Apple® Logo II
Manuel de référence

L'Apple II

Compatible
IIe (128Ko), IIc

Limitation de la garantie et de la responsabilité

Bien qu'Apple ait testé les programmes décrits dans ce manuel et revu leur contenu, ni Apple ni ses fournisseurs de logiciels n'offrent de garantie, expresse ou tacite, concernant ce manuel ou les programmes qui y sont décrits, leur qualité, leurs performances ou leur capacité à satisfaire à quelque application particulière que ce soit.

En conséquence, ces programmes et ce manuel sont vendus "tels quels" et l'acheteur supporte tous les risques en ce qui concerne leur qualité ou leur fonctionnement. Apple ou ses fournisseurs de logiciels ne pourront en aucun cas être tenus pour responsables des préjudices directs ou indirects, de quelque nature que ce soit, résultant d'une imperfection dans les programmes ou le manuel, même s'ils ont été avisés de la possibilité de tels préjudices. En particulier, ils ne pourront encourir aucune responsabilité du fait de programmes ou données mémorisés ou exploités sur des produits Apple, y compris pour les coûts de récupération ou de reproduction de ces programmes ou données.

L'acheteur a toutefois droit à la garantie légale, dans les cas et dans la mesure seulement où la garantie légale est applicable nonobstant toute exclusion ou limitation.

Droit de reproduction

Vous pouvez utiliser le logiciel sur n'importe quel ordinateur vous appartenant, mais vous ne pouvez effectuer de copies dans ce but. Pour certains produits, il est possible d'acheter une licence multi-usages, permettant d'utiliser le logiciel sur plus d'un ordinateur appartenant à l'acheteur, y compris sur un système de disque partagé (veuillez contacter votre distributeur agréé Apple pour plus d'informations sur les licences multi-usages).

Révisions des produits

Apple ne peut garantir que vous soyez informé des révisions opérées sur le logiciel décrit dans ce manuel, même si vous avez retourné la carte d'enregistrement fournie avec le produit. Il vous est recommandé de vous en informer périodiquement auprès de votre concessionnaire agréé Apple.

Apple Computer France
Avenue de l'Océanie
Z.A. de Courtabœuf
B.P. 131 91944 Les Ulis

Apple® Logo II
Manuel de référence

L'Apple II

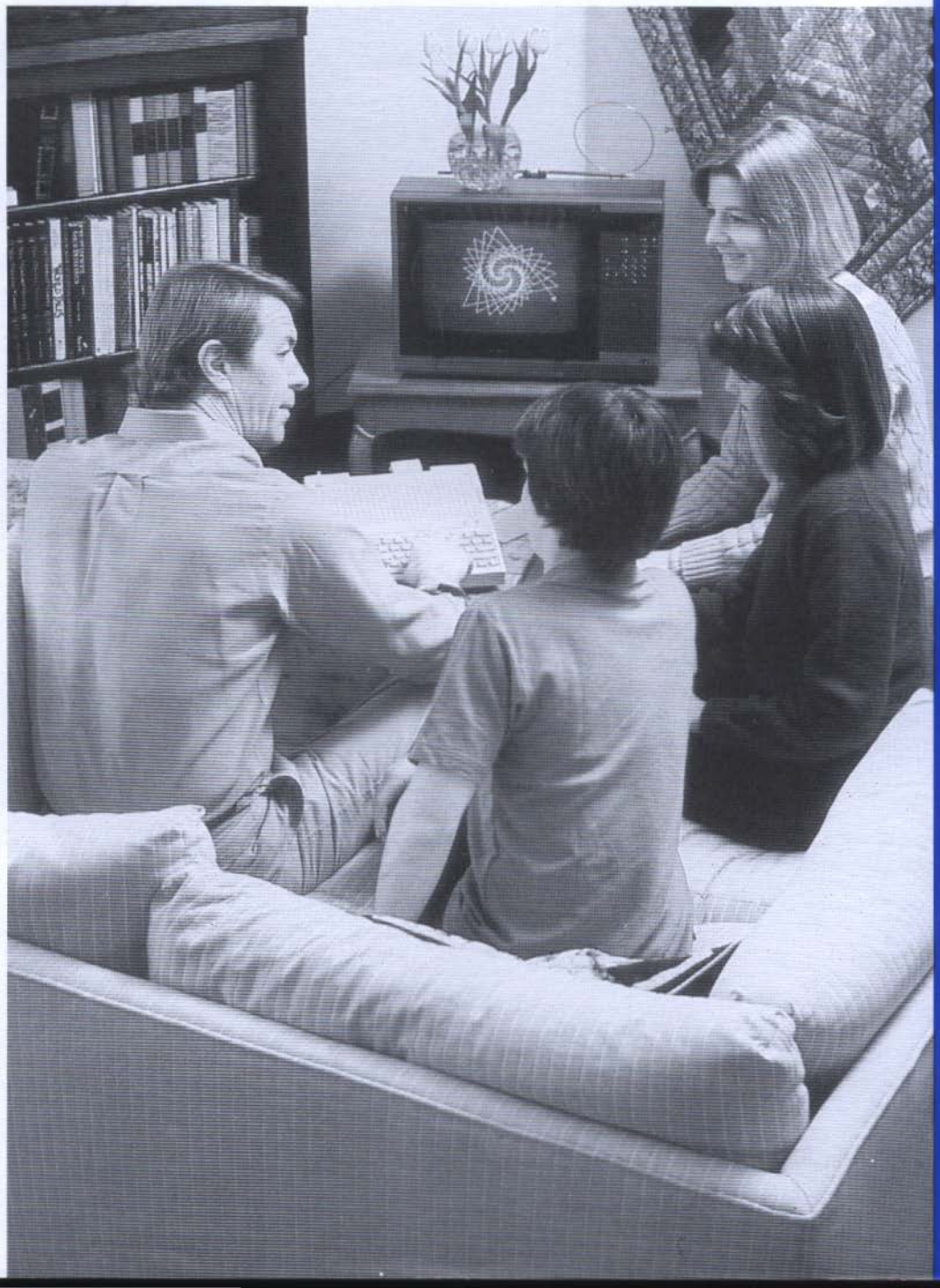


Table des matières

Répertoire des tableaux

Le manuel de référence Apple Logo II

Le présent manuel décrit de façon détaillée comment utiliser Apple Logo II et doit véritablement servir d'outil de consultation. Le manuel qui l'accompagne, *Introduction à la programmation*, se veut un guide pratique qui vous permet de vous familiariser avec Logo et qui vous initie à ses principales caractéristiques.

Ce manuel de référence vous donne une brève description de chacune des primitives du langage Logo accompagnée d'exemples de programmes (procédures). Les titres des chapitres énumérés dans la table des matières donnent un bref aperçu du contenu de chacun des chapitres.

Utilisation de ce manuel

Voici quelques suggestions sur la façon de vous y prendre.

A qui s'adresse ce manuel ?	Ce manuel est destiné aux personnes qui connaissent déjà Logo ou un langage du même type.
Apprendre les principes fondamentaux	Vous devriez consulter le manuel *Introduction à la programmation.*
Obtenir un aperçu des règles de la grammaire Logo	Vous devriez lire le chapitre 2 avant d'aborder le manuel de référence.

Trouver une primitive convenant à une tâche particulière	Référez-vous aux titres des chapitres énumérés dans la table des matières ou au tableau des primitives se trouvant à la fin du manuel. Les primitives y sont regroupées par catégories ; il vous sera plus facile de trouver ce que vous cherchez.
Ce qu'une primitive accomplit	Référez-vous à l'annexe G ou à l'index.
Des renseignements supplémentaires sur Logo	Référez-vous au glossaire pour obtenir la définition ou l'explication d'un mot. L'index vous permet de trouver facilement ce que vous cherchez.
Logo peut vous venir en aide	Maintenez la touche [Ó] ou [ ] enfoncée et pressez [?]. Cela peut s'effectuer en tout temps sauf lorsque vous faites exécuter une procédure. Des renseignements seront affichés.
Logo vous aide à obtenir des indications sur une primitive	Tapez AIDE suivie du nom de la primitive puis pressez [↵]. Le nom de la primitive doit être précédé des guillemets ("). Les données nécessaires à la primitive seront affichées.
Des renseignements supplémentaires sur l'Apple IIe ou l'Apple IIc	Référez-vous au manuel de l'utilisateur fourni avec votre ordinateur.

Repères visuels

Les définitions de procédures et les exemples d'interaction entre Logo et vous apparaissent en un caractère différent de celui qui est employé tout au long du manuel. Ce type de caractère est semblable à celui qui apparaît à l'écran.

D'autres repères visuels se trouvent au fil des chapitres.

Lorsqu'on représente deux touches reliées par un trait d'union, celles-ci doivent être pressées simultanément. Ainsi [pomme] - [↓] signifie que vous devez presser [pomme] et [↓], en même temps. En fait, vous maintenez la touche [pomme] enfoncée et pressez [↓].

Les notes inscrites dans la marge vous reportent à d'autres parties du manuel où vous trouverez des renseignements supplémentaires.

Note : Une section ombragée comme celle-ci contient des conseils ou des renseignements qui peuvent vous être d'une grande utilité.

Attention
Certains paragraphes que vous reconnaîtrez par un triangle vous signalent que des difficultés peuvent survenir.

Le tableau P-1 montre ce qui est affiché à l'écran à la mise en route de Logo.

Tableau P-1. *Ecran Logo*

```
© 1984
LES SYSTEMES D'ORDINATEUR LOGO INC.
á - ? AIDE

BIENVENUE A LOGO
?_
```

Note : Si vous disposez d'un Apple IIe, le symbole [á] figurant dans le message ci-dessus peut être remplacé par la lettre A apparaissant en vidéo inverse. Cette représentation a la même signification que le symbole [á]. Cela est dû au fait que le matériel informatique que vous utilisez peut avoir certaines particularités ; des ordinateurs sont munis d'une puce comportant un générateur de caractères, d'autres non.

Introduction

Introduction

Logo est un langage de programmation conçu pour satisfaire les besoins d'un utilisateur débutant et les exigences d'un programmeur expérimenté. Les caractéristiques de Logo vous permettent d'utiliser le graphique Tortue, pour ainsi créer rapidement et aisément des dessins étonnants, de manipuler des listes et de gérer des fichiers.

Dans ce chapitre, vous trouverez :

- une liste du matériel dont vous avez besoin pour utiliser Apple Logo II ;
- des façons de demander de l'aide à Logo ;
- les règles à suivre pour donner des instructions Logo ;
- une explication sur la façon dont les primitives Logo sont décrites tout au long du manuel.

Matériel requis

Pour utiliser Apple Logo II, vous devez posséder :

- un micro-ordinateur Apple IIc ou Apple IIe muni d'une carte d'extension de mémoire à 80 colonnes, et une unité de disquette. L'Apple IIc a une unité de disquette incorporée ; vous pouvez utiliser une unité de disquette supplémentaire si vous le désirez ;
- un périphérique à écran de visualisation, soit un moniteur, soit un téléviseur ;

- la disquette *Apple Logo II* ;
- une imprimante compatible avec votre ordinateur, si vous désirez imprimer votre travail.

Si vous disposez d'un Apple IIc, vous pouvez utiliser l'imprimante Imagewriter d'Apple pour imprimer du texte et des dessins. Vous devez connecter l'imprimante à la prise marquée d'un 1 à l'arrière de votre ordinateur.

Si vous disposez d'un Apple IIe, vous pouvez utiliser l'imprimante Imagewriter ou une imprimante par matrice de points Apple. Les deux types d'imprimante peuvent servir à imprimer du texte mais vous devrez utiliser des commandes spéciales pour imprimer des dessins à l'aide de l'imprimante par matrice de points.

Pour plus de renseignements sur les imprimantes, référez-vous au manuel de l'utilisateur fourni avec votre ordinateur Apple.

D'autres imprimantes compatibles avec votre ordinateur vous permettront d'imprimer du texte et, dans certains cas, des dessins.

Ecrans d'aide

Logo vous offre deux façons d'obtenir de l'aide ; l'une vous donne des renseignements d'ordre général et l'autre des indications sur une primitive Logo spécifique.

Pour obtenir des renseignements d'ordre général sur Logo, pressez [Ó] - [?]. Logo affiche un écran d'aide selon que vous vous trouvez au niveau supérieur ou en mode d'édition :

- Au niveau supérieur, l'écran d'aide contient des renseignements sur les commandes graphiques, l'utilisation de l'éditeur, la définition d'une procédure et les touches spéciales.
- En mode d'édition, l'écran d'aide vous renseigne sur les touches d'édition.

Avant que ne s'affiche l'écran d'aide, Logo conserve tout ce qui apparaissait à l'écran. Logo affiche l'écran d'aide à 40 colonnes. Vous pouvez faire défiler son contenu en utilisant [↑] ou [↓] ; en pressant [Ó] - [ESC], Logo affiche de nouveau ce qui a été conservé avant que vous ne demandiez de l'aide.

Pour obtenir des renseignements au sujet d'une primitive Logo, vous devez taper `AIDE` suivie du nom de la primitive. Cette dernière doit être précédée des guillemets ("). Logo affiche alors les données dont la primitive a besoin.

Instructions Logo

Cette section contient des indications qui vous permettent de taper vos instructions en lettres majuscules ou en lettres minuscules; celle-ci vous fournit aussi des renseignements sur l'utilisation de certaines touches du clavier.

Lorsque vous tapez un mot, Logo ne fait pas la distinction entre les majuscules et les minuscules. Vous n'avez donc pas besoin de vous en soucier. Par exemple, si vous définissez une procédure dont le nom est `CARRE`, puis demandez à Logo de l'exécuter, ce dernier obéira, que vous ayez tapé le nom de la procédure sous la forme `CARRE`, `Carré` ou `carré`.

Une liste complète des touches d'édition se trouve au chapitre 4 intitulé "Utilisation de l'éditeur Logo".

Le tableau 1-1 donne une liste des touches ainsi que leur fonction respective lorsque vous vous trouvez au niveau supérieur ou en mode d'édition.

Tableau 1-1. *Touches spéciales et touches d'édition*

Touche	Fonction
←	Déplace le curseur d'une position vers la gauche.
→	Déplace le curseur d'une position vers la droite.
ó - ←	Déplace le curseur d'un mot vers la gauche.
ó - →	Déplace le curseur d'un mot vers la droite.
ó - <	Déplace le curseur au début de la ligne où celui-ci se trouve.
ó - >	Déplace le curseur à la fin de la ligne où celui-ci se trouve.

Tableau 1-1. *Touches spéciales et touches d'édition (suite)*

Touche	Fonction
CONTROL - D ou DELETE	Efface le caractère placé à la gauche du curseur.
CONTROL - F	Efface le caractère situé à la position du curseur.
CONTROL - X	Efface tous les caractères de la ligne.
CONTROL - Y	Efface les caractères, de la position du curseur, jusqu'à la fin de la ligne.
CONTROL - R	Insère la dernière ligne que vous avez tapée ou que vous avez effacée à l'aide de CONTROL - Y.
Ó - ?	Affiche des renseignements très utiles.
↵	Demande à Logo d'effectuer vos directives.

Description des primitives

La description de chacune des primitives s'élabore comme suit :

- Vous trouvez d'abord le nom de la primitive ainsi que le nombre et le type de données dont elle a besoin. Les mots de données sont décrits à la fin de ce chapitre.
- La forme abrégée de la primitive, si elle existe, se trouve entre parenthèses.
- On vous indique le type de primitive dont il s'agit, soit une commande, soit une opération, soit une opération de forme infixée.

Dans le cas de certaines primitives comme SOMME, l'utilisation des parenthèses peut être nécessaire. Cette primitive peut alors recevoir un nombre illimité de données. Lorsque plus de deux données sont utilisées, vous devez placer une parenthèse ouverte avant le nom de la primitive et une parenthèse fermée après la dernière donnée.

Le tableau 1-2 donne une liste des mots utilisés dans la syntaxe des primitives Logo. Les mots représentent le type de donnée que nécessite une primitive.

Tableau 1-2. *Mots de données*

Mot de donnée	Description
adr	une adresse (emplacement) de la mémoire
car	lettres de l'alphabet, chiffres et signes de ponctuation
champ	un entier qui donne le nombre d'éléments dans un nombre
<coorx coory>	une liste de deux nombres correspondant aux coordonnées de la Tortue
degrés	un nombre indiquant les degrés d'un angle
distance	un nombre
donnée	mot précédé des guillemets utilisé avec POUR
durée	un entier compris entre 0 et 65,535
entier	un entier sans décimales ; si vous tapez un nombre décimal, Logo ignore la décimale
fichier	un nom de fichier ou un numéro désignant un périphérique
fréq	un entier compris entre 3 et 65,535
largeur	un entier, soit 40, soit 80
liste	une liste de mots ou de listes contenue entre crochets < >
mot	suite de caractères sans espace

Tableau 1-2. *Mots de données (suite)*

Mot de donnée	Description
nomfichier	le nom d'un fichier sur la disquette
nom(liste)	un nom de procédure ou de variable, ou une liste de noms
nombre	un nombre réel ou un entier
<nocolonne noligne>	une liste de deux entiers qui correspond à la position du curseur
nocouleur	un entier compris entre 0 et 5 qui correspond à la couleur du crayon ou du fond
nomanette	un entier (0, 1, 2 ou 3)
obj	objet Logo : mot, liste ou nombre
octet	unité référentielle de donnée utilisée par l'ordinateur ; entier compris entre 0 et 255
précision	un entier compris entre 0 et 6 indiquant le nombre de chiffres après le point dans un nombre réel
préd	une opération qui retourne soit le mot VRAI, soit le mot FAUX
préfixe	un nom qui sert de préfixe à un fichier sur une disquette
prop	un mot.

Grammaire Logo

Grammaire Logo

Logo est un langage de programmation d'une grande flexibilité. Il est constitué de **procédures**. Certaines d'entre elles font partie intégrante du système Logo lui-même ; c'est pourquoi elles sont appelées **primitives**. D'autres procédures sont définies par l'utilisateur. Ces dernières ressemblent aux primitives faisant déjà partie du vocabulaire Logo.

