon suivante :

nom de la primitive
et type de la donnée réclamée **statut**

ECRIS ***obj*** **(commande)**
EC ***obj***

description de l'action

exemple

Pour les primitives qui sont des opérations, nous avons indiqué le type de résultat rendu sous le statut de la primitive.

Pour les données indiquées (ou les résultats rendus) nous avons utilisé les abréviations suivantes :

a	pour	adresse mémoire
car	pour	caractère
n	pour	nombre
obj	pour	objet Logo
pred	pour	le résultat d'un prédicat.

Certaines primitives ont besoin d'un entier comme donnée. Si la donnée fournie n'est pas entière, sa partie entière est prise en compte sauf indication du contraire.

Les restrictions particulières sur les données sont indiquées dans le cas de certaines primitives.

Seules les primitives arithmétiques peuvent accepter des nombres supérieurs à 65535 et inférieurs à −65535.

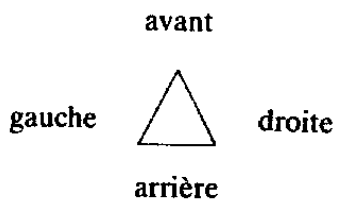
Les primitives commençant par un point doivent être utilisées avec précaution.

Les procédures définies au cours de ce manuel, sont non seulement des exemples d'utilisation des primitives, mais aussi des utilitaires pouvant vous aider dans la programmation. Un point d'exclamation (!) à la fin d'une ligne indique que la ligne Logo continue sur la ligne suivante.

Chapitre 1
Le monde de la tortue

Nous aborderons, dans ce chapitre, les primitives relatives aux déplacements de la tortue, à son crayon et au champ dans lequel elle évolue.

La tortue est un triangle qui symbolise un robot orienté :



Elle est munie d'un crayon avec lequel elle peut laisser des traces quand elle se déplace sur l'écran.

L'**état** de la tortue est donné par sa **position** et son **orientation** dans le champ à un moment donné.

Dès qu'on exécute une des primitives suivantes : AVANCE, (AV), RECULE, (RE), TD, TG, FCAP, FPOS, CT, MT, ORIGINE, ME, FCFG, FEN, ENR, CLOS, POINT, l'écran graphique apparaît. Quatre lignes restent disponibles pour le texte dans la partie inférieure de l'écran.

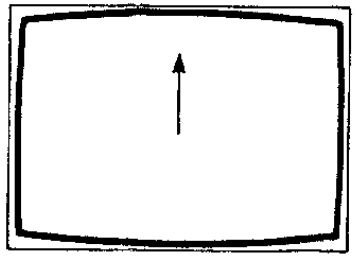
L'écran graphique est effacé quand on utilise l'éditeur Logo.

1.1 Les déplacements

AVANCE *n*
AV *n*

(commande)

fait avancer la tortue dans la direction en cours de ***n*** pas (32767 au maximum). La première décimale de ***n*** est prise en compte. Les valeurs possibles pour ***n*** dépendent de l'état de la tortue et de celui du champ (voir champ p. 34).

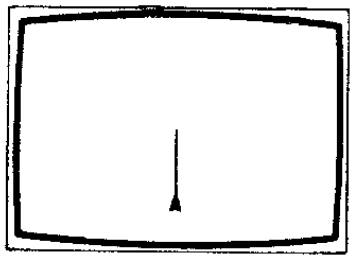


AVANCE 50

RECULE *n*
RE *n*

(commande)

fait reculer la tortue dans la direction en cours de ***n*** pas (32767 au maximum). La première décimale de ***n*** est prise en compte. Les valeurs possibles pour ***n*** dépendent de l'état de la tortue et de celui du champ (voir champ p. 34).

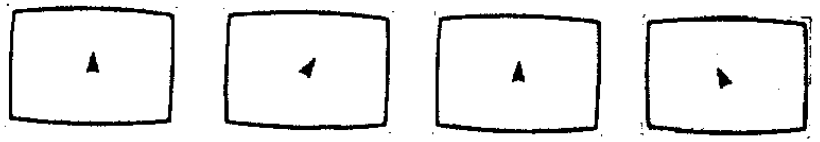


RECULE 50

TD *n*
(Tourne à Droite)

(commande)

fait pivoter la tortue sur sa droite de ***n*** degrés (dans le sens des aiguilles d'une montre). ***n*** est un nombre compris entre −360 et 360. La première décimale de ***n*** est prise en compte.

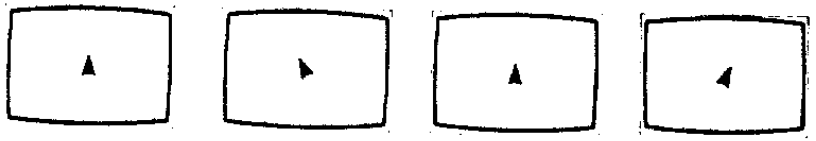


TD 45 TD −45

TG *n*
(Tourne à Gauche)

(commande)

fait pivoter la tortue sur sa gauche de ***n*** degrés (dans le sens contraire des aiguilles d'une montre). ***n*** est un nombre compris entre −360 et 360. La première décimale de ***n*** est prise en compte.