Les procédures peuvent définir, modifier et faire exécuter d'autres procédures en obéissant aux règles de la grammaire Logo. Les sections qui suivent décrivent brièvement ces règles.

Procédures

Voici la définition de la procédure SALUER :

```
POUR SALUER                    ligne titre
ECRIS "BONJOUR
FIN
```

La ligne titre doit toujours commencer par le mot spécial POUR suivi du nom de la procédure. La dernière ligne ne doit contenir que le mot FIN. Dans ce cas-ci, SALUER demande d'exécuter la primitive ECRIS.

Il existe trois façons de définir une procédure :

- taper sa définition au niveau supérieur (celui-ci se reconnaît lorsque le point d'interrogation apparaît à l'écran, suivi du curseur) ;
- utiliser l'éditeur Logo ;
- utiliser la primitive DEFINIS.

Une fois que la procédure est définie, une première façon de la faire exécuter consiste à taper le nom de celle-ci au niveau supérieur :

```
?SALUER                    l'appel de procédure
BONJOUR                    son résultat
```

Une seconde façon de faire exécuter une procédure consiste à appeler la procédure déjà définie à l'intérieur de la définition d'une autre procédure. Voici un exemple :

```
POUR ACCUEILLIR
SALUER
SALUER
SALUER
SALUER
SALUER
FIN
```

Lorsqu'on appelle la procédure ACCUEILLIR, cette dernière exécute la procédure SALUER cinq fois.

```
?ACCUEILLIR
BONJOUR
BONJOUR
BONJOUR
BONJOUR
BONJOUR
```

Ainsi, la **superprocédure** ACCUEILLIR contient la **sous-procédure** SALUER. Vous pouvez créer des programmes très complexes en utilisant des superprocédures et des sous-procédures.

Une procédure peut aussi être une sous-procédure d'elle-même. On dit alors que cette procédure est **récursive**. Vous trouverez plusieurs exemples de cette caractéristique de Logo au fil des chapitres de ce manuel.

Si vous demandez à Logo d'exécuter une procédure non définie, un message est affiché.

```
?CHANTER
NE SAIS QUE FAIRE POUR CHANTER
```

Ponctuation et données d'une procédure

Logo interprète chaque mot comme une demande d'exécution de procédure. Vous devez utiliser des caractères spéciaux pour lui indiquer clairement que tel n'est pas toujours le cas.

Un mot commençant par des guillemets tel "BONJOUR avertit Logo de considérer ce mot comme un mot littéral et non comme un appel de procédure. Dans l'exemple suivant, "BONJOUR est la donnée de la primitive ECRIS.

```
?ECRIS "BONJOUR
BONJOUR
```

Les nombres sont considérés comme des mots littéraux, mais n'ont pas besoin d'être précédés des guillemets.

```
?ECRIS 5
5
```

Une **liste** est constituée d'une suite de mots placés entre crochets. Celle-ci peut être la donnée d'une procédure.

```
?ECRIS <PASSEZ UNE BONNE JOURNEE>
PASSEZ UNE BONNE JOURNEE
```

La liste <PASSEZ UNE BONNE JOURNEE> est une liste littérale ; Logo n'essaie pas de l'exécuter. Vous pouvez voir cela plus clairement dans l'exemple suivant :

```
?ECRIS <2 + 2>
2 + 2
```

Mais si la suite de mots n'est pas contenue entre crochets, Logo tentera de l'exécuter.

```
?ECRIS PASSEZ UNE BONNE JOURNEE
NE SAIS QUE FAIRE POUR PASSEZ
```

ou

```
?ECRIS 2 + 2
4
```

Les procédures que vous définissez peuvent aussi avoir des données. Par exemple :

```
POUR SALUTATION :NOM          ligne titre
EC "BONJOUR
EC :NOM
EC <PASSEZ UNE BONNE JOURNEE>
FIN
```

Le deux points (:) devant un mot indique à Logo que ce mot est le nom d'une variable. La variable contient la donnée d'une procédure et apparaît dans la ligne titre après le nom de la procédure. Ainsi, NOM est la variable dont la valeur est affichée lorsque la procédure SALUTATION est exécutée. On retrouve trois appels de la procédure ECRIS (ou son abréviation EC) à l'intérieur de la procédure SALUTATION ; le second appel utilise la valeur attribuée à NOM.

Voici un exemple de demande d'exécution de la procédure SALUTATION au niveau supérieur.

```
?SALUTATION "LOUISE
BONJOUR
LOUISE
PASSEZ UNE BONNE JOURNEE
```

Dans cet exemple, la donnée de SALUTATION est LOUISE. Logo en fait la valeur de NOM lorsqu'il exécute la procédure.

Commandes et opérations

En Logo, il existe deux types de procédures : les commandes et les opérations. Les **opérations** fournissent une valeur à une autre procédure tandis que les **commandes** (telles ECRIS) n'en fournissent pas.

La primitive SOMME est une opération dont le résultat est la somme de deux données numériques. Dans cet exemple, le résultat de SOMME est la donnée de la commande ECRIS :

```
?ECRIS SOMME 31 28
59
```

Puisqu'une opération ne peut être autre chose que la donnée d'une autre procédure, chaque ligne Logo doit commencer par une commande. Si tel n'est pas le cas, un message est affiché. Par exemple :

```
?SOMME 31 28
NE SAIS QUE FAIRE AVEC 59
```

Les procédures que vous définissez sont, elles aussi, soit des commandes, soit des opérations. Pour définir des procédures agissant comme des opérations, vous devez utiliser la primitive RETOURNE. La procédure suivante, TIRAGE, est une opération.

```
POUR TIRAGE
SI (HASARD 2) = 0 <RETOURNE "FACE>
RETOURNE "PILE
FIN
```

Cette procédure retourne le mot FACE si HASARD 2 donne 0 ou le mot PILE si HASARD 2 donne 1. Vous pouvez fournir le résultat de TIRAGE à ECRIS :

```
?EC TIRAGE
PILE
```

ou

```
?EC TIRAGE
FACE
```

Variables

Une **variable** en Logo est comparable à un contenant portant un nom et pouvant recevoir un **objet** (un mot, une liste ou un nombre) comme contenu. La variable est précédée du deux points ; la valeur qui lui a été attribuée est ainsi utilisée par la procédure. Par exemple :

```
ECRIS :JEAN
```

indique à Logo de chercher un contenant nommé JEAN. S'il en trouve un, Logo fournit ce qu'il contient à ECRIS. La primitive ECRIS affiche alors à l'écran le contenu (ou la valeur) de JEAN.

Si la variable JEAN n'existe pas, Logo affiche le message suivant :

```
JEAN N'A PAS DE VALEUR
```

Il existe deux façons d'attribuer une valeur à une variable :

Voir sous la rubrique "Ponctuation et données d'une procédure".

- Définir une procédure comportant des données et appeler celle-ci en attribuant des valeurs aux données.
- Utiliser la primitive RELIE ou NOMME.

La primitive RELIE nécessite deux données : un mot et une valeur.

```
?RELIE "JEAN 25
?ECRIS :JEAN
25
```

Dans ce cas-ci, le nombre 25 est la valeur. Ce pourrait être un mot ou une liste. En voici un exemple :

```
RELIE "X "JEAN
```

Dans ce cas-ci, la primitive RELIE a deux mots précédés des guilllemets comme données ; la commande place le mot littéral JEAN dans le contenant nommé X. Le contenu de la variable JEAN utilisé dans l'exemple précédent reste inchangé.

```
?ECRIS :X
JEAN
?ECRIS :JEAN
25
```

Variables globales et variables locales

Lorsque vous créez une variable à l'aide de RELIE, cette variable demeure dans l'espace de travail jusqu'à ce que vous l'effaciez. Il s'agit d'une **variable globale**. Il existe aussi des variables qui ne restent dans l'espace de travail que le temps pendant lequel la procédure est exécutée. Ce sont des **variables locales**. Les variables définies comme données de procédures sont des variables locales.

La procédure SALUTATION peut être modifiée de sorte que celle-ci affiche la date.

```
POUR SALUTATION :NOM
EC :DATE
EC "BONJOUR
EC :NOM
EC <PASSEZ UNE BONNE JOURNEE>
FIN
```

Ici, DATE n'apparaît pas dans la ligne titre de SALUTATION puisque c'est une variable globale. Vous pouvez définir la valeur de DATE au niveau supérieur.

```
?RELIE "DATE <LE 31 MAI 1984>
?SALUTATION "LOUISE
LE 31 MAI 1984
BONJOUR
LOUISE
PASSEZ UNE BONNE JOURNEE
```

La variable NOM n'est pas une variable globale. Une fois l'exécution de la procédure SALUTATION terminée, NOM ne contient plus de valeur, alors que la valeur qui avait été attribuée à DATE se trouve encore dans l'espace de travail.

Vous pouvez aussi utiliser RELIE pour définir la variable DATE à l'intérieur de la procédure SALUTATION. DATE demeurera une variable globale même après l'exécution de SALUTATION. La primitive LOCALE vous permet de créer des variables locales à l'intérieur d'une procédure.

Compréhension d'une ligne Logo

Une ligne Logo peut être plus longue qu'une ligne affichée à l'écran. Par exemple :

```
?RELIE "PLUSIEURSNOMS <LOUISE ALAIN LIS!
E JULIEN RICHARD LORRAINE>
```

Le point d'exclamation (!) indique que la ligne Logo se continue sur la ligne suivante. Lorsqu'on tape une ligne Logo au niveau supérieur, celle-ci contient un maximum de 125 caractères en comptant les espaces. Une ligne Logo se termine en pressant ↵.

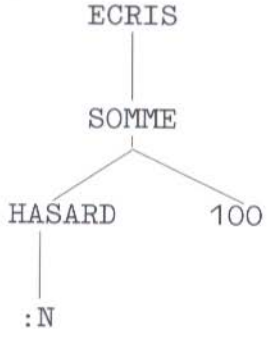
Voici quelques directives facilitant la compréhension d'une ligne Logo complexe :

- Le premier mot d'une ligne Logo doit toujours être une commande.
- Une opération est toujours la donnée d'une autre procédure.
- Chacune des données d'une procédure doit comporter une valeur.
- Quand toutes les données d'une commande ont été exécutées, la procédure suivante doit être une autre commande.

Voici un exemple d'une ligne Logo complexe :

```
ECRIS SOMME HASARD :N 100
```

Le résultat de l'opération SOMME est la donnée de la commande ECRIS. Deux données sont fournies à SOMME : la première est le résultat de HASARD qui, à son tour, nécessite une donnée (soit la valeur attribuée à N). La seconde donnée de SOMME est 100.



Si vous donnez la valeur 10 à N,

```
?RELIE "N 10
```

et que vous considérez la ligne Logo suivante, le résultat sera un nombre compris entre 100 et 109.

```
?ECRIS SOMME HASARD :N 100
101
```

Définition des procédures à l'aide de POUR

21 POUR
22 FIN

Définition des procédures à l'aide de POUR

En utilisant la primitive POUR, vous pouvez définir vos procédures au niveau supérieur ; le contenu de l'écran reste inchangé. Cette méthode est utile si vous devez vous référer à des instructions qu'affiche l'écran au moment où vous tapez la définition de votre procédure.

POUR

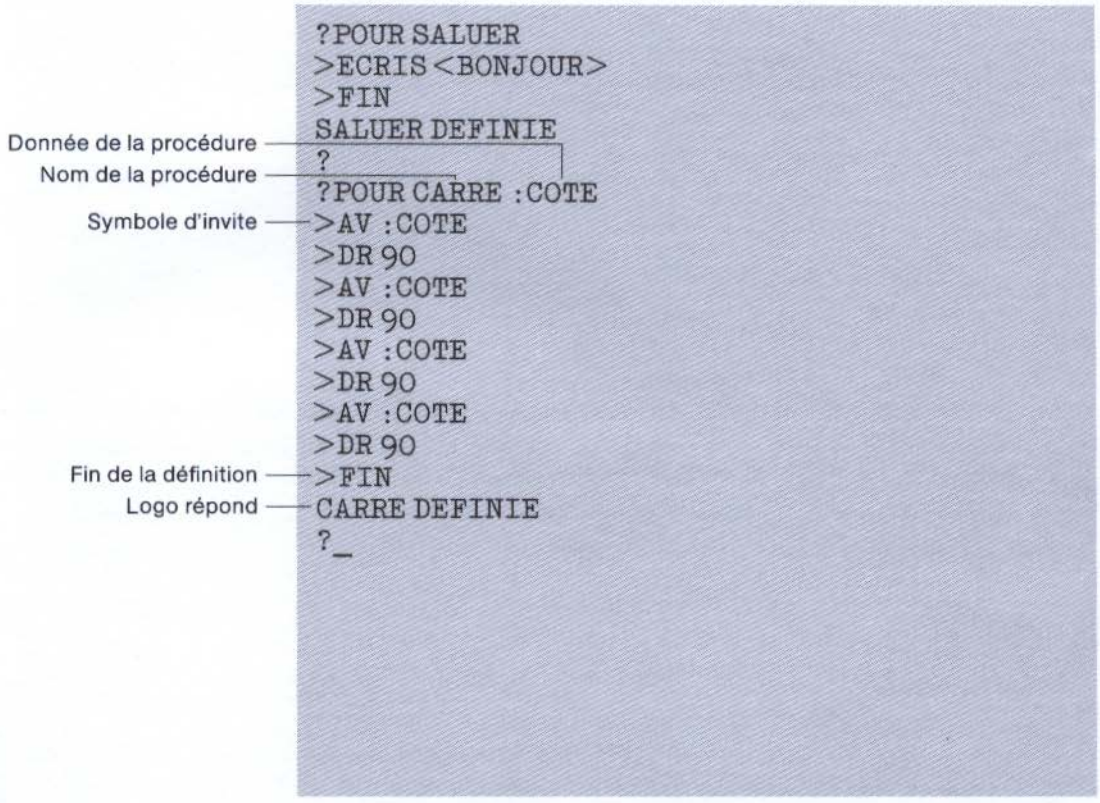
POUR *nom donnée1 donnée2...* (commande)

POUR indique à Logo que vous êtes en train de définir une procédure appelée *nom*, avec, s'il y a lieu, les données indiquées. Au niveau supérieur, le symbole d'invite change de ? à > pour vous rappeler que vous êtes en train de définir une procédure. Logo n'exécute pas les instructions que vous tapez ; il les enregistre comme faisant partie de la définition de votre procédure.

Note : On ne doit pas mettre les guillemets avant *nom* ; POUR le fait automatiquement.

Tapez le mot `FIN` pour indiquer à Logo que vous avez terminé la définition de la procédure et pour retourner au niveau supérieur. Le mot spécial FIN doit apparaître seul sur la dernière ligne de la définition.

Exemples :



Lorsque vous définissez une procédure à l'aide de POUR, rappelez-vous qu'en appuyant sur [ó] - [ESC], vous interrompez le processus de définition. Vous ne pouvez modifier, au niveau supérieur, une procédure que vous avez définie à l'aide de POUR. Vous devez utiliser la commande EDITE ou alors effacer la procédure que vous voulez changer à l'aide de EFFACE (EF) et redéfinir celle-ci.

FIN

FIN (mot spécial)

Lorsque vous utilisez POUR, il est essentiel d'employer FIN pour signaler à Logo que vous avez terminé la définition de la procédure. Le mot FIN doit se trouver seul sur la dernière ligne. Dans l'éditeur Logo, vous devez utiliser FIN pour séparer les procédures si vous en tapez plusieurs les unes à la suite des autres.

Utilisation de l'éditeur Logo

Utilisation de l'éditeur Logo

L'éditeur Logo est interactif à la manière d'un éditeur de texte de type page-écran. Il fournit un moyen efficace de définir et de modifier les procédures. La commande EDITE donne accès à l'éditeur Logo.

Ce chapitre présente :

- le fonctionnement de l'éditeur ;
- les particularités de la commande EDITE ;
- les combinaisons de touches et les méthodes d'édition ;
- des primitives autres que EDITE qui ont trait à l'édition.

Fonctionnement de l'éditeur

Lorsque vous faites appel à l'éditeur, Logo utilise un autre type d'écran. Par exemple

```
?EDITE "POLY
```

```
                    EDITEUR LOGO
==========================================
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN









------------------------------------------
 ó-A accepte, ó-? AIDE, ó-ESC annule
```

Il n'y a plus de symbole d'invite, mais le curseur vous montre l'endroit où vous allez inscrire un caractère.

Note : La procédure POLY ne s'interrompra que lorsque vous appuierez sur les touches ó - ESC.

La procédure que vous éditez se trouve dans un espace appelé **mémoire tampon**. Lorsque vous entrez en mode d'édition, Logo affiche à l'écran le texte contenu dans cette mémoire. Dans l'éditeur Logo, une page-écran peut comprendre un maximum de 20 lignes.

Vous pouvez déplacer le curseur n'importe où dans le texte en utilisant les touches de fonction qui seront décrites plus loin dans ce chapitre. Il vous est aussi possible d'insérer ou d'effacer des caractères à l'aide des combinaisons de touches appropriées.

Chaque touche que vous pressez provoque une action au niveau de l'éditeur. La plupart des caractères du clavier (alphabétiques, numériques, de ponctuation et ↵) sont insérés à l'endroit où se trouve le curseur.

Lorsque vous appuyez sur la touche ↵, le curseur, ainsi que le texte qui suit, se déplace jusqu'à la ligne suivante. Vous pouvez alors taper de nouvelles instructions.

Vous pouvez taper une ligne contenant plus de caractères que ne peut en afficher une ligne d'écran. Lorsque vous arrivez à la fin de la ligne d'écran, n'appuyez pas sur la touche ↵; continuez à taper. Un point d'exclamation (!) apparaît à l'endroit qu'occupe le caractère à l'extrême droite de l'écran. Le curseur se déplace alors à la ligne suivante où s'écrit le reste des caractères de la ligne précédente.