TG 45 TG −45

Pour dessiner une portion de cercle droit ou gauche :

```
POUR ARCD :N
REPETE :N [AVANCE 1 TD 1]
FIN

POUR ARCG :N
REPETE :N [AVANCE 1 TG 1]
FIN
```

Et un cercle de circonférence variable

```
POUR CERCLE :PAS
REPETE 360 [AVANCE :PAS TD 1]
FIN
```

POS
(POSition) (opération)
(liste)

rend une liste formée de 2 nombres. Le premier est la coordonnée horizontale et le deuxième la coordonnée verticale de la tortue dans le champ (voir champ p. 34).

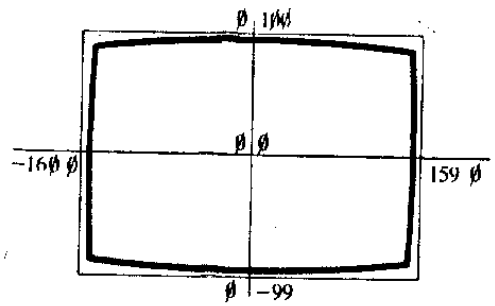
Pour connaître seulement la coordonnée horizontale de la tortue :

```
POUR XCOOR
RENDS PREM POS
FIN
```

...et la coordonnée verticale

```
POUR YCOOR
RENDS DER POS
FIN
```

Système de coordonnées de l'écran graphique



FPOS *liste*
(Fixe POSition) (commande)

fixe la position de la tortue à liste. ***liste*** comporte un nombre pair de nombres représentant les différentes coordonnées horizontale et verticale d'une suite de points qui seront successivement parcourus par la tortue. FPOS ne modifie pas l'orientation de la tortue. Les coordonnées horizontales et les coordonnées verticales dépendent du champ (voir champ p. 34).

La première décimale de chaque coordonnée est prise en compte.

Pour dessiner un cadre

```
POUR CADRE
LC
FPOS [−60 −60]
BC
FPOS [−60 60  60 60  60 −60  −60 −60]
FIN
```

Pour fixer seulement la coordonnée horizontale de la tortue :

```
POUR FX :X
FPOS LISTE :X DER POS
FIN
```

et sa coordonnée verticale :

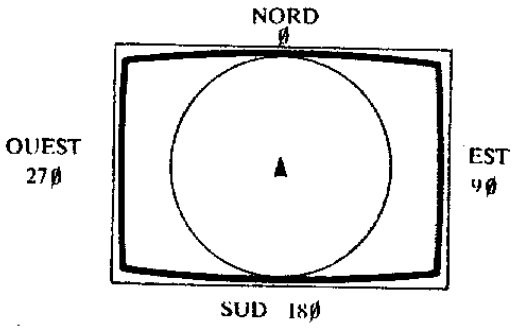
```
POUR FY :Y
FPOS LISTE PREM POS :Y
FIN
```

CAP (opération)
(n)

rend un nombre compris entre 0 et 359,9 qui est l'orientation en cours de la tortue (en degrés).

FCAP *n*
(Fixe CAP) (commande)

fixe le cap de la tortue dans la direction correspondant à ***n*** degrés (comme sur une boussole, le nord étant le haut de l'écran). Notez que TD et TG définissent des mouvements relatifs à l'état de la tortue tandis que FCAP est indépendant de l'orientation en cours. ***n*** doit être compris entre −360 et 360. La première décimale de ***n*** est prise en compte.



1.2 Le champ

Le champ de la tortue est repéré par un système de coordonnées cartésiennes, dont l'origine [0 0] se situe au centre de l'écran.

Initialement, la tortue peut se déplacer dans toute la zone limitée par le bord de l'écran (y compris sur les lignes d'écriture). Par défaut, lorsqu'on pilote la tortue, 4 lignes au bas de l'écran sont utilisées pour l'écriture.

Le champ visible de la tortue est encadré par un bord dont la couleur est indépendante.

Note un peu technique :

L'écran se compose de 200 lignes graphiques ; chaque ligne est formée de 40 segments de 8 points (octets). A l'intérieur de chaque segment les points peuvent prendre la couleur du crayon ou la couleur du fond.

L'écran de votre ordinateur affiche le contenu d'un ensemble de mémoires organisées, en octets.

Ainsi, les 8 premiers points en haut à gauche de l'écran seront solidaires et représentent les 8 bits (0 ou 1) de l'octet d'adresse 16384 pour les TO7 et TO7-70 ou l'octet d'adresse 0 pour le MO5. Les points suivants sont contenus dans les octets des mémoires suivantes, jusqu'à 24383 sur TO7 et TO7-70 et 7999 sur MO5 pour les 8 derniers points en bas à droite de l'écran.

Pour un octet donné, les bits à 0 apparaîtront dans la couleur du fond et ceux à 1 dans la couleur du crayon. Un octet ne peut comporter que deux couleurs, car l'information sur la colorimétrie porte sur chaque octet. Cette information est d'ailleurs contenue dans une mémoire dont l'adresse est la même que celle de l'octet qui définit les points allumés. Pour déclarer au système qu'on accède à la mémoire couleur ou la mémoire forme, le bit de poids faible de l'octet 59331 sur TO7 ou TO7-70 ou 42944 sur MO5 sera mis respectivement à 0 ou à 1. Attention, les autres bits de cet octet peuvent perturber la couleur du bord, le clavier ou même la page graphique.

CF (opération)
(Couleur Fond) **(n)**

rend un nombre entier qui est le numéro de code de la couleur du champ.

```
?ECRIS CF
2
```

FCFG *n* (commande)
(Fixe Couleur Fond Graphique)

fixe la couleur du champ à celle correspondant au numéro ***n*** du code couleur.

0	NOIR
1	ROUGE
2	VERT
3	JAUNE
4	BLEU
5	MAGENTA (VIOLET)
6	CYAN (BLEU CLAIR)
7	BLANC

8	GRIS
9	ROUGE PALE
10	VERT PALE
11	JAUNE PALE
12	BLEU PALE
13	MAGENTA PALE
14	CYAN PALE
15	ORANGE

sur MO5 et TO7-70 seulement

Les codes supérieurs à 7 sur TO7 (ou 15 sur MO5 ou TO7-70) sont interprétés modulo 8 sur TO7 (ou 16 sur MO5 ou TO7-70).

```
?FCFG 2
```
le champ devient vert.

FCB *n* (commande)
(Fixe Couleur Bord)

fixe la couleur du bord de l'écran à celle correspondant au numéro ***n*** du code couleur ci-dessus.

La procédure suivante vous permettra de choisir la couleur du bord de l'écran en appuyant sur une touche, la barre d'espacement par exemple. La touche S arrête la procédure.

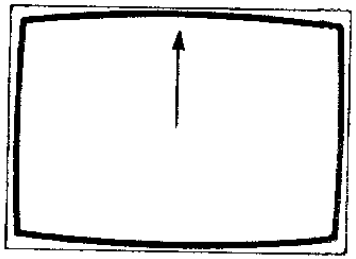
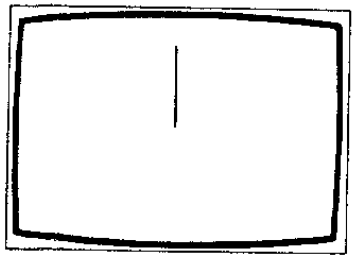
```
POUR CHOIXBORD :X
SI EGAL ? LISCAR "S[STOP]
FCB :X
CHOIXBORD :X + 1
FIN
```

Cette procédure vous permettra de changer la couleur de l'écran à l'aide de la touche S.

```
POUR CHOIX :N
FCFG :N
CHOIXBORD 0
CHOIX :N + 1
FIN
```

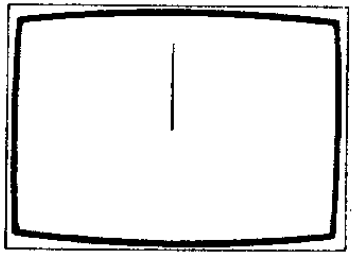
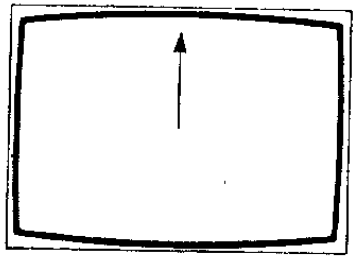
CT (commande)
(Cache Tortue)

cache la tortue, c'est-à-dire la rend invisible, sans changer son état. L'exécution des procédures graphiques est alors plus rapide.

MT (commande)
(Montre Tortue)

montre la tortue, c'est-à-dire la rend visible, sans changer son état (mais on ne peut la voir que si elle est sur l'écran !). Au démarrage ou après VE, la tortue est visible.

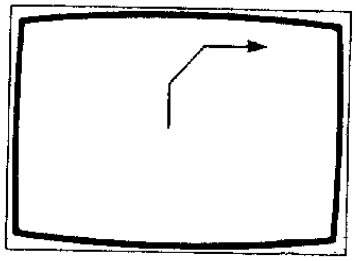
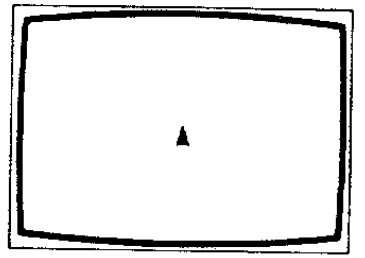



VISIBLE? (opération)
(pred)

rend VRAI si la tortue est visible, FAUX sinon.

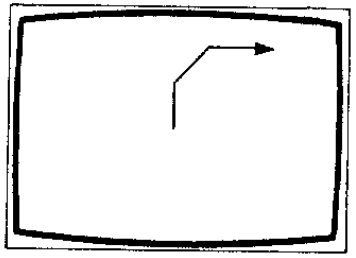
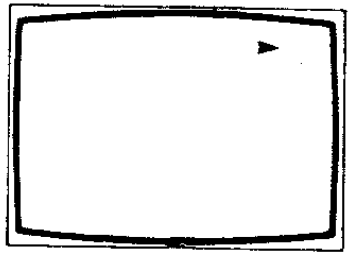
VE (commande)
(Vide Ecran)

Permet d'obtenir un écran graphique réinitialisé : 4 lignes de texte, couleurs cyan et bleu, tortue à l'origine ([0 0]) cap au Nord, crayon baissé.