La même règle s'applique hors de l'éditeur.

Exemple :

```
?POUR ECRMESSAGES :PERSONNE
>EC PHRASE :PERSONNE<, JE VAIS TAPER UN !
TRES LONG MESSAGE POUR VOUS>
>EC PHRASE <A BIENTOT,> :PERSONNE
>FIN
ECRMESSAGES DEFINIE
?_
```

L'éditeur dispose d'une mémoire auxiliaire de ligne appelée **tampon réserve**. Vous pouvez l'utiliser pour déplacer des lignes dans une procédure ou pour les insérer à différents endroits. Le tampon réserve peut contenir un maximum de 125 caractères alors que la mémoire tampon peut contenir jusqu'à 6144 caractères.

CONTROL - X enlève une ligne entière et CONTROL - Y, une portion de ligne; ces lignes sont placées dans le tampon réserve. CONTROL - R insère, à l'endroit désigné par le curseur, la ligne ou la portion de ligne gardée dans le tampon réserve.

Vous pouvez passer de l'écran graphique à l'écran d'édition en utilisant respectivement CONTROL - L et CONTROL - T. Le passage d'un mode à l'autre ne modifie pas le contenu de chacun des écrans.

Lorsque vous quittez l'éditeur à l'aide de ó - A, Logo lit les lignes contenues dans la mémoire tampon comme si vous les aviez tapées au niveau supérieur.

Logo traite la définition d'une procédure (les instructions précédées de la ligne titre et du mot spécial POUR) contenue dans la mémoire tampon comme si vous l'aviez tapée au niveau supérieur. Si la mémoire tampon contient une définition dans laquelle vous n'avez pas inclus le mot spécial FIN, ce dernier sera inscrit automatiquement.

Si la mémoire d'édition contient des instructions Logo qui ne font pas partie d'une définition de procédure, elles seront exécutées lorsque vous quitterez l'éditeur.

Vous pouvez définir plus d'une procédure à la fois lorsque vous êtes dans l'éditeur. Les définitions terminées, vous pouvez retourner à l'écran graphique où vous retrouverez vos derniers graphiques Tortue.

Edition de procédures à l'aide de EDITE

EDITE
EDITE *nom(liste)* (ED) (commande)

La commande EDITE affiche l'écran d'édition Logo. Si vous faites suivre la commande EDITE du nom d'une procédure déjà définie, l'éditeur démarre avec la (ou les) définition(s) de la (ou des) procédure(s) *nom(liste)* contenue(s) dans la mémoire tampon de l'éditeur. La donnée de EDITE peut être un nom de procédure ou une liste de noms. Dans ce dernier cas, l'écran d'édition affiche la définition de toutes les procédures énumérées dans la liste.

Si une procédure *nom* n'est pas définie, la mémoire tampon de l'éditeur contient uniquement la ligne titre : POUR *nom*. Si aucune donnée n'est fournie à EDITE, la mémoire tampon a le même contenu qu'à la dernière utilisation de l'éditeur ; si c'est la première fois que vous faites appel à l'éditeur, la mémoire tampon est vide.

Ó - A est le moyen usuel pour quitter l'éditeur. Logo lit chaque ligne de la mémoire tampon d'édition comme si vous l'aviez tapée au niveau supérieur. Si Logo atteint la fin de la mémoire tampon et que le mot spécial FIN n'a pas été tapé, ce dernier est ajouté automatiquement.

Ó - ESC abandonne l'édition. Vous pouvez l'utiliser si ce que vous avez modifié ne vous satisfait pas ou si vous décidez de ne rien changer. Toute modification déjà apportée dans le tampon de l'éditeur sera ignorée.

Méthodes d'édition

Cette section présente les différentes touches ou combinaisons de touches que vous pouvez utiliser lorsque vous tapez une définition de procédure dans l'éditeur. Celles qui sont marquées d'un astérisque peuvent être utilisées aussi bien dans l'éditeur qu'au niveau supérieur.

Note : Rappelez-vous qu'en appuyant sur les touches [ó] - [?] lorsque vous êtes dans l'éditeur, Logo affiche un écran d'aide où les touches d'édition sont expliquées.

Déplacer le curseur

Les touches fléchées déplacent le curseur dans la direction vers laquelle elles pointent.

* [←] Déplace le curseur d'une position vers la gauche.

* [→] Déplace le curseur d'une position vers la droite.

[↓] Déplace le curseur à la ligne suivante. Le curseur tente de se rendre sous le caractère placé directement en dessous de celui sous lequel il se trouve. Si la ligne suivante est plus courte, le curseur se rend à la fin de cette dernière. Si le curseur est en fin de mémoire tampon, il ne bouge pas.

Exemples :

```
VOICI UNE LIGNE DE TEXTE                          curseur sous X
VOICI UNE AUTRE LIGNE DE TEXTE                    curseur sous un espace
UNE PLUS COURTE_                                  curseur à la fin
CETTE LIGNE EST SI LONGUE QU'ELLE N'ENT!
RE PAS DANS LA LARGEUR DE L'ECRAN                 curseur sous N
VOICI LA LIGNE SUIVANTE                           curseur sous T
```

↑	Déplace le curseur à la ligne précédente. Le curseur tente de se rendre sous le caractère placé directement au-dessus de celui sous lequel il se trouve. Si la ligne précédente est plus courte, le curseur se rend à la fin de cette dernière.
* Ó - ←	Déplace le curseur d'un mot vers la gauche.
* Ó - →	Déplace le curseur d'un mot vers la droite.
* Ó - <	Place le curseur au début de la ligne où il se trouve.
* Ó - >	Place le curseur à la fin de la ligne où il se trouve.
Ó - ↑	Place le curseur au début de la page affichée à l'écran. S'il s'y trouve déjà, Ó - ↑ place le curseur au début de la page précédente qui est alors affichée à l'écran.
Ó - ↓	Place le curseur à la fin de la page affichée à l'écran. S'il s'y trouve déjà, Ó - ↓ place le curseur au début de la page suivante qui est alors affichée à l'écran.
Ó - 1 à Ó - 9	Ó - 1 place le curseur au début de la mémoire tampon et Ó - 9 à la fin. Ó - 2 à Ó - 8 placent le curseur en différents endroits dans la mémoire tampon.

Insérer et effacer du texte

Cette section présente les touches ou combinaisons de touches qui permettent d'insérer ou d'effacer du texte.

↵	Accepte la ligne telle qu'elle apparaît et déplace le curseur, ainsi que le texte qui le suit, au début d'une nouvelle ligne.
* CONTROL - D * ou DELETE	Efface le caractère placé à la gauche du curseur.
* CONTROL - F	Efface le caractère situé à la position du curseur.

* CONTROL - X — Efface tous les caractères d'une ligne. Celle-ci peut contenir un maximum de 125 caractères. Ce texte est placé dans le tampon réserve.

* CONTROL - Y — Efface le reste de la ligne qui se trouve à la droite du curseur. Le texte est placé dans le tampon réserve.

CONTROL - R — Lorsque vous êtes dans l'éditeur, insère à la position du curseur une copie du texte qui se trouve dans le tampon réserve. Employée hors de l'éditeur, donne une copie de la dernière ligne tapée.

CONTROL - O — Ouvre une ligne à la position du curseur.

Quitter l'éditeur

Cette section présente les combinaisons de touches qui vous permettent de quitter l'éditeur.

Ó - A — Accepte les instructions et signale à Logo de lire le contenu de la mémoire tampon comme si vous l'aviez tapé au niveau supérieur.

Ó - ESC — Abandonne l'édition. Tous les changements que vous apportez sont placés dans la mémoire d'édition. Utilisez Ó - ESC si ce que vous avez modifié ne vous satisfait pas ou si vous décidez de ne rien changer. Si vous étiez en cours de définition, la procédure restera la même qu'avant la session d'édition. Si vous tapez Ó - ESC par mégarde, vous pouvez retourner à l'éditeur en utilisant EDITE sans avoir à fournir de données.

Autres primitives d'édition

Outre EDITE, il existe trois autres primitives qui vous permettent d'entrer en mode d'édition : EDN, EDNS et EDFICHIER.

EDN et EDNS permettent d'éditer les variables. EDN ramène l'éditeur Logo et place la variable indiquée et sa valeur dans la mémoire tampon d'édition. EDNS ramène l'éditeur Logo ainsi que toutes les variables et leurs valeurs. EDFICHIER ramène l'éditeur et place le contenu du fichier indiqué dans la mémoire tampon d'édition. Le fichier sera sauvegardé sous le nom utilisé pour l'édition.

EDN et EDNS sont décrites au chapitre 8 intitulé "Variables".

Pour obtenir plus de détails au sujet de EDFICHIER, consultez le chapitre 15 intitulé "Gestion des fichiers".

Graphique Tortue

Graphique Tortue

Référez-vous au chapitre 6 pour plus de détails au sujet de ECRANG, ECRAND et ECRANT.

Apple Logo dispose de trois types d'écran : l'écran graphique, l'écran divisé et l'écran texte. Chaque fois que vous utilisez une primitive qui fait intervenir la Tortue, Logo fait apparaître l'écran divisé. Les commandes ECRANG, ECRAND et ECRANT vous permettent de passer d'un type d'écran à un autre.

Au cours de ce chapitre, vous serez introduit aux commandes qui vous permettent de travailler à l'écran graphique ainsi qu'à un certain nombre d'opérations qui renseignent sur l'état de la Tortue et de son crayon, et sur le mode d'écran. Les primitives sont regroupées comme suit :

- celles qui changent l'état de la Tortue ;
- celles qui renseignent sur l'état de la Tortue ;
- celles qui modifient l'état du crayon ou le mode d'écran ;
- celles qui renseignent sur l'état du crayon et le mode d'écran.

La plupart de ces primitives ont été présentées dans le manuel *Introduction à la programmation*. Il est préférable d'avoir lu le manuel d'introduction avant de poursuivre la lecture de ce chapitre.

Commandes relatives à l'état de la Tortue

Cette section décrit, dans l'ordre suivant, les commandes qui font intervenir la Tortue.

AVANCE	FY
CACHETORTUE	GAUCHE
DROITE	MONTRETORTUE
FCAP	ORIGINE
FPOS	RECULE
FX	VE

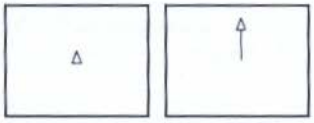
Les dimensions de l'écran sont de 240 pas de Tortue sur la hauteur et de 280 pas sur la largeur. Lorsque vous utilisez les coordonnées cartésiennes (dans FPOS, par exemple), 120 correspond à la limite supérieure de l'écran sur l'axe des y et −119 à la limite inférieure de l'écran. Sur l'axe des x, −140 correspond à la limite gauche et 139 à la limite droite. Ce sont les dimensions de l'écran lorsque le rapport d'échelle est .8. Remarquez que AVANCE et RECULE n'utilisent pas les coordonnées cartésiennes.

AVANCE

AVANCE *distance* (AV) (commande)

La commande AVANCE déplace la Tortue dans la direction vers laquelle elle pointe du nombre de pas indiqué par *distance*. Si le crayon est baissé, la Tortue trace une ligne correspondant à *distance*.

Exemples :



AVANCE 70

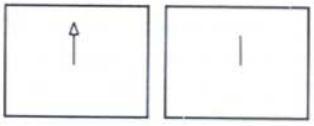
```
POUR CARRE :COTE
REPETE 4<AVANCE :COTE DROITE 90>
FIN
```

CARRE 30

CACHETORTUE

CACHETORTUE (CT) (commande)

Cette commande rend la Tortue invisible. Celle-ci dessine plus rapidement quand elle est cachée.



DROITE

DROITE *degrés* (DR) (commande)

Cette commande fait tourner la Tortue vers la droite (dans le sens des aiguilles d'une montre) du nombre de *degrés* indiqué. Il y a erreur si *degrés* est supérieur à 4.19E6.

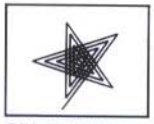
Exemples :

DROITE 45 fait tourner la Tortue de 45 degrés vers la droite.

DROITE −45 fait tourner la Tortue de 45 degrés vers la gauche.

DROITE 45 DROITE −45

```
POUR SPI :COTE :ANGLE :AUG
AV :COTE DR :ANGLE
SPI :COTE + :AUG :ANGLE :AUG
FIN
```



SPI 5 144 3

FCAP

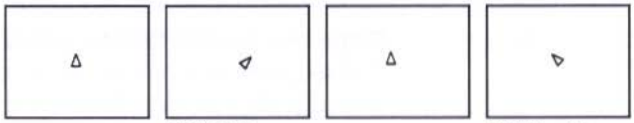
FCAP *degrés* (commande)

FCAP (fixe cap) fait tourner la Tortue de telle sorte que son cap ait la valeur indiquée par *degrés* (n'importe quel nombre décimal inférieur ou égal à 4.19E6). Les nombres positifs définissent une direction dans le sens des aiguilles d'une montre et les nombres négatifs dans le sens inverse à partir du nord. Remarquez que DROITE et GAUCHE définissent des mouvements relatifs tandis que FCAP définit un mouvement absolu, quel que soit le cap de la Tortue à ce moment.

Exemples :

FCAP 45 oriente la Tortue vers le nord-est.

FCAP −45 oriente la Tortue vers le nord-ouest.



FCAP 45 FCAP −45

Voir POS.

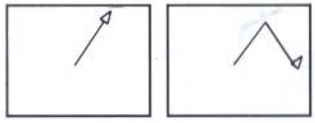
FPOS

FPOS <*coorx coory*> (commande)

La commande FPOS (fixe position) place la Tortue à la position indiquée par <*coorx coory*>. Si le crayon est baissé, la Tortue trace une ligne jusqu'à sa nouvelle position.

Exemple :

FPOS <100 0> place la Tortue au milieu du côté droit de l'écran.



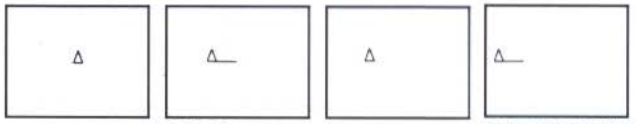
FX

FX *coorx* (commande)

FX (fixe x) déplace la Tortue horizontalement jusqu'au point d'abscisse x indiqué par *coorx*. L'ordonnée y reste inchangée. Si le crayon est baissé, la Tortue trace une ligne jusqu'à sa nouvelle position.

Exemple :

FX −50 déplace la Tortue horizontalement vers le côté gauche de l'écran. La limite gauche de l'écran est −140.



FX −50 FX 2 * COORX

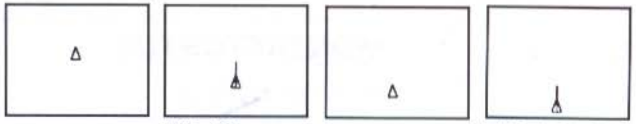
FY

FY *coory* (commande)

FY (fixe y) déplace la Tortue verticalement jusqu'au point d'ordonnée y indiqué par *coory*. L'abscisse x ne change pas. Si le crayon est baissé, la Tortue trace une ligne jusqu'à sa nouvelle position.

Exemple :

FY −50 déplace la Tortue vers le bas de l'écran. La limite inférieure de l'écran est −119 lorsque le rapport d'échelle est fixé à .8.



FY −50 FY 2 * COORY

GAUCHE

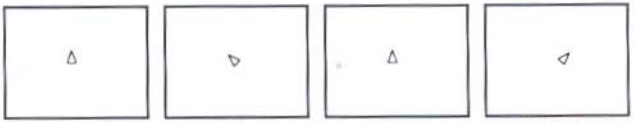
GAUCHE *degrés* (GA) (commande)

Cette commande fait tourner la Tortue vers la gauche (dans le sens inverse des aiguilles d'une montre) du nombre de *degrés* indiqué. Il y a erreur si *degrés* est supérieur à 4.19E6.

Exemples :

GAUCHE 45 fait tourner la Tortue de 45 degrés vers la gauche.

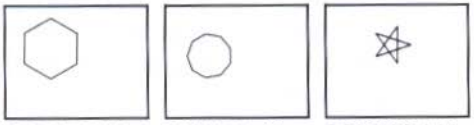
GAUCHE −45 fait tourner la Tortue de 45 degrés vers la droite.



GAUCHE 45 GAUCHE −45

La procédure POLY dessine des figures telles qu'illustrées ci-dessous :

```
POUR POLY :COTE :ANGLE
AVANCE :COTE
GAUCHE :ANGLE
POLY :COTE :ANGLE
FIN
```



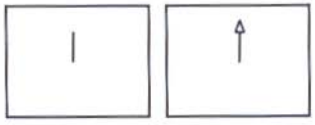
POLY 70 60 POLY 30 40 POLY 80 144

Voir CACHETORTUE.

MONTRETORTUE

MONTRETORTUE (MT) (commande)

Cette commande rend la Tortue visible.

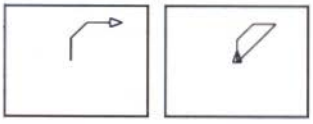


ORIGINE

ORIGINE (commande)

La commande ORIGINE amène la Tortue au centre de l'écran et fixe le cap à 0. Si le crayon est baissé, la Tortue trace une ligne jusqu'à l'origine. Cette instruction est équivalente à

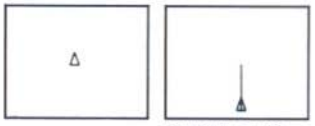
```
FPOS <0 0>
FCAP 0
```



RECULE

RECULE *distance* (RE) (commande)

La commande RECULE fait reculer la Tortue du nombre de pas indiqué par *distance*. Si le crayon est baissé, la Tortue trace une ligne correspondant à *distance*.

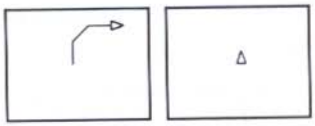


RECULE 70

VE

VE (commande)

VE (vide écran) efface l'écran graphique, remet la Tortue à la position <0 0> (cette position est appelée position origine) et fixe le cap de la Tortue à 0 (nord).