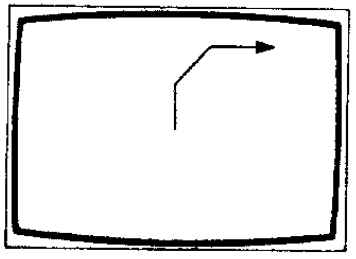
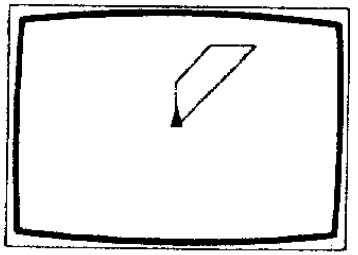
NETTOIE (commande)

efface le champ sans modifier ni l'état de la tortue, ni celui du crayon (voir le crayon p. 40).

ORIGINE (commande)

ramène la tortue au centre de l'écran cap au Nord sans modifier l'état de son crayon (voir p. 40). ORIGINE est équivalent à FPOS [0 0] et FCAP 0.
Notez que suivant l'état de son crayon, la tortue laissera ou non la trace de son déplacement.

ECH (opération)
(ECHelles) (liste)

rend une liste de deux entiers. Le premier nombre (échelle horizontale) représente 100 fois la taille de la projection sur l'axe horizontal d'un pas de tortue, par rapport à la taille de la projection du pas initial. Le second (échelle verticale) représente le même rapport sur l'axe vertical. Au démarrage, l'échelle est fixée à [100 100].

FECH ***liste*** (commande)
(Fixe ECHelles)

fixe les échelles horizontale et verticale du pas de la tortue à ***liste***. ***liste*** comporte 2 nombres compris entre 0 et 200. Cette commande permet de modifier la taille des pas effectués par la tortue au moyen de AV et RE (voir ECH).
Les échelles ne sont pas affectées par les réinitialisations de l'écran graphique (VE, FEN, etc.).

```
?FECH [150 180]
?ECRIS ECH
150 180
```

Ces possibilités de changement d'échelle permettent soit de modifier le rapport entre les unités horizontales et verticales (déformation des courbes : cercle → ellipse, carré → rectangle...) soit d'effectuer l'agrandissement ou la miniaturisation d'un dessin.

Reprenons le cercle précédemment défini (p. 31) et utilisons les possibilités de variations du pas de la tortue pour construire un faisceau de cercles.

```
POUR FAISCEAU
FECH [50 50] CERCLE .5
FECH [100 100] CERCLE .5
FECH [150 150] CERCLE .5
FECH [200 200] CERCLE .5
FIN
```

ou transformer un cercle en ellipses :

```
POUR ELLIPSES
FECH [20 100] CERCLE 1
FECH [100 20] CERCLE 1
FIN
```


CLOS (commande)

limite le champ de la tortue au bord de l'écran. Les coordonnées devront alors rester entre −99 et 100 verticalement, et entre − 160 et 159 horizontalement. Cette commande réinitialise l'écran graphique. Par défaut, le champ de la tortue est CLOS.

ENR (commande)
(ENRoule)

enroule le champ de la tortue sur lui-même : le bord supérieur est rattaché au bord inférieur, le bord droit au bord gauche. Quels que soient ses déplacements, la tortue évolue sur l'écran : quand elle atteint un bord de l'écran, elle réapparaît sur le bord opposé. Ses coordonnées restent toujours entre −99 et 100 verticalement et entre −160 et 159 horizontalement. Cette commande réinitialise l'écran graphique.

FEN (commande)
(FENêtre)

étend le champ de la tortue. Les coordonnées horizontales et verticales pourront alors s'étendre de −32768 à 32767,9. L'écran est une fenêtre ouverte au centre de ce champ. La tortue peut alors se déplacer en dehors des limites de l'écran.
Le champ, très grand, est aussi enroulé sur lui-même. Cette commande réinitialise l'écran graphique.

Les procédures suivantes permettent à la tortue de rebondir dès qu'elle atteint un bord de l'écran :

```
POUR REBOND :N
SI EGAL? :N 0 [STOP]
AVANCE 1
SI HORS POS [FCAP 360 − CAP]
REBOND :N − 1
FIN

POUR HORS :L
SI PLP? 159 PREM :L [RENDS VRAI]
SI PLG? −160 PREM :L [RENDS VRAI]
SI PLG? −99 DER :L [TD 180 RENDS VRAI]
SI PLP? 100 DER :L [TD 180 RENDS VRAI]
RENDS FAUX
FIN
```

POINT *liste* (commande)

liste comporte un nombre pair de nombres, représentant les coordonnées horizontale et verticale d'une suite de points qui apparaîtront successivement sur l'écran, dans la couleur courante du crayon.
L'état de la tortue n'est pas modifié. Les coordonnées doivent être comprises entre −160 et 159 horizontalement, et entre −99 et 100 verticalement. La première décimale de chaque coordonnée est prise en compte.

```
POUR NEBULEUSE :X
SI :X > 80 [LC ORIGINE STOP]
LC AV :X POINT POS TD 70
NEBULEUSE :X + 2
FIN
```

1.3 Le crayon

L'**état** du crayon de la tortue est donné par sa **situation** (levé ou baissé) et sa **couleur.**

BC (commande)
(Baisse Crayon)

baisse le crayon de la tortue pour dessiner sur l'écran. Celle-ci laissera la trace de ses déplacements dans la couleur courante. Au démarrage ou après une réinitialisation de l'écran graphique, le crayon est baissé et de couleur cyan.

LC (commande)
(Lève Crayon)

lève le crayon de la tortue : elle ne laissera pas de trace lors de ses déplacements. La couleur du crayon n'est pas modifiée.

BC? (opération)
(Baisse Crayon?) (pred)

rend VRAI si le crayon de la tortue est baissé, FAUX sinon.

CC (opération)
(Couleur Crayon) (n)

rend le nombre qui correspond au code de la couleur courante du crayon.

```
?FCC 6
?ECRIS CC
6
```

FCC *n* (commande)
(Fixe Couleur Crayon)

fixe la couleur du crayon à celle correspondant au numéro ***n*** du code couleur si ***n*** est un nombre positif ou nul.
Pour ***n*** entier compris entre 0 et 7 pour le TO7 ou 15 pour MO5 et TO7-70, la table des codes de couleur est la suivante :

couleur crayon	code	couleur crayon	code	
NOIR	0	GRIS	8	Sur MO5 ou TO7-70 seulement
ROUGE	1	ROUGE PALE	9	
VERT	2	VERT PALE	10	
JAUNE	3	JAUNE PALE	11	
BLEU	4	BLEU PALE	12	
MAGENTA	5	MAGENTA PALE	13	
CYAN	6	CYAN PALE	14	
BLANC	7	ORANGE	15	

Si ***n*** est négatif, FCC fixe la couleur qui sera prise par le fond des octets sur lesquels passera la tortue.

Pour ***n*** entier négatif compris entre −1 et −8 pour le TO7 ou −16 pour MO5 et TO7-70 la table de code est :

couleur crayon	code	couleur crayon	code	
NOIR	—1	GRIS	— 9	Sur MO5 ou TO7-70 seulement
ROUGE	—2	ROUGE PALE	—10	
VERT	—3	VERT PALE	—11	
JAUNE	—4	JAUNE PALE	—12	
BLEU	—5	BLEU PALE	—13	
MAGENTA	—6	MAGENTA PALE	—14	
CYAN	—7	CYAN PALE	—15	
BLANC	—8	ORANGE	—16	

Si vous avez avancé de 50 pas et que vous voulez effacer ce dernier trait, il suffit de le retracer de la même couleur que le fond.

C'est ce que réalisent les procédures suivantes :

```
POUR GOMME
DONNE "COULEURCOURANTE CC
FCC — (CF + 1)
FIN

POUR ARRETEGOMME
FCC :COULEURCOURANTE
FIN

?AVANCE 50
?GOMME
?RECULE 50
?ARRETEGOMME
?RECULE 30
```

La procédure suivante vous donnera un échantillon de toutes les couleurs de crayon possibles, à vous de choisir :

```
POUR ECHANTILLON :X
FCC :X
REPETE 10 [AVANCE 50 TD 90 AVANCE 0.5 TG 90
RECULE 50 TD 90 AVANCE 0.5 TG 90]
ECHANTILLON :X + 1
FIN
```


Chapitre 2

Le monde des mots et des listes

Il existe deux types d'objets Logo : les **mots** et les **listes.** Les primitives que nous présentons dans ce chapitre servent à les examiner, à les démonter ou à les construire.

Les mots

Un mot est composé de caractères. Lorsqu'ils apparaissent dans la ligne Logo, les mots sont délimités par des caractères séparateurs (généralement l'espace). Les deux-points (:) ou le guillemet (") qui précèdent éventuellement un mot doivent être accolés au début du mot.
Par abus de langage, un mot comprenant un seul caractère est appelé caractère.
Le caractère $ permet d'écrire des mots comprenant un caractère séparateur (voir p. 25).
Il existe un mot ne comprenant aucun caractère : on l'appelle le **mot vide.** Dans une ligne Logo, un guillemet suivi d'un séparateur représente le mot vide (deux-points suivis d'un séparateur représentent sa chose).

Les listes

Une liste est une suite d'un nombre quelconque d'objet Logo (mots ou autres listes), qui en sont ses membres.

Dans une ligne Logo, une liste se représente entre crochets. Il existe une liste ne comprenant aucun membre : on l'appelle la **liste vide** et on l'écrit [ ].

2.1. Examiner

EGAL? ***obj1 obj2***
obj1 **=** ***obj2***

(opération)
(pred)

rend VRAI si ***obj1*** et ***obj2*** sont des mots ou des listes identiques, FAUX sinon.
Dans la forme infixée (***obj1*** **=** ***obj2***) les données doivent être de chaque côté du symbole égal (=).
Pour la notation infixée voir Chapitre 7.

```
?ECRIS EGAL? "BONJOUR "BONJOUR
VRAI

?ECRIS EGAL? [BONJOUR] [[BON] JOUR]
FAUX
```

Attention : Deux listes sont égales si tous leurs membres sont égaux un à un.
Deux mots sont égaux s'ils ont les mêmes caractères (sauf dans le cas des nombres).
Deux nombres sont égaux s'ils représentent la même valeur.

```
?ECRIS MOT 1 "E2
1E2

?ECRIS 100
100

?ECRIS EGAL? MOT 1 "E2 100
VRAI
```

VIDE? ***obj***

(opération)
(pred)

rend VRAI si ***obj*** est la liste vide ou le mot vide, FAUX sinon.

```
?ECRIS VIDE? [ ]
VRAI
```