Opérations relatives à l'état de la Tortue

Cette section décrit, dans l'ordre suivant, les opérations qui renseignent sur l'état de la Tortue.

CAP	POSITION
COORX	VERS
COORY	VISIBLEP

CAP

CAP (opération)

Cette opération retourne le cap de la Tortue qui est un nombre décimal compris entre 0 et 359. Logo compte les degrés comme sur la boussole où 0 est le nord, 90 l'est, 180 le sud et 270 l'ouest. A la mise en route de Logo, la Tortue est à son point d'origine, c'est-à-dire au centre de l'écran, pointant vers le haut, cap 0.

Exemple :

```
FCAP 180
SI CAP = 180 <EC <VOUS ALLEZ DROIT AU !
SUD>>
```

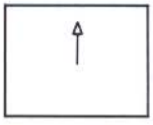
COORX

COORX (opération)

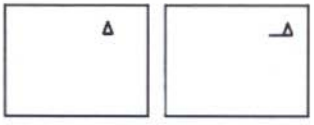
Cette opération retourne l'abscisse de la position actuelle de la Tortue.

Exemple :

```
?FX 45.3
?EC COORX
45.3
```



FX 2 * COORX déplace la Tortue horizontalement jusqu'au point dont l'abscisse est le double de la position précédente.



COORY

COORY (opération)

Cette opération retourne l'ordonnée de la position actuelle de la Tortue.

Exemple :

```
FY 50.0
?EC COORY
50.0
```

FY 2 * COORY déplace la Tortue jusqu'à un point dont l'ordonnée est le double de la position précédente.

POSITION

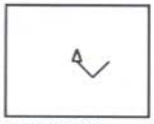
POSITION (POS) (opération)

L'opération POS retourne les coordonnées de la position actuelle de la Tortue sous la forme d'une liste <coorx coory>. A la mise en route de Logo, la Tortue est au centre de l'écran ; POS retourne <0 0>.

Exemples :

```
POUR LETTREV
RELIE "SAUVEPOS POS
V
LC
FPOS :SAUVEPOS
BC
FIN

POUR V
DR 135 AV 20
GA 90 AV 20
GA 45
FIN
```



LETTREV

LETTREV appelle la procédure V et replace la Tortue à la position où elle se trouvait avant l'appel de LETTREV.

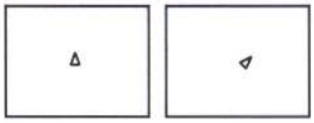
VERS

VERS <*coorx coory*> (opération)

Cette opération retourne le cap que la Tortue doit adopter pour pointer vers la position désignée par <*coorx coory*>.

Exemple :

FCAP VERS <20 10> oriente la Tortue dans la direction indiquée par la position <20 10>.



VISIBLEP

VISIBLEP (opération)

Cette opération retourne VRAI si la Tortue est visible, FAUX si elle ne l'est pas.

Commandes relatives au crayon et à l'écran

Cette section décrit, dans l'ordre suivant, les commandes qui ont trait au crayon et à l'écran.

BARRIERE
BC
ENROULE
FCC
FENETRE
FFOND
GC
IC
LC
NETTOIE
PEINS
POINT

BARRIERE

BARRIERE (commande)

Voir aussi FENETRE et ENROULE.

La commande BARRIERE enferme la Tortue dans les limites de l'écran. Toute tentative de franchir ces limites provoque l'apparition d'un message et laisse la Tortue immobile. Si la Tortue est déjà hors de l'écran, Logo replace celle-ci à sa position d'origine <0 0>.

Exemple :

```
BARRIERE
VE
DR 5
AV 500
```

provoque l'apparition du message :

```
TORTUE HORS LIMITES
```

BC

BC (commande)

BC (baisse crayon) place le crayon de la Tortue en position d'écriture. Alors que le crayon se trouve dans cet état, la Tortue trace des lignes de la couleur actuelle du crayon. A la mise en route de Logo, le crayon de la Tortue est baissé.

BC AV 100

ENROULE

ENROULE (commande)

Voir aussi BARRIERE et FENETRE.

Cette commande permet à la Tortue de franchir les limites de l'écran, c'est-à-dire que si la Tortue dépasse un côté de l'écran, elle réapparaît immédiatement sur le côté opposé. Elle n'échappe jamais aux limites visibles de l'écran.

Exemple :

```
?ENROULE
?VE DR 5
?AV 500
?EC POS
43.5779 18.0974
```

FCC

FCC *nocouleur* (commande)

FCC (fixe couleur crayon) fixe la couleur du crayon à la couleur définie par le *nocouleur*. Le *nocouleur* est l'un des nombres suivants :

0	noir
1	blanc
2	vert
3	violet
4	orange
5	bleu

Pour plus de détails sur la relation entre les couleurs du crayon et du fond, voir FOND.

Si la couleur du crayon n'est pas exactement celle que vous désirez, essayez d'abord d'ajuster le contraste de votre téléviseur. Quand deux lignes de couleurs différentes sont à proximité l'une de l'autre dans le sens horizontal, elles s'influencent réciproquement et l'une d'elles ne semblera pas correspondre au numéro de couleur indiqué. Vous n'y pouvez rien.

FENETRE

FENETRE (commande)

Voir aussi BARRIERE et ENROULE.

FENETRE rend le champ de la Tortue illimité. A l'écran, vous ne voyez qu'une partie du champ de la Tortue comme si vous regardiez par un guichet pratiqué au centre d'un grand écran dont le reste est masqué. La Tortue disparaît lorsqu'elle se meut hors de ce champ partiel qu'est l'écran de votre téléviseur, mais elle continue sa course invisible dans le champ entier. Lorsque le rapport d'échelle est .8, l'écran visible mesure en hauteur 240 pas de Tortue. En largeur, il mesure 280 pas. Le champ entier mesure 40960 pas en hauteur et 32768 en largeur (ceci peut varier selon le rapport d'échelle). Lorsque la Tortue est en dehors des limites de l'écran, le passage de FENETRE à BARRIERE ou ENROULE ramène celle-ci à sa position d'origine <0 0>.

Exemple :

```
?FENETRE
?VE DR 5
?AV 500
?EC POS
43.5779 498.097
```

FFOND

FFOND *nocouleur* (commande)

FFOND (fixe fond) fixe la couleur du fond au *nocouleur* indiqué. Le *nocouleur* peut être :

0	noir
1	blanc
2	vert
3	violet
4	orange
5	bleu
6	noir (pour moniteur monochrome)

Voir les exemples donnés à FOND.

Remarquez que 0 et 6 représentent tous deux du noir. FFOND 6 est recommandé si vous disposez d'un moniteur monochrome ; cela donne des lignes plus fines.

Il existe des limitations inévitables lorsque vous dessinez en couleur sur un fond de couleur. Le crayon noir ou blanc ne présente aucun problème quelle que soit la couleur du fond. Il en va de même lorsque le crayon dessine en couleur sur fond noir ou blanc. Si vous essayez un tracé vert ou violet sur un fond orange ou bleu, ou inversement, voici ce qui se passe :

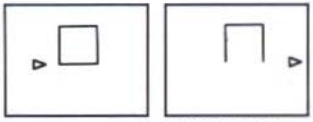
fond orange ou bleu	vert devient orange violet devient bleu
fond vert ou violet	orange devient vert bleu devient violet

Si vous changez la couleur du fond après avoir tracé des lignes de couleur, ces dernières risquent d'être altérées.

GC

GC (commande)

En utilisant GC (gomme crayon), le crayon de la Tortue se transforme en gomme à effacer. La Tortue efface toute ligne sur laquelle elle passe. Pour changer cet état, commandez BC ou LC.

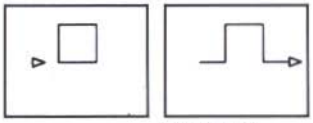


GC AV 100

IC

IC (commande)

En utilisant IC (inverse crayon), les couleurs du crayon sont inversées. IC trace des lignes là où il n'y en a pas et les efface là où elles existent. Les résultats de cette inversion sont complexes ; ils dépendent de la couleur du fond, de celle du crayon, du caractère horizontal ou vertical des lignes. On obtient les meilleurs résultats sur un fond noir.

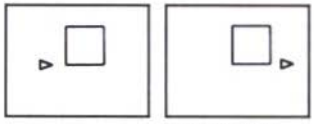


IC AV 100

LC

LC (commande)

LC (lève crayon) remonte le crayon de la Tortue qui ne laisse ainsi plus de trace en se déplaçant, à moins que vous ne redonniez la commande BC.

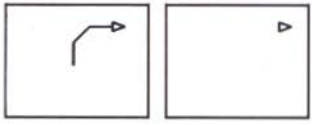


LC DR 100

NETTOIE

NETTOIE (commande)

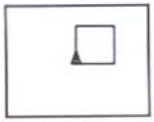
Cette commande efface l'écran graphique sans affecter la Tortue.



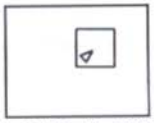
PEINS

PEINS (commande)

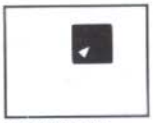
La commande PEINS remplit, de la couleur actuelle du crayon, une forme dessinée par la Tortue. Si la Tortue ne se trouve pas à l'intérieur de cette forme, c'est le fond qui se colore. En utilisant la commande PEINS, Logo ne tient pas compte des traits qui ne sont pas de la couleur actuelle du crayon.



REPETE 4
<AV 50 DR 90>



LC DR 45 AV 20



BC PEINS

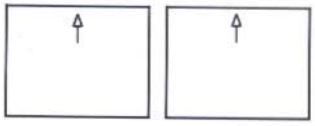
POINT

POINT <*coorx coory*> (commande)

Cette commande dessine un point de la couleur actuelle du crayon à la position désignée par les coordonnées, sans bouger la Tortue ni tracer de ligne, même si le crayon est baissé.

Exemple :

POINT <120 0> place un point au milieu du côté droit de l'écran.



Opérations relatives au crayon et à l'écran

Cette section décrit, dans l'ordre suivant, les opérations qui renseignent sur l'état du crayon et le mode d'écran.

CC
CRAYON
FOND
POINTP

CC

CC (opération)

CC (couleur crayon) retourne un nombre correspondant au numéro de la couleur actuelle du crayon.

0	noir
1	blanc
2	vert
3	violet
4	orange
5	bleu

A la mise en route de Logo, CC est 1.

CRAYON

CRAYON (opération)

CRAYON retourne une liste donnant l'état actuel du crayon de la Tortue. Cet état peut être BC, GC, LC ou IC. A la mise en route de Logo, CRAYON retourne BC.

FOND

FOND (opération)

Cette opération retourne un nombre représentant la couleur du fond :

0 noir
1 blanc
2 vert
3 violet
4 orange
5 bleu
6 noir (pour moniteur monochrome)

A la mise en route de Logo, FOND est 0.

POINTP

POINTP <*coorx coory*> (opération)

L'opération POINTP retourne VRAI s'il y a un point à la position désignée par <*coorx coory*>, FAUX s'il n'y en a pas.

Commandes de texte et d'écran

Commandes de texte et d'écran

Votre ordinateur Apple dispose de 24 lignes de 40 ou 80 caractères chacune, selon la donnée fournie à FLARGEUR. L'écran entier peut être consacré au texte ou au graphique. Vous pouvez aussi disposer d'un écran comportant 20 lignes consacrées au graphique, 4 lignes restant disponibles pour le texte au bas de l'écran. A la mise en route de Logo, l'écran est entièrement voué au texte.

L'écran peut contenir 40 ou 80 caractères par ligne. La primitive FLARGEUR (fixe largeur) vous permet de passer du mode 40 caractères au mode 80 caractères, et inversement.

Note : Si vous disposez d'un Apple IIe, Logo sera en mode 40 caractères à la mise en route.

Si vous disposez d'un Apple IIc, Logo vérifiera, à la mise en route, si le bouton de sélection d'affichage est à la position 40 ou 80 caractères et utilisera l'un ou l'autre mode.

Il est possible de modifier le mode d'écran de deux façons :

- à l'aide des commandes Logo usuelles qui peuvent être tapées au niveau supérieur ou insérées dans les procédures : ECRANG (écran graphique), ECRANT (écran texte), ECRAND (écran divisé) et FLARGEUR ;
- à l'aide de touches de fonction ayant un effet quasi immédiat (même si une procédure est en cours d'exécution) ; ces touches sont CONTROL-L, CONTROL-S et CONTROL-T. Elles ne peuvent être insérées dans une procédure.

.ECHELLE et .FECHELLE sont décrites au chapitre 18.

Les primitives .ECHELLE et .FECHELLE (fixe échelle) sont toutes deux des commandes relatives à l'écran.

Primitives qui modifient ce qui apparaît à l'écran

Cette section décrit, dans l'ordre suivant, les commandes qui modifient ce qui apparaît à l'écran.

CURSEUR
ECRAND
ECRANG
ECRANT
FCURSEUR
FLARGEUR
LARGEUR
VT

CURSEUR

CURSEUR (opération)

Voir aussi FCURSEUR.

Cette opération retourne une liste de deux nombres : les numéros de la colonne et de la ligne indiquant la position du curseur. L'extrême gauche en haut de l'écran est <0 0> et l'extrême droite <39 0> lorsque l'écran est en mode 40 caractères, et <79 0> lorsque l'écran est en mode 80 caractères.

Exemple :

La procédure TAB passe à la position d'arrêt suivante après l'exécution d'un TAPE. Les positions d'arrêt sont à toutes les 8 colonnes.

```
POUR TAB
TAPE CAR 32
SI SUPP (RESTE PREMIER CURSEUR 8) ∅<TA!
B>
FIN
POUR TABLEAU
TAPE "NOM TAB TAB EC "NOTE EC<>
TAPE "MARTIN TAB TAB EC 18
TAPE "SIMON TAB TAB EC 17.5
TAPE "GENEVOIS TAB EC 19.5
FIN

?TABLEAU
NOM                    NOTE

MARTIN                 18
SIMON                  17.5
GENEVOIS               19.5
```

ECRAND

ECRAND (commande)

La commande ECRAND (écran divisé) partage l'écran en deux parties. Les vingt premières lignes sont consacrées au graphique et les quatre dernières au texte.

ECRANG

ECRANG (commande)

ECRANG (écran graphique) consacre tout l'écran au graphique ; seul le champ de la Tortue apparaît à l'écran. Tout ce que vous tapez reste invisible ; toutefois, Logo tient compte des instructions que vous lui donnez.

Si Logo a besoin de vous renvoyer un message, il passera de lui-même en mode ECRAND.

ECRANT

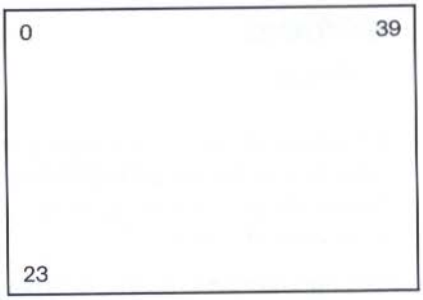
ECRANT (commande)

ECRANT (écran texte) réserve l'écran entier au texte ; le champ de la Tortue restera invisible jusqu'à l'exécution d'une procédure graphique.

FCURSEUR

FCURSEUR *nocolonne noligne* (commande)

La commande FCURSEUR (fixe curseur) place le curseur à la position indiquée par *nocolonne* et *noligne*. Les lignes sont numérotées de 0 à 23 de haut en bas. Les colonnes (position du caractère sur une ligne) sont numérotées de 0 à 39 en mode 40 caractères et de 0 à 79 en mode 80 caractères.



Il y a erreur si le numéro de ligne n'est pas compris entre 0 et 23, ou le numéro de colonne entre 0 et 38 (ou 0 et 78 en mode 80 caractères). Si *nocolonne* ou *noligne* est un nombre décimal, Logo le transforme en nombre entier.

Exemple :

`FCURSEUR <20 12>` place le curseur près du centre de l'écran.

```
POUR MVTCURS :X :Y
FCURSEUR LISTE (:X + PREMIER CURSEUR) (!
:Y + DERNIER CURSEUR)
FIN

?VT
?EC "A MVTCURS 2 5 EC "B
```

FLARGEUR

FLARGEUR *largeur* (commande)

Voir aussi LARGEUR.

La commande FLARGEUR donne à l'écran la *largeur* d'affichage indiquée, soit 40, soit 80. A la mise en route de Logo, la largeur d'affichage dépend du type d'ordinateur que vous utilisez : si vous disposez d'un Apple IIe, la largeur d'affichage est 40. Si vous disposez d'un Apple IIc, celle-ci dépend de la position du bouton de sélection d'affichage.

Exemple :

FLARGEUR 80 place l'écran en mode 80 caractères.

LARGEUR

LARGEUR (opération)

Voir aussi FLARGEUR pour changer la largeur d'affichage de l'écran.

Cette opération retourne la largeur d'affichage actuelle de l'écran, soit 40, soit 80. A la mise en route de Logo, LARGEUR retourne 40 si vous disposez d'un Apple IIe, ou 40 ou 80 selon la position qu'indique le bouton de sélection d'affichage, si vous disposez d'un Apple IIc.

VT

VT (commande)

La commande VT (vide texte) vide entièrement l'écran du texte et place le curseur en haut et à gauche de la partie texte de l'écran. Si vous avez utilisé l'écran divisé, le curseur se placera à la quatrième ligne avant le bas de l'écran.

Touches de fonction qui modifient l'écran

Cette section décrit, dans l'ordre suivant, les touches de fonction qui vous permettent de changer le mode d'écran.

CONTROL-L
CONTROL-S
CONTROL-T

CONTROL-L

CONTROL - L (touche spéciale)

CONTROL - L est équivalente à ECRANG. Elle peut être tapée n'importe quand.

Si vous appuyez sur CONTROL - L lorsque vous êtes dans l'éditeur Logo, l'écran graphique est affiché. Pressez sur CONTROL - T pour ramener l'éditeur de texte.

CONTROL-S

CONTROL - S (touche spéciale)

CONTROL - S est équivalente à ECRAND. Elle peut être tapée n'importe quand.