LISTE? ***obj***

(opération)
(pred)

rend VRAI si ***obj*** est une liste, FAUX sinon.

```
?ECRIS LISTE? "BONJOUR
FAUX

?ECRIS LISTE? [1 2 3 4 5]
VRAI
```


MOT? ***obj***

(opération)
(pred)

rend VRAI si ***obj*** est un mot, FAUX sinon.
MOT? ***obj*** est équivalent à NON LISTE? ***obj***.

```
?ECRIS MOT? "OUI
VRAI
```

MEMBRE? ***obj liste***

(opération)
(pred)

rend VRAI si ***obj*** est un membre de ***liste***, FAUX sinon.

```
?ECRIS MEMBRE? "BEAU [IL FAIT BEAU]
VRAI
```

La procédure suivante indique si un caractère fait partie d'un mot :

```
POUR MEMBREMOT? :LETTRE :MOT
SI VIDE? :MOT [RENDS "FAUX]
SI EGAL? :LETTRE PREM :MOT [RENDS "VRAI]
RENDS MEMBREMOT? :LETTRE SP :MOT
FIN

?ECRIS MEMBREMOT? "Z "REGARDEZ
VRAI
```

Pour connaître les mots faisant partie d'une liste mais pas d'une autre :

```
POUR COMPLEMENT :LISTE1 :LISTE2 :MOT
SI ET MEMBRE? :MOT :LISTE1 NON MEMBRE? :MOT
:LISTE2 [ECRIS VRAI] [ECRIS FAUX]
FIN

?COMPLEMENT [JE TU IL ELLE NOUS VOUS ILS
ELLES] [JE TU IL] "TU
FAUX
```

NOMBRE? ***obj***

(opération)
(pred)

rend VRAI si ***obj*** est un nombre, FAUX sinon.

```
?ECRIS NOMBRE? [TROIS]
FAUX

?ECRIS NOMBRE? 3
VRAI
```

ASCII *mot* (opération)
(n)

rend le code ASCII du premier caractère de ***mot***. Si ***mot*** est le mot vide, ASCII rend 0. Voir les codes ASCII en annexe B.

```
?ECRIS ASCII "A
65
```

Pour coder vos messages :

Tout d'abord coder un mot :

```
POUR CODE :MOT
SI VIDE? :MOT [RENDS []] [RENDS MP ASCII PREM!
:MOT CODE SP :MOT]
FIN

?ECRIS CODE "JEAN
74 69 65 78

POUR CODER :MES
SI VIDE? :MES [RENDS "] [RENDS PH CODE PREM :MES CODER!
SP :MES]
FIN

?ECRIS CODER [JEAN EST DANS LA CHAMBRE]
74 69 65 78 69 83 84 68 65 78 83 76 65 67
72 65 77 66 82 69
```

CAR *n*
(CARactère) (opération)
(car)

rend le caractère dont le code ASCII est ***n***, ***n*** étant un entier positif. Si $n > 255$, il est réduit modulo 256. Si $n = 0$, CAR rend le mot vide.
Si $1 < n < 31$ CAR rend des caractères non imprimables et si $128 < n < 255$ CAR rend un caractère utilisateur (voir Annexe B).

```
?ECRIS CAR 91
[

POUR ALPHABET :DEBUT :FIN :LISTE
SI PLG? :DEBUT :FIN [RENDS :LISTE]
RENDS ALPHABET SOMME 1 :DEBUT :FIN MD CAR!
:DEBUT :LISTE
FIN

?ECRIS ALPHABET 65 90 []
A B C D E F . . . . . Z
```


COMPTE *liste* (opération)
(n)

rend le nombre de membres de liste.

```
?ECRIS COMPTE [COMBIEN DE MOTS?]
3
```

Cette procédure permet de compter le nombre de lettres dans un mot :

```
POUR COMPTER :MOT
SI VIDE? :MOT [RENDS 0] [RENDS 1 + COMPTER!
SP :MOT]
FIN

?ECRIS COMPTER "ANTICONSTITUTIONNELLEMENT
25
```

2.2 Démonter

PREM *obj*
(PREMier) (opération)
(obj)

Si ***obj*** est une liste, rend le premier membre de cette liste.
Si ***obj*** est un mot, rend le premier caractère du mot.
PREM n'aime ni la liste vide, ni le mot vide.

```
?ECRIS PREM "SALUT
S

?ECRIS PREM [SALUT A TOUS]
SALUT
```

SP *obj*
(Sauf Premier) (opération)
(obj)

Si ***obj*** est une liste, rend la liste sans son premier membre.
Si ***obj*** est un mot, rend le mot sans son premier caractère.
SP n'aime ni la liste vide, ni le mot vide.

```
?ECRIS SP "SALUT
ALUT

?ECRIS SP [SALUT A TOUS]
A TOUS
```

DER ***obj*** (opération)
(DERnier) (obj)

Si ***obj*** est une liste, rend le dernier membre de cette liste.
Si ***obj*** est un mot, rend le dernier caractère du mot.
DER n'aime ni la liste vide, ni le mot vide.

```
?ECRIS DER "SALUT
T
```

SD ***obj*** (opération)
(Sauf Dernier) (obj)

Si ***obj*** est une liste, rend la liste sans son dernier membre.
Si ***obj*** est un mot, rend le mot sans son dernier caractère.
SD n'aime ni la liste vide, ni le mot vide.

```
?ECRIS SD "SALUT
SALU
?ECRIS SD [SALUT A TOUS]
SALUT A
```