CONTROL-T

CONTROL - T (touche spéciale)

CONTROL - T est équivalente à ECRANT ; elle consacre tout l'écran au texte. Elle peut être tapée n'importe quand. CONTROL - T ramène l'éditeur de texte si vous avez utilisé CONTROL - L dans l'éditeur.

Mots et listes

Mots et listes

Dans le langage Logo, il y a deux types d'objets : les **mots** et les **listes**. Ce chapitre traite des primitives qui servent :

- à décomposer les mots et les listes ;
- à regrouper les mots et les listes ;
- à examiner les mots et les listes ;
- à inscrire en majuscules ou en minuscules les caractères formant les mots et les listes.

Quelques indications sur les mots

Un **mot** est composé de caractères. En voici quelques exemples :

BONJOUR
X
314
3.14
R2D2
COCHOND'INDE
COCHON.D'INDE
COCHON-D'INDE taper sous la forme COCHON!-D'INDE
HEN3RI
QUI? !COMMENT!

Chaque caractère est un **élément** du mot. Le mot HEN3RI contient 6 éléments.

H E N 3 R I

Un mot est généralement délimité par des espaces. Cela signifie qu'il y a un espace avant (à moins que le mot ne soit précédé des : ou des ") et un espace après, ce qui permet de distinguer le mot du reste de la ligne. Il existe certains autres caractères délimiteurs :

<> ( ) = + − *

Pour considérer chacun de ces caractères comme un caractère alphabétique normal, faites-le précéder d'un point d'exclamation "!".

Exemple :

```
?EC "COCHON!-D'INDE
COCHON-D'INDE
```

Remarquez que les " et les : ne sont pas des caractères délimiteurs.

Voir l'annexe E intitulée "Interprétation" pour plus de détails sur la façon dont Logo lit les caractères délimiteurs.

Un **mot vide** est un mot qui ne contient aucun élément. Vous le tapez au clavier de la façon suivante :

Voir la primitive VIDEP présentée dans ce chapitre pour des exemples de mot vide.

```
EC "
```

Quelques indications sur les listes

Une **liste** est composée d'objets Logo. Chacun d'eux peut être un mot ou une autre liste. Une liste s'inscrit entre crochets. En voici quelques exemples :

```
<BONJOUR, COMMENT ALLEZ!-VOUS?>
<X Y Z>
<SALUT>
<<MAISON HOUSE><FENETRE WINDOW><CHIE!
N DOG>>
<1 <1 2><17 <17 2>>>
<>
```

La liste <BONJOUR, COMMENT ALLEZ!-VOUS?> contient 3 éléments :

BONJOUR,
COMMENT
ALLEZ-VOUS?

Remarquez que la liste <1 <1 2><17 <17 2>>> ne contient pas six éléments, mais bien trois, le second et le troisième étant eux-mêmes des listes.

Premier élément : 1
Deuxième élément : <1 2>
Troisième élément : <17 <17 2>>

Voir la primitive VIDEP présentée dans ce chapitre pour des exemples de liste vide.

La liste <>, qui ne contient aucun élément, est la **liste vide**.

Décomposer les mots et les listes

Les opérations utilisées pour décomposer les mots et les listes sont les suivantes :

DERNIER, DE
ELEM
MEMBRE
PREMIER, PR
SAUFDERNIER, SD
SAUFPREMIER, SP

Le tableau suivant démontre le fonctionnement des primitives PREMIER, PR et SAUFPREMIER, SP. Si vous voulez voir le résultat de ces opérations, utilisez la commande MONTRE.

PREMIER	"JEAN	J
SP	"JEAN	EAN
PREMIER	<MARIE JEAN LUC>	MARIE
SP	<MARIE JEAN LUC>	<JEAN LUC>
PREMIER	<<MARIE JEAN> LUC>	<MARIE JEAN>
SP	<<MARIE JEAN> LUC>	<LUC>
PREMIER	<MARIE <JEAN LUC>>	MARIE
SP	<MARIE <JEAN LUC>>	<<JEAN LUC>>
PREMIER	<> ou "	erreur
SP	<> ou "	erreur

DERNIER, DE et SAUFDERNIER, SD fonctionnent de la même façon sauf qu'elles utilisent le dernier élément des mots et des listes.

DERNIER

DERNIER *obj* (DE) (opération)

Cette opération retourne le dernier élément d'*obj*. DERNIER appliquée au mot vide ou à la liste vide est une erreur.

Exemples :

Opération		**Résultat**
DERNIER	<MARIE SEBASTIEN ONESIME>	ONESIME
DERNIER	"CANELLE	E
DERNIER	<CANELLE>	CANELLE
DERNIER	<<LE> SOLDAT VA <PARTIR SAUTER DORMIR>>	<PARTIR SAUTER DORMIR>
DERNIER	"	erreur
DERNIER	<>	erreur

```
POUR INVEPELLE :DONNEE
SI VIDEP :DONNEE <STOP>
EC DERNIER :DONNEE
INVEPELLE SD :DONNEE
FIN
```

```
?INVEPELLE "CHOCOLAT
T
A
L
O
C
O
H
C
```

ELEM

ELEM *entier obj* (opération)

ELEM retourne l'élément d'*obj* dont la position correspond à *entier*. Par exemple, si *entier* est 3, ELEM retourne le troisième élément de *obj*, ce dernier pouvant être un mot ou une liste. Il y a erreur si *entier* est supérieur au nombre d'éléments que contient *obj* ou si *obj* est la liste vide.

Exemples :

```
?RELIE "ANIMAUX <CHIEN CHAT GIRAFE>
?EC ELEM 3 :ANIMAUX
GIRAFE
?EC ELEM 1 :ANIMAUX
CHIEN
```

MEMBRE

MEMBRE *obj1 obj2* (opération)

Cette opération retourne la partie de *obj2* dans laquelle *obj1* est le premier élément. Si *obj1* n'est pas un élément de *obj2*, MEMBRE retourne la liste vide ou le mot vide. Cette opération est utile pour avoir accès à l'information contenue dans un fichier ou pour trier de longues listes.

Exemples :

```
?MONTRE MEMBRE "A <A B C>
<A B C>

?MONTRE MEMBRE "Tortue <Graphique Tortu!
e Logo>
<Tortue Logo>

?MONTRE MEMBRE <PIAGET PAPERT><ENFANT !
ORDINATEUR <GEOMETRIE MATHEMATIQUES><P!
IAGET PAPERT>>
<<PIAGET PAPERT>>

?EC MEMBRE "ABC "XYZABCDEF
ABCDEF
```

PREMIER

PREMIER *obj* (PR) (opération)

Cette opération retourne le premier élément d'*obj*. PREMIER appliquée à la liste vide ou au mot vide est une erreur. PREMIER d'un mot est un caractère. PREMIER d'une liste peut être un mot ou une liste.

Exemples :

Opération		**Résultat**
PREMIER	<CAILLOU HIBOU JOUJOU>	CAILLOU
PREMIER	"PALAIS	P
PREMIER	<JOUJOU>	JOUJOU
PREMIER	<<LES TROIS> <CHAT CHIEN RAT> <GRIFFE MORD RONGE>>	<LES TROIS>
PREMIER	"	erreur
PREMIER	<>	erreur

```
POUR EPELLE :DONNEE
SI VIDEP :DONNEE <STOP>
EC PREMIER :DONNEE
EPELLE SP :DONNEE
FIN

?EPELLE "SOURIS
S
O
U
R
I
S

?EPELLE <A VOS SOUHAITS>
A
VOS
SOUHAITS
```

SAUFDERNIER

SAUFDERNIER *obj* (SD) (opération)

Cette opération retourne la liste ou le mot *obj* sauf son dernier élément.

Exemples :

Opération	**Résultat**
SAUFDERNIER <ALBERT EINSTEIN>	<ALBERT>
SAUFDERNIER "HEURES	HEURE
SAUFDERNIER <HEURES>	<>
SAUFDERNIER <<LES TROIS> <OISEAU ABEILLE FLEUR>>	<<LES TROIS>>
SAUFDERNIER "	erreur
SAUFDERNIER <>	erreur

La donnée de la procédure suivante doit être un verbe en "er" :

```
POUR CONJUGUE :VERBE
EC PH<JE>SD :VERBE
EC PH<NOUS>MOT SD SD :VERBE "ONS
FIN

?CONJUGUE "CHANTER
JE CHANTE
NOUS CHANTONS
```

SAUFPREMIER

SAUFPREMIER *obj* (SP) (opération)

Cette opération retourne la liste ou le mot *obj* sauf son premier élément. SAUFPREMIER d'une liste ou d'un mot vide est une erreur.

Exemples :

Opération	**Résultat**
SAUFPREMIER <ALBERT EINSTEIN>	<EINSTEIN>
SAUFPREMIER "TROIS	ROIS
SAUFPREMIER <TROIS>	<>
SAUFPREMIER <<LES DEUX><CHIEN CHAT><JAPPE MIAULE>>	<<CHIEN CHAT><JAPPE MIAULE>>
SAUFPREMIER "	erreur
SAUFPREMIER <>	erreur

```
POUR TRIANGLE :OBJET
SI VIDEP :OBJET <STOP>
EC :OBJET
TRIANGLE SP :OBJET
FIN
```

```
?TRIANGLE "TROIS
TROIS
ROIS
OIS
IS
S
```

```
?TRIANGLE <A PAS DE TORTUE>
A PAS DE TORTUE
PAS DE TORTUE
DE TORTUE
TORTUE
```

Regrouper les mots et les listes

Les opérations utilisées pour regrouper les mots et les listes sont les suivantes :

CONVERTIS
LISTE
METSDERNIER, MD
METSPREMIER, MP
MOT
PHRASE, PH

Le tableau suivant compare les primitives LISTE, METSDERNIER, METSPREMIER, MOT et PHRASE.

Opération	Donnée 1	Donnée 2	Résultat
LISTE	"RADIO	"ACTIF	<RADIO ACTIF>
MD	"RADIO	"ACTIF	erreur
MP	"RADIO	"ACTIF	erreur
MOT	"RADIO	"ACTIF	RADIOACTIF
PH	"RADIO	"ACTIF	<RADIO ACTIF>
LISTE	"LOGO	<EST MERVEILLEUX>	<LOGO <EST MERVEILLEUX>>
MD	"LOGO	<EST MERVEILLEUX>	<EST MERVEILLEUX LOGO>
MP	"LOGO	<EST MERVEILLEUX>	<LOGO EST MERVEILLEUX>
MOT	"LOGO	<EST MERVEILLEUX>	erreur
PH	"LOGO	<EST MERVEILLEUX>	<LOGO EST MERVEILLEUX>
LISTE	"ORDINATEURS	<>	<ORDINATEURS<>>
MD	"ORDINATEURS	<>	<ORDINATEURS>
MP	"ORDINATEURS	<>	<ORDINATEURS>
MOT	"ORDINATEURS	<>	erreur
PH	"ORDINATEURS	<>	<ORDINATEURS>

CONVERTIS

CONVERTIS *mot* (opération)

Cette opération retourne une liste obtenue en mettant *mot* en liste. CONVERTIS est très utile pour mettre en liste ce que retourne LISMOT.

Exemples :

```
?MONTRE CONVERTIS "mot
<mot>
?RELIE "Donnée LISMOT
chiens chats hamsters
?MONTRE :Donnée
chiens chats hamsters
?MONTRE CONVERTIS :Donnée
<chiens chats hamsters>
```

LISTE

LISTE *obj1 obj2* (opération)
(LISTE *obj1 obj2 obj3 obj4...)*

Cette opération retourne une liste dont les éléments sont *obj1*, *obj2*, etc.

Exemples :

Opération	**Résultat**
LISTE "ROSE <TULIPE ŒILLET>	<ROSE <TULIPE ŒILLET>>
(LISTE "ROSE "TULIPE "ŒILLET)	<ROSE TULIPE ŒILLET>
LISTE <LA RAISON DU PLUS FORT> <EST TOUJOURS LA MEILLEURE>	<<LA RAISON DU PLUS FORT> <EST TOUJOURS LA MEILLEURE>>
LISTE "A <>	<A <>>

Si LISTE est utilisée avec une seule donnée, vous devez employer les parenthèses.

```
?RELIE "ANIMAUX "CRAPAUDS
?MONTRE (LISTE :ANIMAUX)
<CRAPAUDS>
```

METSDERNIER

METSDERNIER *obj liste* (MD) (opération)

Cette opération retourne une nouvelle liste qui ajoute *obj* comme dernier élément de *liste.*

Exemples :

Opération	**Résultat**
MD "SOURIS <HAMSTER COCHOND'INDE>	<HAMSTER COCHOND'INDE SOURIS>
MD <LE LA LES> <CHAT GIRAFE>	<CHAT GIRAFE <LE LA LES>>
MD "A <>	<A>
DERNIER MD "SOURIS <HAMSTER COCHOND'INDE>	SOURIS

La procédure suivante ajoute un mot à un dictionnaire français-espagnol :

```
POUR NOUVEAUMOT :DONNEE
RELIE "DICTIONNAIRE MD :DONNEE :DICTION!
NAIRE
FIN

?RELIE "DICTIONNAIRE <<MAISON CASA><ES!
PAGNOL ESPANOL><COMMENT COMO>>
?MONTRE :DICTIONNAIRE
<<MAISON CASA><ESPAGNOL ESPANOL><COMM!
ENT COMO>>

?NOUVEAUMOT <TABLE MESA>

?MONTRE :DICTIONNAIRE
<<MAISON CASA><ESPAGNOL ESPANOL><COMM!
ENT COMO><TABLE MESA>>
```

METSPREMIER

METSPREMIER *obj liste* (MP) (opération)

Cette opération retourne une nouvelle liste obtenue en ajoutant *obj* en première place de *liste*.

Exemples :

Opération	**Résultat**
MP "SOURIS <CHIEN CHAT>	<SOURIS CHIEN CHAT>
MP <LE GRAND><BOL EN VERRE>	<<LE GRAND> BOL EN VERRE>
MP "A <>	<A>

MOT

MOT *mot1 mot2* (opération)
(MOT *mot1 mot2 mot3...)*

Cette opération retourne un mot composé de ses données.

Exemples :

Opération	**Résultat**
MOT "COU "LEUR	COULEUR
(MOT "APO "CALY "PSE)	APOCALYPSE
MOT "COU <LEUR>	erreur
MOT "V "IVRE	VIVRE

La procédure SUFFIXE ajoute MUM à la fin de sa donnée.

```
POUR SUFFIXE :MOT
RETOURNE MOT :MOT "MUM
FIN

?EC SUFFIXE "MAXI
MAXIMUM
```

Le principe de la procédure SUFFIXE appliqué aux procédures LATIN et CUISINE permet de traduire en "latin de cuisine" la phrase proposée.

```
POUR LATIN :PH
SI VIDEP :PH<RT<>>
RT PH CUISINE PREMIER :PH LATIN SP :PH
FIN

POUR CUISINE :MOT
SI MEMBREP PREMIER :MOT<A E I O U><RT!
 MOT :MOT "MUM>
RT CUISINE MOT SP :MOT PREMIER :MOT
FIN

?EC LATIN <PERSONNE N'A JAMAIS PARLE LE!
 LATIN DE CUISINE PARMI LES HUMAINS>
ERSONNEPMUM AN' NUM AMAISJMUM ARLEPMUM E!
LMUM ATINLMUM EDMUM UISINECMUM ARMIPMUM!
 ESLMUM UMAINSHMUM
?_
```

PHRASE

PHRASE *obj1 obj2* (PH) (opération)
(PHRASE *obj1 obj2 obj3...)*

PHRASE retourne une liste composée de ses données.

Exemples :

Opération	**Résultat**
PH "PAPIER "CAHIER	<PAPIER CAHIER>
PH <PAPIER><CAHIER>	<PAPIER CAHIER>
PH "POMME <PECHE POIRE ABRICOT>	<POMME PECHE POIRE ABRICOT>
PH <LA RAISON DU PLUS FORT><EST TOUJOURS LA MEILLEURE>	<LA RAISON DU PLUS FORT EST TOUJOURS LA MEILLEURE>

La procédure suivante annonce une naissance :

```
POUR ANNONCE :PRENOM :NOM
EC <NOUS AVONS LA JOIE D'ANNONCER LA NA!
ISSANCE DE>
EC (PH :PRENOM "R. :NOM)
EC <4573 GRAMMES>
FIN

?ANNONCE "MARIE! -ANDREE "THEORET
NOUS AVONS LA JOIE D'ANNONCER LA NAISSA!
NCE DE
MARIE-ANDREE R. THEORET
4573 GRAMMES
```

Autres exemples :

Opération	Résultat
(PH "POMME "PECHE "POIRE)	<POMME PECHE POIRE>
(PH "MONET)	<MONET>
PH "MONET <>	<MONET>

L'emploi des parenthèses est nécessaire lorsque PHRASE utilise une seule donnée.

```
?RELIE "ANIMAUX "CHATONS
?MONTRE (PH :ANIMAUX)
<CHATONS>
```

Comparez ce que PHRASE et LISTE retournent quand vous les appliquez à des données qui sont des listes de listes.

Opération	Résultat
PH <LE CHIEN><AIME <LES CARAMELS>>	<LE CHIEN AIME <LES CARAMELS>>
LISTE <LE CHIEN><AIME <LES CARAMELS>>	<<LE CHIEN><AIME <LES CARAMELS>>>

Examiner les mots et les listes

Les opérations suivantes permettent d'examiner les mots et les listes.

ASCII	LISTEP
AVANTP	MEMBREP
CAR	MOTP
COMPTE	NOMBREP
EGALP	VIDEP

ASCII

ASCII *car* (opération)

Voir CAR. Voir aussi l'annexe F pour une liste complète du code ASCII.

Cette opération retourne le code ASCII de *car*. ASCII signifie *American Standard Code for Information Interchange*. Si *car* contient plus d'un caractère, ASCII n'opère que sur le premier.

Exemples :

ASCII "B retourne 66.

La procédure CODESECRET définit un nouveau mot en utilisant le code chiffré de César qui ajoute 3 à chaque lettre. Notez que cette procédure ne fonctionne pas si vous utilisez les minuscules.

```
POUR CODESECRET :MOT
SI VIDEP :MOT<RETOURNE "
RETOURNE MOT CODE PREMIER :MOT CODESECR!
ET SP :MOT
FIN

POUR CODE :LETTRE
RELIE "NUMERO (ASCII :LETTRE) + 3
SI SUPP :NUMERO ASCII "Z<RELIE "NUMERO!
 :NUMERO - 26>
RETOURNE CAR :NUMERO
FIN

?EC CODESECRET "CHAT
FKDW

?EC CODESECRET "CRAYON
FUDBRQ
```

AVANTP

AVANTP *mot1 mot2* (opération)

AVANTP retourne VRAI si *mot1* se trouve avant *mot2*. Logo fait la comparaison en utilisant le code ASCII de chacun des caractères des deux mots. Logo traite les majuscules avant les minuscules.

Exemples :

Opération	**Résultat**
AVANTP "A "a	VRAI
AVANTP "pomme "ZOO	FAUX
AVANTP MAJUSCULE "pomme MAJUSCULE "Zoo	VRAI

La procédure TRI retourne une liste de mots dans l'ordre alphabétique.

```
POUR TRI :MOT :LISTE
SI VIDEP :MOT <RT :LISTE>
RELIE "LISTE INSERE PREMIER :MOT :LISTE
RT TRI SP :MOT :LISTE
FIN

POUR INSERE :M :L
SI VIDEP :L <RT (LISTE :M)>
SI AVANTP :M PREMIER :L <RT MP :M :L>
RT MP PREMIER :L INSERE :M SP :L
FIN
```

Essayez ceci :

```
EC TRI <A D E F T C Z><>
A C D E F T Z
```

Puis tapez :

```
EC TRI <ZOO BAR BOA><A D E F C T Z>
A BAR BOA C D E F T Z ZOO
```

CAR

CAR *entier* (opération)

Voir l'annexe F pour une liste complète du code ASCII.

CAR retourne le caractère dont le code ASCII est *entier*. Il y a erreur si *entier* n'est pas un nombre compris entre 0 et 255.

Le caractère peut être normal (blanc sur fond noir) ou vidéo inverse (noir sur fond blanc). Le code ASCII est organisé de la manière suivante :

0 - 31	lettres majuscules
32 - 47	ponctuation
48 - 57	chiffres
58 - 63	ponctuation
64 - 90	lettres majuscules
91 - 96	ponctuation
97 - 122	lettres minuscules
123 - 127	ponctuation
128 - 154	lettres majuscules (vidéo inverse)
155 - 191	chiffres et ponctuation (vidéo inverse)
192 - 218	symboles Souris
219 - 255	lettres minuscules (vidéo inverse)

Pour transformer un caractère normal en caractère vidéo inverse, utilisez la procédure suivante :

```
POUR TRANSFORMER :CAR
SI SUPP (ASCII :CAR) 127<RT :CAR>
SI OU INFP (ASCII :CAR) 64 ET SUPP (ASC!
II :CAR) 96 INFP (ASCII :CAR) 128<RT C!
AR 128 + ASCII :CAR><RT CAR 64 + ASCII!
 :CAR>
FIN
```

La procédure INVERSER affiche un mot en vidéo inverse.

```
POUR INVERSER :MOT
SI VIDEP :MOT<RT "">
RT MOT TRANSFORMER PREMIER :MOT INVERSE!
R SP :MOT
FIN

?EC INVERSER "YOGOURT
YOGOURT
?
```

COMPTE

COMPTE *obj* (opération)

Cette opération retourne le nombre d'éléments contenus dans *obj*. Ce dernier peut être un mot ou une liste.

Exemples :

Opération	**Résultat**
COMPTE <REGARDE LE VOILIER>	3
COMPTE <REGARDE <LE VOILIER> BLANC>	2
COMPTE "ORDINATEUR	10

```
?RELIE "CLASSE <LORRAINE JULIEN RICHARD!
LISE ALAIN MARIE! -ANDREE NICOLE>
?EC COMPTE :CLASSE
7
```

La procédure suivante affiche un élément pris au hasard dans un mot ou une liste :

```
POUR PIQUEHASARD :DONNEE
EC ELEM (1 + HASARD COMPTE :DONNEE) :DO!
NNEE
FIN

?PIQUEHASARD :CLASSE
MARIE-ANDREE
```

EGALP

EGALP *obj1 obj2* (opération)

Voir la liste des formes infixées au chapitre 9.

Cette opération retourne VRAI si *obj1* et *obj2* sont des nombres égaux, des mots ou des listes identiques. Dans le cas contraire, EGALP retourne FAUX. Cette opération est équivalente au symbole =.

Exemples :

Opération	**Résultat**
EGALP "ROUGE PREMIER <ROUGE JAUNE>	VRAI
EGALP 100 50 * 2	VRAI
EGALP <LE CHAT GRIS> <LE CHAT>	FAUX
EGALP " <>	FAUX (Le mot vide et la liste vide ne sont pas identiques.)

La procédure suivante indique si la première donnée (un caractère) est un élément de la seconde (un mot) :

```
POUR INCLUS :CAR :MOT
SI VIDEP :MOT<RT "FAUX>
SI EGALP :CAR PREMIER :MOT<RT "VRAI>
RT INCLUS :CAR SP :MOT
FIN

?EC INCLUS "A "AUTOMOBILE
VRAI
?EC INCLUS "TOI "AUTOMOBILE
FAUX
```

LISTEP

LISTEP *obj* (opération)

Cette opération retourne VRAI si *obj* est une liste ; sinon, retourne FAUX.

Exemples :

Opération	**Résultat**
LISTEP 3	FAUX
LISTEP <3>	VRAI
LISTEP <>	VRAI
LISTEP "	FAUX
LISTEP <A B C <D E> <F <G>>>	VRAI
LISTEP SP "CHOCOLAT	FAUX
LISTEP SP <CHOCOLAT>	VRAI

MEMBREP

MEMBREP *obj1 obj2* (opération)

Cette opération retourne VRAI si *obj1* est un élément de *obj2*; sinon, retourne FAUX.

Exemples :

Opération	**Résultat**
MEMBREP 3 <2 5 <3> 6>	FAUX
MEMBREP 3 <2 5 3 6 >	VRAI
MEMBREP <2 5> <2 5 3 6>	FAUX
MEMBREP "PIN "LAPIN	VRAI
MEMBREP <QUEBEC ONTARIO> <<QUEBEC ONTARIO> MANITOBA>	VRAI
MEMBREP <QUEBEC ONTARIO> <QUEBEC ONTARIO MANITOBA>	FAUX
MEMBREP SP "RAT <AR AS AT AU>	VRAI

La procédure suivante détermine si la donnée est une voyelle :

```
POUR VOYELLEP :LETTRE
RT MEMBREP :LETTRE <A E I O U Y>
FIN

?EC VOYELLEP "F
FAUX
?EC VOYELLEP "A
VRAI
```

MOTP

MOTP *obj* (opération)

MOTP retourne VRAI si *obj* est un mot ; sinon, retourne FAUX.

Note : En Logo, les nombres sont des mots.

Exemples :

Opération	**Résultat**
MOTP "POUF	VRAI
MOTP <O GRE>	FAUX
MOTP 3	VRAI
MOTP <3>	FAUX
MOTP <>	FAUX
MOTP "	VRAI
MOTP SP "AUTO	VRAI
MOTP SP <AUTO>	FAUX

NOMBREP

NOMBREP *obj* (opération)

Cette opération retourne VRAI si *obj* est un nombre ; sinon, retourne FAUX.

Exemples :

Opération	**Résultat**
NOMBREP 3	VRAI
NOMBREP <3>	FAUX
NOMBREP 3.14E23	VRAI
NOMBREP <>	FAUX
NOMBREP "	FAUX
NOMBREP SP 3165.2	VRAI
NOMBREP SP <ELEPHANT>	FAUX

VIDEP

VIDEP *obj* (opération)

Cette opération retourne VRAI si *obj* est le mot vide ou la liste vide ; sinon, retourne FAUX.

Exemples :

Opération	**Résultat**
VIDEP 3	FAUX
VIDEP SP "CHAPEAU	FAUX
VIDEP SP "U	VRAI
VIDEP SP <CHAPEAU>	VRAI

La procédure CRIS associe aux animaux leur cri respectif :

```
POUR CRIS :ANIMAUX :BRUITS
SI OU VIDEP :BRUITS VIDEP :ANIMAUX <EC !
<VOILA, C'EST TOUT> STOP>
EC PH PREMIER :ANIMAUX PREMIER :BRUITS
CRIS SP :ANIMAUX SP :BRUITS
FIN
```

```
?CRIS <CHIENS OISEAUX COCHONS> <ABOIENT!
 PEPIENT GROGNENT>
CHIENS ABOIENT
OISEAUX PEPIENT
COCHONS GROGNENT
VOILA, C'EST TOUT
```

La procédure ENVERS affiche les éléments d'un mot ou d'une liste à l'envers :

```
POUR ENVERS :CHOSE
SI VIDEP :CHOSE <EC <> STOP>
TAPE DERNIER :CHOSE
SI LISTEP :CHOSE <TAPE CAR 32>
ENVERS SD :CHOSE
FIN
```

```
?ENVERS "ELEPHANT
TNAHPELE
?ENVERS "CITROUILLE
ELLIUORTIC
?ENVERS <ALICE AIME MATHIEU>
MATHIEU AIME ALICE
?ENVERS "OTTO
OTTO
```

Afficher en majuscules ou en minuscules

MAJUSCULE et MINUSCULE vous permettent d'utiliser les majuscules et les minuscules.

MAJUSCULE

MAJUSCULE *mot* (opération)

Cette opération retourne *mot* en majuscules.

Exemples :

Opération	**Résultat**
MAJUSCULE "bonjour	BONJOUR
MAJUSCULE "petit	PETIT

```
POUR COULEURP :MOT
SI MEMBREP MAJUSCULE :MOT <ROUGE BLEU J!
AUNE><RT "VRAI><RT "FAUX>
FIN

?EC COULEURP "rouge
VRAI
?EC COULEURP "vert
FAUX
```

MINUSCULE

MINUSCULE *mot* (opération)

Cette opération retourne *mot* en minuscules.

Exemples :

Opération	**Résultat**
MINUSCULE "Bonjour	bonjour
MINUSCULE "GROS	gros

```
POUR OUIP :MOT
SI EGALP MINUSCULE :MOT "oui <RT "VRAI>!
<RT "FAUX>
FIN

?EC OUIP "OUI
VRAI
?EC OUIP "SEPT
FAUX
```

Variables

Variables

Ce chapitre vous donne des indications sur l'utilisation des variables en Logo et vous fournit une description de chacune des primitives que vous pouvez utiliser avec les variables. Ces primitives sont les suivantes :

CHOSE
EDN
EDNS
LOCALE
NOMME
NOMP
RELIE

Quelques indications sur les variables

D'autres détails sur les variables sont fournis au chapitre 2.

Une **variable** est un contenant qui renferme un objet Logo. Ce contenant porte un nom et a une valeur. L'objet contenu dans le contenant est appelé la **valeur** de la variable. Vous pouvez créer une variable de deux façons : en utilisant la commande RELIE ou NOMME, ou en employant les données d'une procédure.

En Logo, il existe deux types de variables : les variables locales et les variables globales. Les variables servant de données à une procédure sont des variables locales : elles ne sont utilisées que lorsque la procédure est exécutée et disparaissent de l'espace de travail lorsque l'exécution de la procédure est terminée.

Les variables créées au moyen de RELIE sont des variables globales. La commande LOCALE vous permet de transformer celles-ci en variables locales et ainsi libère votre espace de travail des variables que vous n'utilisez plus.

CHOSE

CHOSE *nom* (opération)

L'opération CHOSE retourne l'objet que renferme le contenant *nom*, c'est-à-dire la valeur de la variable *nom*; CHOSE "QUELCONQUE est équivalent à :QUELCONQUE.

Exemple :

La procédure suivante utilise un incrément, c'est-à-dire une valeur constante (dans le cas présent, la valeur 1) qui s'ajoute à celle de la variable.

```
POUR INC :X
SI NON NOMP :X<STOP>
SI NOMBREP CHOSE :X<RELIE :X 1 + CHOSE!
 :X>
FIN
```

Prenez note de l'utilisation de RELIE :X plutôt que de RELIE "X car ce n'est pas à X que doit s'appliquer l'incrément. La valeur de X n'est pas un nombre mais le nom d'une autre variable; c'est à la valeur de cette dernière que l'incrément doit s'appliquer.

D'autres exemples sont fournis sous la rubrique RELIE.

```
?RELIE "TOTAL 7
?EC :TOTAL
7
?INC "TOTAL
?EC :TOTAL
8
?INC "TOTAL
?EC :TOTAL
9
```

EDN

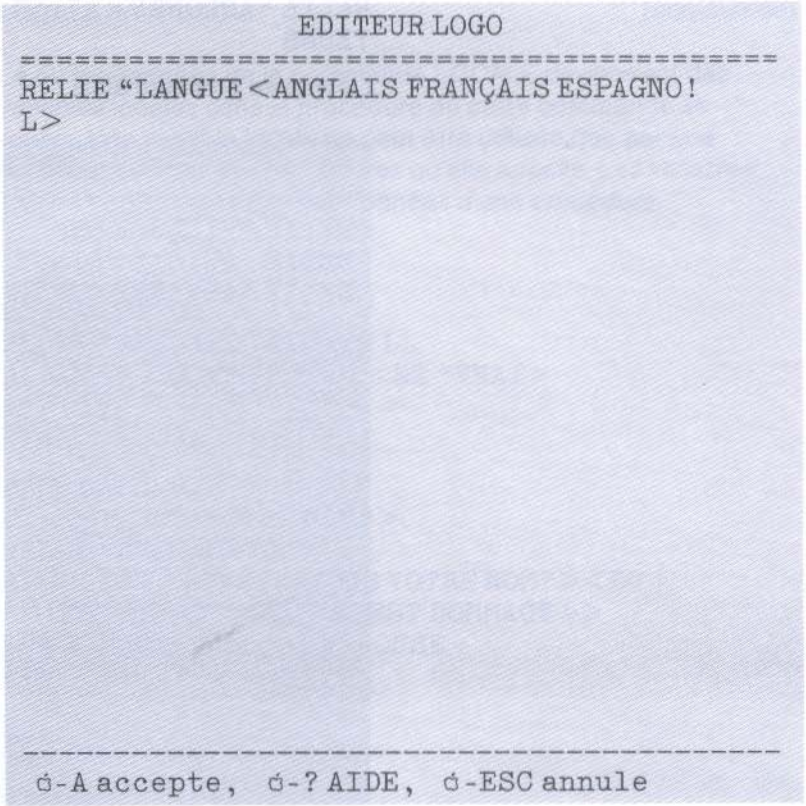
EDN *nom(liste)* (commande)

La commande EDN (édite nom) fait en sorte de ramener l'éditeur Logo qui affiche alors le(s) nom(s) de la (des) variable(s) ainsi que la (les) valeur(s) qui lui (leur) corresponde(nt). Vous pouvez éditer le (les) nom(s) de la (des) variable(s) et sa (leurs) valeur(s). Lorsque vous quittez l'éditeur, Logo lit le contenu du tampon d'édition comme si vous aviez tapé chaque ligne au niveau supérieur. Les modifications que vous avez apportées aux variables ou aux valeurs en mode d'édition sont acceptées.

Exemple :

```
?EDN "LANGUE
```

L'écran apparaît ainsi :



Vous pouvez maintenant apporter des modifications à la variable puis presser [ó] - [A] pour quitter l'éditeur.

EDNS

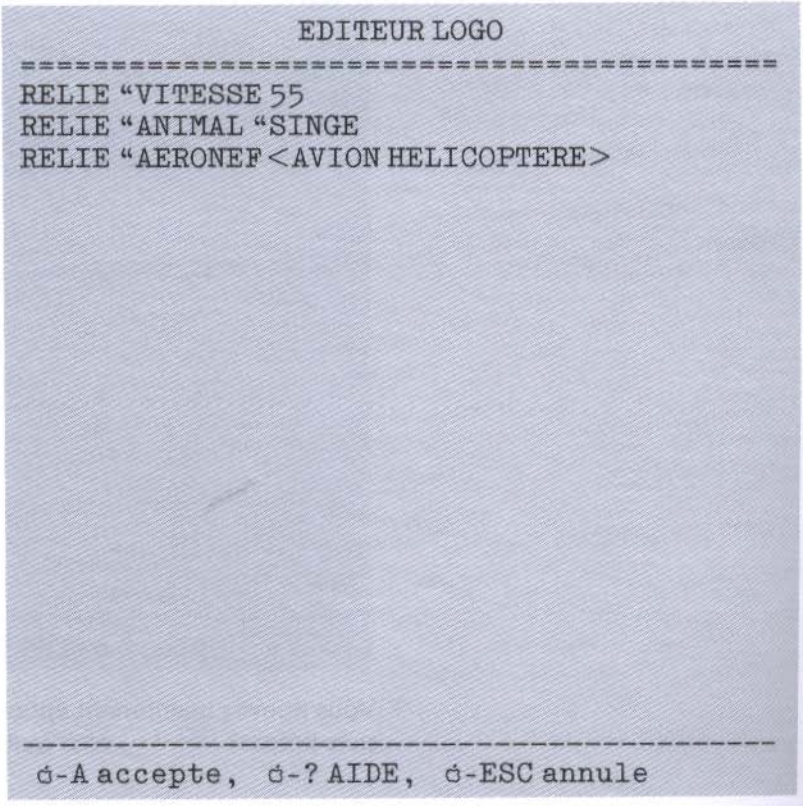
EDNS (commande)

La commande EDNS (édite noms) ramène l'éditeur Logo avec toutes les variables et leurs valeurs. Vous pouvez alors éditer ces noms de variables et leurs valeurs. Lorsque vous quittez l'éditeur, Logo lit le contenu du tampon d'édition comme si vous aviez tapé chaque ligne au niveau supérieur. Les modifications apportées aux variables ou aux valeurs en mode d'édition sont acceptées.