Ecrivons un mot à l'envers :

```
POUR ENVERS :MOT
SI VIDE? :MOT [ECRIS [ ] STOP]
TAPE DER :MOT ENVERS SD :MOT
FIN

?ENVERS "BONJOUR
RUOJNOB
```

ITEM ***n liste*** (opération)
(obj)

rend le ***n***-ième membre de ***liste***. ***n*** doit être en entier positif, inférieur ou égal au nombre de membres de ***liste***.

```
?ECRIS ITEM 2 [COMBIEN DE MOTS ?]
DE
```

Pour tirer un membre au hasard dans une liste :

```
POUR TIRE :LIS
ECRIS ITEM HASARD 1 + COMPTE :LIS :LIS
FIN
```

Par exemple :

```
?TIRE [JEAN FRANCOIS CLAUDE LUCIE LAURENCE]
CLAUDE
```


Cette procédure permet de calculer la première occurence d'une lettre dans un mot :

```
POUR OCC1 :LET :MOT
SI VIDE? :MOT [RENDS [PAS DANS CE MOT]]
SI EGAL? PREM :MOT :LET [RENDS 1] [RENDS!
SOMME 1 OCC1 :LET SP :MOT]
FIN

?ECRIS OCC1 "A "TOURNANT
6
```

2.3 Construire

MOT ***mot1 mot2*** (opération)
(mot)

rend le mot obtenu en mettant ***mot2*** à la suite de ***mot1***, c'est-à-dire en soudant ***mot1*** et ***mot2***.
Souder le mot vide à un mot ne change pas ce mot.

```
?ECRIS MOT "BON "JOUR
BONJOUR
```

PH ***obj1 obj2*** (opération)
(PHrase) (liste)

rend une liste construite à partir de ***obj1*** et ***obj2***.
exemple : PH [TRIC] [TRAC] rend [TRIC TRAC]

```
?ECRIS PH "BONJOUR [TOUT LE MONDE]
BONJOUR TOUT LE MONDE
```

Si ***obj1*** et ***obj2*** sont deux listes, PH rend la liste dont les membres sont les membres de ***obj1*** suivis des membres de ***obj2*** (soude les deux listes).

Si ***obj1*** et ***obj2*** sont des mots, PH agit comme LISTE (voir LISTE).

Si ***obj1*** est un mot et ***obj2*** est une liste, PH agit comme MP (voir MP).

Si ***obj1*** est une liste et ***obj2*** est un mot, PH ***obj1 obj2*** est équivalent à MD ***obj2 obj1***.

```
PH [LE SOLEIL] [DE MINUIT]
rend [LE SOLEIL DE MINUIT]

PH "LUNE "ROUSSE
rend [LUNE ROUSSE]
```

TT TONNERRE [DE BREST]
rend [TONNERRE DE BREST]

PH [LE CHIEN EST UN ANIMAL] "SYMPATHIQUE
rend [LE CHIEN EST UN ANIMAL SYMPATHIQUE]

LISTE *obj1 obj2* (opération)
(liste)

rend la liste dont le premier membre est ***obj1*** et le second ***obj2***.

LISTE [UNE BELLE] [BANDE DESSINEE] rend la liste [[UNE BELLE] [BANDE DESSINEE]]

Ces primitives permettent principalement d'agrandir des listes ; en voici un exemple :
Pour réaliser une procédure qui se comporte comme FPOS, mais qui prend pour argument non pas une liste mais deux nombres :

```
?POUR FPOSBIS :NOMBRE1 :NOMBRE2
FPOS LISTE :NOMBRE1 :NOMBRE2
FIN

?FPOSBIS -34 55
?EC POS
-34 55
```

MP *obj liste* (opération)
(Met en Premier) (liste)

rend une liste formée de ***liste*** à laquelle on a ajouté ***obj*** comme premier membre.

MP "A [ ] rend la liste [A]
MP [PARIS EST] [UNE CAPITALE] rend [[PARIS EST] UNE CAPITALE]

MD *obj liste* (opération)
(Met en Dernier) (liste)

rend une liste formée de ***liste*** à laquelle on a ajouté ***obj*** comme dernier membre.

MD [TANTE PIM] [PAM POUM] rend la liste [PAM POUM [TANTE PIM]]

Chapitre 3

Les primitives opérant sur les valeurs logiques

Il s'agit des opérations classiques sur les valeurs logiques VRAI et FAUX. Les données des primitives décrites dans ce chapitre sont donc des résultats de prédicats.
On dit que le résultat d'un prédicat est vrai quand c'est le mot VRAI et faux quand c'est le mot FAUX.