Exemple :

```
?IMNS
RELIE "VITESSE 55
RELIE "ANIMAL "SINGE
RELIE "AERONEF <AVION HELICOPTERE>
?EDNS
```

L'écran se présente ainsi :



Si vous effectuez les modifications suivantes :

```
RELIE "ANIMAL "SOURIS
RELIE "VITESSE 55
RELIE "AERONEF <AVION HELICOPTERE PLAN!
EUR
```

puis quittez l'éditeur, vous obtenez :

```
?IMNS
RELIE "ANIMAL "SOURIS
RELIE "VITESSE 55
RELIE "AERONEF <AVION HELICOPTERE PLANE!
UR>
```

LOCALE

LOCALE *nom(liste)* (commande)

La ou les données de la commande LOCALE deviennent des variables locales dans la procédure où cette commande se trouve. Une variable locale ne peut être utilisée que par une procédure ou par des procédures qu'elle appelle. Les variables locales sont semblables aux données d'une procédure.

```
POUR OUINON :QUESTION
LOCALE "REPONSE
EC :QUESTION
RELIE "REPONSE PREMIER LL
SI EGALP :REPONSE "OUI <RT "VRAI>
RT "FAUX
FIN

POUR SALUER
EC <QUEL EST VOTRE NOM?>
RELIE "REPONSE LL
SI OUINON <AIMEZ!-VOUS VOTRE NOM?><EC !
<C'EST BIEN>><EC <C'EST DOMMAGE>>
EC PH <ENCHANTE,> :REPONSE
FIN
```

```
?SALUER
QUEL EST VOTRE NOM?
MARIE TREMBLAY
AIMEZ-VOUS VOTRE NOM?
NON
C'EST DOMMAGE
ENCHANTE, MARIE TREMBLAY
```

Voici ce qui se produirait si la commande LOCALE n'apparaissait pas dans la procédure OUINON ; cette dernière détruirait la valeur de la variable REPONSE que SALUER s'attend à y trouver. En omettant la commande LOCALE, les variables ne sont pas locales, c'est-à-dire que chaque procédure utilise une variable appelée REPONSE qui contient la réponse de l'utilisateur à chacune des questions.

```
?SALUER
QUEL EST VOTRE NOM?
MARIE TREMBLAY
AIMEZ-VOUS VOTRE NOM?
NON
C'EST DOMMAGE
ENCHANTE, NON
```

NOMME

NOMME *obj nom* (commande)

La commande NOMME place *obj* dans le contenant *nom*, c'est-à-dire qu'elle attribue la valeur *obj* à la variable *nom*.

Exemples :

```
?NOMME 259 "EMPLOI
?EC :EMPLOI
259
?NOMME "SOUDEUR "EMPLOI
?EC :EMPLOI
SOUDEUR
```

La commande NOMME est équivalente à RELIE sauf que ses données sont inversées. Ainsi NOMME "SOUDEUR "EMPLOI a le même effet que RELIE "EMPLOI "SOUDEUR.

NOMP

NOMP *mot* (opération)

L'opération NOMP retourne VRAI si *mot* a une valeur, c'est-à-dire si *mot* existe ; sinon, NOMP retourne FAUX.

Exemples :

```
?EC NOMP "ANIMAL
FAUX
?RELIE "ANIMAL "RHINOCEROS
?EC :ANIMAL
RHINOCEROS
?EC NOMP "ANIMAL
VRAI
```

Une autre utilisation de la primitive NOMP apparaît dans la procédure INC ; celle-ci suit la description de l'opération CHOSE.

RELIE

RELIE *nom obj* (commande)

La commande RELIE place *obj* dans le contenant *nom*, c'est-à-dire qu'elle donne à la variable *nom* la valeur *obj*.

Exemples :

```
RELIE "EMPLOI 259
?EC :EMPLOI
259
?RELIE "EMPLOI "SOUDEUR
?EC :EMPLOI
SOUDEUR
?RELIE "SOUDEUR 32
?EC :SOUDEUR
32
?EC CHOSE :EMPLOI
32
?RELIE :EMPLOI <ALAIN TOUGAS>
```

A ce stade, :EMPLOI est SOUDEUR et CHOSE :EMPLOI est <ALAIN TOUGAS>.

```
?EC "EMPLOI
EMPLOI
?EC :EMPLOI
SOUDEUR
?EC CHOSE "EMPLOI
SOUDEUR
?EC CHOSE :EMPLOI
ALAIN TOUGAS
```

```
POUR TEMPS
EC <QUEL TEMPS FAIT! -IL AUJOURD'HUI?>
RELIE "REPONSE LL
SI :REPONSE = <PLUVIEUX><EC <JE SOUHAI!
TE QUE LA PLUIE CESSE> STOP>
SI :REPONSE = <ENSOLEILLE><EC <J'ESPER!
E QUE ÇA CONTINUERA> STOP>
EC (PH <JE ME DEMANDE SI CE SERA> :REPO!
NSE "DEMAIN)
FIN
```

```
?TEMPS
QUEL TEMPS FAIT-IL AUJOURD'HUI?
ENSOLEILLE
J'ESPERE QUE ÇA CONTINUERA
?TEMPS
QUEL TEMPS FAIT-IL AUJOURD'HUI?
NUAGEUX
JE ME DEMANDE SI CE SERA NUAGEUX DEMAIN
?TEMPS
QUEL TEMPS FAIT-IL AUJOURD'HUI?
PLUVIEUX
JE SOUHAITE QUE LA PLUIE CESSE
```

Opérations arithmétiques

Opérations arithmétiques

Ce chapitre présente les opérations arithmétiques que l'on peut exprimer sous deux formes, soit la **forme préfixée** et la **forme infixée**. Le nom de la primitive se place avant les données si l'on utilise la forme préfixée et entre les données si l'on emploie la forme infixée.

Ce chapitre comprend :

- une introduction aux opérations arithmétiques Logo ;
- une description des opérations de forme préfixée ;
- une description des opérations de forme infixée.

Quelques indications sur les opérations arithmétiques

Logo reconnaît les nombres entiers et les nombres décimaux.

3 est un nombre entier
3.14 et 3. sont des nombres décimaux

Des primitives Logo servent à additionner, soustraire, multiplier et diviser les nombres. Certaines sont aussi utiles pour trouver les sinus, les cosinus, les arcs tangentes et les racines carrées, et pour vérifier si un nombre est égal, inférieur ou supérieur à un autre nombre.

Le résultat d'une opération arithmétique peut être un nombre entier ou un nombre décimal, selon le type d'opération.

- ENTIER, DIV, HASARD, RESTE et ARRONDIS retournent toujours des nombres entiers.
- ARCTAN, COS, SIN, RC, QUOTIENT et / retournent toujours des nombres décimaux.
- Les autres primitives (+, –, *) retournent des entiers si les données sont des nombres entiers. Si les données sont des nombres décimaux, ces primitives retournent des décimales.

Ainsi 7 / 2 est 3.5 (nombre décimal) et DIV 7 2 est 3 (nombre entier).

De plus, 3.5 + 6.5 est 10.0 (nombre décimal) et 3 + 7 est 10 (nombre entier). Remarquez que 3 + 7. est 10.0 (nombre décimal).

L'entier le plus grand que connaît Logo est 2147483647, soit 2^{31}-1; le plus petit est – 2147483647, soit – 2^{31}.

La **notation scientifique** est définie comme une façon de représenter un nombre en utilisant un exposant.

Les nombres décimaux peuvent avoir jusqu'à six chiffres après le point et peuvent être élevés à une puissance comprise entre –38 et 37. Les nombres décimaux à plus de six chiffres sont convertis à la forme exponentielle (notation scientifique). Par exemple :

1.0E10 signifie 10^{10} ou 10000000000

1.0N10 signifie 10^{-10} ou 0.0000000001

N indique un exposant négatif.

Logo arrondit les nombres décimaux qui contiennent plus de six chiffres. Par exemple, 2718281828459.045 devient 2.71828E12.

L'addition, la soustraction, la multiplication et la division ont une forme infixée. Dans ce cas, la notation infixée se place entre les données, jamais avant. L'addition et la multiplication peuvent aussi prendre une forme préfixée qui agit comme une opération Logo. Cette forme demande deux données ou plus. Par exemple, les deux expressions suivantes sont équivalentes :

```
2 + 1
SOMME 2 1
```

La primitive EGALP est décrite au chapitre 7 intitulé "Mots et listes".

Outre les primitives décrites ci-après, EGALP est fréquemment utilisée avec les opérations arithmétiques. L'opération de forme infixée =, décrite dans ce chapitre, en est l'équivalent.

Comment Logo lit les opérations arithmétiques

Lorsqu'une ligne Logo comprend plusieurs opérations mathématiques, Logo les traite selon l'ordre de priorité suivant :

–	moins unaire ; indique un nombre négatif (–3) ou l'inverse additif d'une donnée (–COORX) ;
*, /	multiplication et division ;
+, –	addition et soustraction ;
=	égal
SUPP, INFP	plus grand que, plus petit que.
autres opérations mathématiques	regroupent les opérations que vous définissez ainsi que les primitives d'opération telles SIN, DIFFERENCE et SOMME.

Ainsi

COS 25 + 10

est lu comme suit :

COS (25 + 10)

L'utilisation des parenthèses vous permet de changer cet ordre de priorité. Logo traite d'abord les opérations contenues dans les parenthèses. Toutefois, si ces dernières contiennent plusieurs opérations, Logo utilise l'ordre de priorité décrit ci-dessus.

Exemples :

```
?EC 2 * 4 + 8 / 4
10.0
?EC 2 * (4 + 8 / 4)
12.0
?EC (2 * 4 + 8) / 4
4.0
```

Opérations de forme préfixée

Cette section décrit les opérations de forme préfixée ; elles apparaissent dans l'ordre suivant :

ARCTAN	PRODUIT
ARRONDIS	QUOTIENT

COS
DIFFERENCE
DIV
ENTIER
FORMAT
HASARD
RC
REHASARD
RESTE
SIN
SOMME

ARCTAN

ARCTAN *nombre* (opération)

L'**arc tangente** d'un nombre correspond à un angle dont la tangente est ce nombre.

Cette opération retourne l'arc tangente (tangente inverse) de *nombre*. Son résultat est un nombre décimal compris entre −90 et 90 et s'exprime en degrés. Si *nombre* se rapproche de −1, on ne peut se fier au résultat.

Exemples :

Opération	**Résultat**
ARCTAN 2	63.4348
ARCTAN 444	89.871

Les procédures suivantes définissent ARCSIN et ARCCOS :

```
POUR ARCSIN :X
RETOURNE ARCTAN :X / (RC 1 - :X * :X)
FIN

POUR ARCCOS :X
RETOURNE ARCTAN (RC 1 - :X * :X) / :X
FIN
```

ARRONDIS

ARRONDIS *nombre* (opération)

Voir les exemples sous la rubrique ENTIER.

Cette opération retourne *nombre* arrondi à l'entier le plus proche. L'entier maximal est 2 147 483 647.

Exemples :

Opération	Résultat
ARRONDIS 5.2129	5
ARRONDIS 5.5129	6
ARRONDIS .5	1
ARRONDIS –5.8	–6
ARRONDIS –12.3	–12

COS

COS *degrés* (opération)

COS retourne le cosinus de *degrés* ; son résultat est un nombre décimal. Il y a erreur si *degrés* est plus grand que 4.19E6.

Exemples :

Opération	Résultat
COS 60	0.5
COS 30	0.866025

Voici une définition de la fonction tangente :

```
POUR TANGENTE :ANGLE
RETOURNE (SIN :ANGLE) / COS :ANGLE
FIN

?EC TANGENTE 45
1.0
```

DIFFERENCE

DIFFERENCE *nombre1 nombre2* (opération)

Cette opération retourne le résultat obtenu en soustrayant *nombre2* de *nombre1*.

Exemples :

Opération	**Résultat**
DIFFERENCE 7 1	6
DIFFERENCE (5 + 6) (3 * 7)	−10
DIFFERENCE 10 5	5
DIFFERENCE 6.3 107.4	−101.1

DIV

DIV *entier1 entier2* (opération)

Cette opération retourne le résultat de la division de *entier1* par *entier2* en supprimant les décimales. Il y a erreur si *entier2* a comme valeur 0. Si l'une des données est un nombre décimal, Logo retourne un nombre entier.

Exemples :

Opération	**Résultat**
DIV 12 5	2
DIV −12 5	−2
DIV 9 2	4
DIV 3 0	erreur

ENTIER

ENTIER *nombre* (opération)

Voir aussi ARRONDIS.

Cette opération retourne la partie entière de *nombre* en supprimant les décimales, s'il y a lieu. L'entier maximal est 2 147 483 647.

Exemples :

Opération	**Résultat**
ENTIER 5.2129	5
ENTIER 5.5129	5
ENTIER 5	5
ENTIER −5.8	−5
ENTIER −12.3	−12

La procédure suivante indique si un nombre est entier ou non :

```
POUR ENTIERP :N
SI NON NOMBREP :N<RT PH :N<N'EST PAS !
UN NOMBRE>>
RT (COMPTE :N) = (COMPTE ENTIER :N)
FIN

?EC ENTIERP 17
VRAI
?EC ENTIERP 100 / 8
FAUX
?EC ENTIERP "UN
UN N'EST PAS UN NOMBRE
?EC ENTIERP RC 50
FAUX
```

FORMAT

FORMAT *nombre champ décimale* (opération)

Cette opération retourne *nombre* sous la forme d'un mot, dans un nombre d'espaces indiqué par *champ* ; *décimale* correspond au nombre de chiffres après le point. La donnée de *champ* doit être un entier compris entre 1 et 128, celle de *décimale* un entier compris entre 0 et 6.

Si *nombre* ne contient pas suffisamment de chiffres pour occuper tout l'espace indiqué par *champ*, Logo ajoute des espaces avant ce nombre. Le point décimal (.) et le signe moins (—) occupent chacun un espace dans *champ*.

Bien qu'elle s'applique à tous les nombres entiers, FORMAT ne fonctionne qu'avec quelques nombres décimaux. Ce sont :

—999999.0 à 0.000001
0.000001 à 999999.0

Logo affiche tous les autres nombres décimaux en notation scientifique ; FORMAT ne peut traiter ceux-ci. FORMAT doit plutôt retourner le nombre sous la forme d'un mot, justifié à droite, dans le nombre d'espaces indiqué par *champ*.

Note : Les nombres décimaux ne peuvent contenir plus de six *chiffres significatifs*. Cette règle s'applique également à la donnée de FORMAT.

Il y a erreur si *champ* est 0 ou est plus petit que le nombre de chiffres placés avant le point dans *nombre*. Si *décimale* est égal à 0, FORMAT retourne un *nombre* entier. Un ou plusieurs zéros sont ajoutés si *décimale* est plus grand que le nombre de chiffres placés après le point dans *nombre*.

Le résultat de FORMAT est composé du nombre de chiffres après le point indiqué par *décimale*.

Lorsque les données de *champ* et de *décimale* ne varient pas, cette primitive est utile pour imprimer des nombres qui apparaissent en colonne.

Exemples :

```
POUR ALIGNER
(EC FORMAT 14.9 10 2 FORMAT 1.9807 10 2!
 FORMAT 175.90 10 2)
(EC FORMAT 145.876 10 2 FORMAT 324.676 !
10 2 FORMAT 10042 10 2)
FIN
```

```
?ALIGNER
    14.90          1.98        175.90
   145.87        324.67      10042.00
```

HASARD

HASARD *entier* (opération)

Cette opération retourne un entier non négatif, aléatoire, compris entre 0 et le nombre inférieur à *entier*.

Exemple :

HASARD 6 pourrait retourner 0, 1, 2, 3, 4 ou 5. Le programme suivant simule le tirage d'un dé.

```
POUR TIRDE
RETOURNE 1 + HASARD 6
FIN
```

```
?EC TIRDE
3
?EC TIRDE
5
?EC TIRDE
3
```

INFP

INFP *nombre1 nombre2* (opération)

L'opération AVANTP est décrite au chapitre 7.

Cette opération retourne VRAI si *nombre1* est inférieur à *nombre2*; sinon, elle retourne FAUX. Cette opération équivaut à AVANTP mais on ne peut lui fournir que des nombres comme données.

Exemples :

Opération	**Résultat**
INFP 2 3	VRAI
INFP –7 –10	FAUX

PRODUIT

PRODUIT *nombre1 nombre2* (opération)
(PRODUIT *nombre1 nombre2 nombre3...)*

Cette opération retourne le produit de ses données. La notation infixée * équivaut à PRODUIT. S'il y a une seule donnée, PRODUIT retourne cette donnée.

Exemples :

Opération	**Résultat**
PRODUIT 6 2	12
(PRODUIT 2 3 4)	24
PRODUIT 2.5 4	10.0

```
POUR CUBE :N
RT (PRODUIT :N :N :N)
FIN

?EC CUBE 2
8
```

QUOTIENT

QUOTIENT *nombre1 nombre2* (opération)

Cette opération retourne le résultat de la division de *nombre1* par *nombre2*. La notation infixée / équivaut à QUOTIENT. Il y a erreur si *nombre2* a comme valeur 0.

Exemples :

Opération	**Résultat**
QUOTIENT 12 5	2.4
QUOTIENT −12 5	−2.4
QUOTIENT 6 2.5	2.4
QUOTIENT 3.2 0	erreur

RC

RC *nombre* (opération)

L'opération RC retourne la racine carrée de *nombre*. Il y a erreur si *nombre* est négatif.