Liste des prédicats décrits dans d'autres chapitres :

Prédicat	Chapitre
BC?	1
BOUTON?	9
CONTACT?	9
EGAL? (=)	2
LISTE?	2
MEMBRE?	2
MOT?	2
NOM?	6
NOMBRE?	2
PLG? (>)	7
PLP? (<)	7
PRIM?	4
PROC?	4
TOUCHE?	9
VIDE?	2
VISIBLE?	1

ET *pred1 pred2* (opération)
(pred)

rend VRAI si ***pred1*** et ***pred2*** sont tous deux vrais simultanément, FAUX sinon.

```
?ECRIS ET EGAL? 3 (2 + 1) EGAL? 8 (2 * 4)
VRAI

?ECRIS ET "VRAI "FAUX
FAUX
```

OU *pred1 pred2* (opération)
(pred)

rend VRAI si ***pred1*** ou bien ***pred2*** est vrai, (ou si les deux sont vrais), FAUX sinon.

```
?ECRIS OU EGAL? 3 (2 + 1) EGAL? 8 (2 * 3)
VRAI
```

Pour que la tortue ne sorte pas de son écran :

```
POUR REBONDIS
SI OU PREM POS > 80 PREM POS < -80 [TD 90]
SI OU DER POS < -80 DER POS > 80 [TD 90]
AVANCE 1
REBONDIS
FIN
```

NON *pred* (opération)
(pred)

rend la valeur logique opposée à celle de ***pred***. Si ***pred*** est vrai, rend FAUX, si ***pred*** est faux, rend VRAI.

```
?ECRIS NON EGAL? (4 + 1) (21 - 6)
VRAI
```

VRAI (opération)
(pred)

rend toujours le mot VRAI. Ceci autorise de ne pas faire précéder le mot VRAI de guillemet.

```
?ECRIS VRAI
VRAI
```

FAUX (opération)
(pred)

rend toujours le mot FAUX. Ceci autorise de ne pas faire précéder le mot FAUX de guillemet.

```
?ECRIS FAUX
FAUX
```


Chapitre 4
Les procédures

Les primitives présentées dans ce chapitre permettent de définir et d'examiner des procédures.

4.1 Définir des procédures

POUR *mot donnée1 donnée2...* (commande)

Sert à définir une procédure appelée ***mot*** avec, s'il y a lieu, des données.
Chaque nom de donnée doit être précédée de deux-points (:).
Quand on utilise POUR en mode direct, le symbole > apparaît en début de ligne pour indiquer que l'on est en mode de définition de procédure : les instructions tapées sont mémorisées sans être exécutées.

Exemple de définition de procédure en mode direct :

```
?POUR CARRES :COTE
>REPETE 4 [AVANCE :COTE TD 90]
>FIN
VOUS VENEZ DE DEFINIR CARRES
```

Une procédure ne peut pas être définie à l'intérieur d'une autre. On peut néanmoins utiliser POUR et amorcer ainsi une nouvelle définition.

```
?POUR DEFINITION
>ECRIS [NOUVELLE PROCEDURE]
>POUR NOUVELLE
>FIN
VOUS VENEZ DE DEFINIR DEFINITION

?DEFINITION
NOUVELLE PROCEDURE
>ECRIS "SALUTATIONS
>FIN
VOUS VENEZ DE DEFINIR NOUVELLE
```

```
?IM "DEFINITION
POUR DEFINITION
ECRIS [NOUVELLE PROCEDURE]
POUR NOUVELLE
FIN

?IM "NOUVELLE
POUR NOUVELLE
ECRIS "SALUTATIONS
FIN
```

FIN (mot spécial)

sert à indiquer la fin de la définition d'une procédure. FIN n'a de sens qu'en mode de définition de procédure, et il doit apparaître seul sur une ligne. Dès qu'une procédure a été définie, un message Logo apparaît :

VOUS VENEZ DE DEFINIR...

on est à nouveau en "mode direct".

Avec l'éditeur (voir Chapitre 11) on peut définir plusieurs procédures les unes à la suite des autres, en n'oubliant pas de terminer chaque définition de procédure par la ligne FIN.

Pour définir une procédure à partir d'une liste :

```
POUR DEFINIS :LISTE
SORTIE 4
ECRISMEMBRES :LISTE
SORTIE 1
ENTREE 4
FIN

POUR ECRISMEMBRES :LISTE
SI VIDE? :LISTE [STOP]
ECRIS PREM :LISTE
ECRISMEMBRES SP :LISTE
FIN
```

La donnée de DEFINIS est une liste dont les membres sont les différentes lignes de la procédure à définir.

```
?DEFINIS [[POUR TRIANGLE :X] [REPETE 3 [AVANCE :X TD 120]] [FIN]]
```


4.2 Examiner

PROC? *mot* (opération)
(PROCédure?) (pred)

rend VRAI si ***mot*** est le nom d'une procédure utilisateur définie dans l'espace de travail (voir Chapitre 8), FAUX sinon.

```
PROC? "AVANCE rend FAUX.
```

PRIM? *mot* (opération)
(PRIMitive?) (pred)

rend VRAI si ***mot*** est le nom d'une primitive du langage Logo, FAUX sinon.

```
PRIM? "AVANCE rend VRAI
```

Les procédures suivantes permettent d'obtenir sous forme d'une liste le texte d'une procédure déjà définie.

```
POUR TEXTE :PROCEDURE
SORTIE 4
IM :PROCEDURE
SORTIE 1
ENTREE 4
RENDS LISMEMBRES [ ]
FIN

POUR LISMEMBRES :LISTE
DONNE "LISTE MD LL :LISTE
SI EGAL? DER :LISTE [FIN] [RENDS :LISTE]
RENDS LISMEMBRES :LISTE
FIN

?ECRIS TEXTE "CARRE
[POUR CARRE] [REPETE 4 [AVANCE 50 TD 90]]
[FIN]
```