Exemples :

Opération	**Résultat**
RC 25	5.0
RC 259	16.0935

La procédure suivante retourne la distance entre la position de la Tortue et le point d'origine :

```
POUR DIST.ORIGINE
RT RC SOMME COORX * COORX COORY * COORY
FIN
```

La procédure DISTANCE retourne la distance entre deux positions fournies comme données :

```
POUR DISTANCE :POS1 :POS2
RT RC SOMME CA (PREMIER :POS1 – PREMIER!
 :POS2) CA (DERNIER :POS1 - DERNIER :PO!
S2)
FIN
```

```
POUR CA :N
RT :N * :N
FIN

?EC DISTANCE <-70 10><50 60>
130.0
```

REHASARD

REHASARD (opération)

REHASARD (reproduit hasard) rend le résultat de HASARD reproductible : après avoir effectué REHASARD, les appels à HASARD retournent chaque fois la même séquence de résultats.

Exemple :

```
POUR LANCE.DE :JETS
SI :JETS = 0 <STOP>
EC 1 + HASARD 6
LANCE.DE :JETS - 1
FIN

?LANCE.DE 6
3
2
6
6
3
1
?LANCE.DE 6
5
5
5
1
3
1
?REHASARD
?LANCE.DE 6
4
3
6
6
1
2
```

```
?REHASARD
?LANCE.DE 6
4
3
6
6
1
2
```

RESTE

RESTE *entier1 entier2* (opération)

Cette opération retourne le reste de la division de *entier1* par *entier2*. Le reste est toujours un nombre entier. Si les données ne sont pas des nombres entiers, elles sont converties en nombres entiers. Il y a erreur si *entier2* a comme valeur 0.

Exemples :

Opération	**Résultat**
RESTE 12 10	2
RESTE 12 5	2
RESTE 12 15	12
RESTE −12 5	−2

La procédure suivante détermine si un nombre est pair :

```
POUR PAIRP :NOMBRE
RT 0 = RESTE :NOMBRE 2
FIN
```

```
?EC PAIRP 5
FAUX
?EC PAIRP 12462
VRAI
```

La procédure DIVISEURP, plus générale, indique si la première donnée est un diviseur de la seconde :

```
POUR DIVISEURP :DIVIDENDE :DIVISEUR
RT 0 = RESTE :DIVIDENDE :DIVISEUR
FIN
```

```
?EC DIVISEURP 15 3
VRAI
?EC DIVISEURP 15 4
FAUX
```

SIN

SIN *degrés* (opération)

Voir COS.

L'opération SIN retourne le sinus de *degrés*. Il y a erreur si *degrés* est supérieur à 4.19E6.

Exemple :

SIN 30 retourne 0.5.

SOMME

SOMME *nombre1 nombre2* (opération)
(SOMME *nombre1 nombre2 nombre3...)*

Cette opération retourne la somme de ses données. L'opération de forme infixée + équivaut à SOMME.

S'il y a une seule donnée, SOMME retourne celle-ci.

Exemples :

Opération	**Résultat**
SOMME 5 2	7
(SOMME 1 3 2 −1)	5
SOMME 2.3 2.561	4.861

SUPP

SUPP *nombre1 nombre2* (opération)

Cette opération retourne VRAI si *nombre1* est plus grand que *nombre2*; sinon, l'opération retourne FAUX.

Exemples :

Opération	**Résultat**
SUPP 4 3	VRAI
SUPP –10 –7	FAUX

Opérations de forme infixée

Cette section décrit les opérations de forme infixée; elles apparaissent dans l'ordre suivant :

+
/
=
*
–

Puisque ces symboles d'opération sont des séparateurs de mots, il n'est pas nécessaire d'ajouter des espaces avant et après. Le symbole de division (/) fait exception à cette règle (voir l'explication qui suit). Ainsi les expressions suivantes sont équivalentes :

2 + 5
2+5

Le symbole de division (/) doit être séparé de ses données mais la barre oblique (/), utilisée dans le préfixe d'une disquette (voir le chapitre 15), ne doit pas être précédée ou suivie d'un espace.

4 / 8
3 / 9

Addition

nombre1 + nombre2 (opération forme infixée)

Cette opération retourne la somme des données. Le symbole + équivaut à SOMME qui, elle, est une opération de forme préfixée.

Exemples :

Opération	**Résultat**
5 + 2	7
1 + 3 + 2 + 1	7
2.54 + 12.3	14.84

Division

nombre1 / nombre 2 (opération forme infixée)

Cette opération retourne *nombre1 / nombre 2*. Elle équivaut à QUOTIENT. Il y a erreur si *nombre2* a comme valeur 0.

Exemples :

Opération	**Résultat**
6 / 3	2.0
8 / 3	2.66667
2.5 / 3.8	0.657895
0 / 7	0.0
7 / 0	erreur

Egalité

obj1 = obj2 (opération forme infixée)

L'opération = équivaut à EGALP ; cette dernière est décrite au chapitre 7.

Cette opération retourne VRAI si *obj1* et *obj2* sont des nombres égaux, des mots ou des listes identiques ; sinon, retourne FAUX.

Notez que l'utilisation des parenthèses modifie la façon dont Logo utilise le signe égal ainsi que le démontre l'exemple suivant :

```
PREMIER "3.1416 = 3 retourne F
(PREMIER "3.1416) = 3 retourne VRAI
```

Dans le premier cas, Logo vérifie d'abord si 3.1416 est égal à 3 et passe ensuite le résultat, soit FAUX, à PREMIER qui à son tour retourne le PREMIER de FAUX, soit F.

Exemples :

Opération	**Résultat**
100 = 50 * 2	VRAI
3 = PREMIER "3.1416	VRAI
<LE LA LES> = <LE LA>	FAUX
7. = 7	VRAI (Un nombre est équivalent à l'entier correspondant.)
" = <>	FAUX (Le mot vide et la liste vide ne sont pas identiques.)

Multiplication

*nombre1 * nombre2* (opération forme infixée)

Cette opération retourne le produit des données. Le symbole * équivaut à PRODUIT qui, elle, est une opération de forme préfixée.

Exemples :

Opération	**Résultat**
6 * 2	12
2 * 3 * 4	24
1.3 * 1.3	1.69

La procédure FACTORIELLE retourne la factorielle de sa donnée. Par exemple, FACTORIELLE 5 donne le résultat de 5 * 4 * 3 * 2 * 1.

```
POUR FACTORIELLE :N
SI :N = 0<RT 1><RT :N * FACTORIELLE :!
N-1>
FIN

?EC FACTORIELLE 4
24
?EC FACTORIELLE 1
1
```

Soustraction

nombre1 – nombre2 (opération forme infixée)

Pour plus de détails sur la façon dont Logo utilise le signe moins, voir l'annexe E intitulée "Interprétation".

Cette opération retourne le résultat obtenu en soustrayant *nombre2* de *nombre1*. Si *nombre1* est absent et s'il n'y a pas d'espace après le signe moins, l'expression retourne l'opposé de *nombre 2* (0 – *nombre2*).

Exemples :

```
?EC 7 - 1
6
?EC 7-1
6
?EC PRODUIT 7-1
-7
?EC - 3
-3
?EC -3 - -2
-1
```

La procédure ABS retourne la valeur absolue de sa donnée.

```
POUR ABS :NOMB
RT SI INFP :NOMB 0 <-:NOMB><:NOMB>
FIN
```

```
?EC ABS -35
35
?EC ABS 35
35
```

APPROXIMATION indique si la valeur de deux nombres est rapprochée :

```
POUR APPROXIMATION :A :B
RT INFP (ABS :A - :B) .01
FIN
```

```
?EC APPROXIMATION COORX 100
VRAI
?EC COORX
99.9934
```

Remarquez qu'il y a possibilité de confusion entre le signe moins utilisé avec une donnée et le signe moins employé avec deux données.

Logo résout ainsi le problème :

7–1 est 6
7 – 1 est également 6
7– 1 est également 6
Mais 7 –1 est une paire de nombres (7 et –1).

Contrôle d'exécution et instructions conditionnelles

Contrôle d'exécution et instructions conditionnelles

Ce chapitre décrit les primitives et les touches de fonction que vous pouvez utiliser pour ainsi modifier la façon qu'a Logo d'exécuter une procédure. Ces primitives et ces touches spéciales sont regroupées en cinq catégories :

- des primitives servant aux **instructions conditionnelles** ; celles-ci demandent à Logo d'exécuter différentes instructions, si telle condition est satisfaite ;
- des primitives qui interrompent l'exécution d'une procédure ;
- des primitives qui indiquent à Logo de répéter des instructions un certain nombre de fois, ou de passer, ou encore de remettre le contrôle à une autre instruction ;
- des primitives pour la mise au point de programmes ;
- des touches de fonction spéciales qui interrompent le contrôle d'exécution temporairement ou de façon permanente.

Quelques indications sur le contrôle d'exécution

Logo lit les définitions de procédures ligne par ligne en tenant compte des instructions qui se trouvent dans chaque ligne. Si une procédure contient une sous-procédure, Logo lit les lignes de cette dernière avant de reprendre l'exécution de la superprocédure. Le **contrôle d'exécution** consiste en l'ordre dans lequel Logo traite les instructions. Si vous désirez modifier le contrôle d'exécution de Logo, vous pouvez le faire selon l'une des manières suivantes :

des instructions conditionnelles	indiquent à Logo de faire une chose si telle condition est satisfaite ; sinon, Logo doit faire autre chose ;

des instructions d'itération	indiquent à Logo d'exécuter une liste d'instructions une ou plusieurs fois ;
des instructions d'arrêt	indiquent à Logo de stopper l'exécution d'une procédure avant que la fin de celle-ci ne soit atteinte ;
des instructions de pause	indiquent à Logo d'interrompre momentanément l'exécution de la procédure, et de reprendre son exécution plus tard.

Utilisation des instructions conditionnelles

Les **instructions conditionnelles** permettent à Logo d'exécuter différentes instructions si une condition est satisfaite. Les **prédicats** sont des opérations qui retournent VRAI ou FAUX et qui permettent de créer une condition. Le résultat de l'opération doit être la première donnée de SI, SIFAUX ou SIVRAI. Les primitives utiles à la création de conditions sont les suivantes :

SI
SIFAUX, SIF
SIVRAI, SIV
TESTE

Les primitives SIFAUX, SIVRAI et TESTE accomplissent la même tâche que la primitive SI. C'est à vous de choisir laquelle serait la plus appropriée.

SI

SI *préd liste1* (commande ou opération)
SI *préd liste1 liste2*

Si *préd* est VRAI, Logo exécute *liste1* ; si *préd* est FAUX, Logo exécute *liste2*, s'il y a lieu. Dans les deux cas, si la liste choisie retourne quelque chose, la primitive SI est une opération. Si la liste ne retourne rien, SI est une commande.

Exemples :

La procédure DECIDE est écrite de trois manières différentes. SI en tant que commande apparaît dans les deux premières (l'une avec deux données pour SI, l'autre avec trois données). La troisième façon utilise SI comme opération (avec trois données).

SI en tant que commande

```
POUR DECIDE
SI 0 = HASARD 2 <RT "OUI>
RT "NON
FIN
```

```
POUR DECIDE
SI 0 = HASARD 2 <RT "OUI><RT "NON>
FIN
```

SI comme opération

```
POUR DECIDE
RETOURNE SI 0 = HASARD 2 <"OUI><"NON>
FIN
```

SIFAUX

SIFAUX *liste* (SIF) (commande)

Voir TESTE.

La commande SIFAUX exécute *liste* si le résultat de TESTE utilisée précédemment était FAUX ; sinon, rien ne se passe. Remarquez que si TESTE n'a pas été exécutée dans la même procédure ou dans une superprocédure, ou encore au niveau supérieur, SIFAUX ne fait rien.

Exemple :

```
POUR QUIZ
ECRIS <QUELLE EST LA PLUS GRANDE VILLE !
DU QUEBEC?>
TESTE "MONTREAL = MAJUSCULE LISMOT
SIVRAI <ECRIS "EXACT!>
SIFAUX <ECRIS "FAUX>
FIN
```

```
?QUIZ
QUELLE EST LA PLUS GRANDE VILLE DU QUEB!
EC?
VAUDREUIL
FAUX
```

SIVRAI

SIVRAI *liste* (SIV) (commande)

Voir TESTE.

Cette commande exécute *liste* si le résultat de TESTE utilisée précédemment était VRAI ; sinon, rien ne se passe. Remarquez que si TESTE n'a pas été exécutée dans la même procédure ou dans une superprocédure, ou encore au niveau supérieur, SIVRAI ne fait rien.

Exemple :

```
POUR QUIZ2
EC <QUI EST LE MEILLEUR? >
TESTE "MOI = MAJUSCULE LISMOT
SIVRAI <EC <C'EST JUSTE> STOP>
EC <ESSAIE AUTRE CHOSE>
QUIZ2
FIN
```

```
?QUIZ2
QUI EST LE MEILLEUR?
JULIEN
ESSAIE AUTRE CHOSE
QUI EST LE MEILLEUR?
MOI
C'EST JUSTE
```

TESTE

TESTE *préd* (commande)

Cette commande mémorise si *préd* est VRAI ou FAUX pour que SIFAUX ou SIVRAI puisse ensuite l'utiliser.

Exemple :

```
POUR BIENVENUE
EC <COMMENT ÇA VA? >
TESTE "BIEN = MAJUSCULE LISMOT
SIVRAI <EC <HEUREUX DE L'APPRENDRE>>
FIN
```

```
?BIENVENUE
COMMENT ÇA VA?
MAL

?BIENVENUE
COMMENT ÇA VA?
BIEN
HEUREUX DE L'APPRENDRE
```

Interruption des procédures

Les commandes qui stoppent une procédure temporairement, ou de façon permanente, sont les suivantes :

ATTENDS
CO
PAUSE
RETOURNE, RT
STOP

Pour **stopper** une procédure avant qu'elle n'atteigne le mot FIN, vous devez utiliser les commandes STOP et RETOURNE. Logo rend alors le contrôle à la procédure appelante, c'est-à-dire à la procédure qui l'utilise, ou au niveau supérieur. La commande RETOURNE peut aussi fournir de l'information à la procédure appelante. Remarquez que les commandes STOP et RETOURNE ne stoppent que la procédure dans laquelle elles apparaissent.

Pour interrompre une procédure sans la stopper de façon permanente, utilisez les commandes PAUSE ou ATTENDS : la première est surtout utile à la mise au point de programmes. La seconde est surtout utilisée dans des procédures où il est essentiel de tenir compte du facteur temps, comme dans l'animation graphique.

La description de LISCAR, LISCARS, LISLISTE et LISMOT apparaît au chapitre 13.

Note : D'autres primitives telles LISCAR, LISCARS, LISLISTE et LISMOT permettent aussi d'interrompre des procédures de façon temporaire.

ATTENDS

ATTENDS *entier* (commande)

Cette commande indique à Logo d'attendre pendant *entier* soixantième(s) de seconde.

Exemple :

La procédure DONNE.POS affiche d'une manière continuelle la position de la Tortue ; cette position est déterminée au hasard. La commande ATTENDS vous permet de lire la position avant qu'elle ne change.

```
POUR DONNE.POS
DR 10 * HASARD 36
AV 10 * HASARD 10
EC POS
ATTENDS 100
DONNE.POS
FIN
```

```
?VE CT
?DONNE.POS
0.0 90.0
-46.9846 72.889
-41.7752 43.3547
.
.
.
```

CO

CO (commande)

La commande CO (continue) reprend l'exécution d'une procédure interrompue au moyen de PAUSE ou de CONTROL - Z . L'exécution reprend à l'endroit où elle avait été interrompue.

PAUSE

PAUSE (commande ou opération)

La commande PAUSE ne peut être employée qu'à l'intérieur d'une procédure, non au niveau supérieur. Elle interrompt l'exécution d'une procédure et vous indique que Logo fait une pause ; il vous est alors possible de taper des instructions au clavier. Pour vous indiquer qu'une PAUSE a lieu et que vous ne vous trouvez pas au niveau supérieur, le nom de la procédure interrompue apparaît, suivi du point d'interrogation. Au cours d'une pause, [ó] - [ESC] ne fonctionne pas ; le seul moyen de retourner au niveau supérieur est d'exécuter RENVOIE "NIVEAUSUP.

Pendant une pause, il vous est possible de demander à Logo quelle est la valeur d'une variable. Voir EC :MAX dans l'exemple ci-dessous.

On reprend l'exécution de la procédure en tapant CO (continue).

Exemples :

```
POUR MARCHE :MAX
DR HASARD 360
AV HASARD :MAX
EC POS
PAUSE
MARCHE :MAX
FIN

?MARCHE 100
60.4109 -13.947
PAUSE ...
MARCHE?EC CAP
103
MARCHE?EC :MAX
100
MARCHE?CO
68.43812.1059
```

RETOURNE

RETOURNE *obj* (RT) (commande)

La commande RETOURNE ne peut être employée qu'à l'intérieur d'une procédure, non au niveau supérieur. Elle fait d'*obj* le résultat de votre procédure, et rend le contrôle à la procédure appelante. Même si la primitive RETOURNE est une commande, la procédure dans laquelle elle se trouve est une opération puisque cette procédure retourne quelque chose. Comparer avec STOP.

Exemples :

```
POUR MOLIERE
RT <JEAN! -BAPTISTE POQUELIN>
FIN

?EC PH MOLIERE <EST UN AUTEUR COMIQUE>
JEAN-BAPTISTE POQUELIN EST UN AUTEUR CO!
MIQUE
```

QUELRANG retourne la position d'un élément dans une liste.

```
POUR QUELRANG :MEMBRE :LISTE
SI NON MEMBREP :MEMBRE :LISTE <RT 0>
SI :MEMBRE = PREMIER :LISTE <RT 1>
RT 1 + QUELRANG :MEMBRE SP :LISTE
FIN

?RELIE "VOYELLES <A E I O U Y>
?EC QUELRANG "E :VOYELLES
2
?EC QUELRANG "U :VOYELLES
5
?EC QUELRANG "W :VOYELLES
0
```

Une autre façon de définir la procédure qui retourne la valeur absolue d'un nombre se trouve sous la rubrique Soustraction au chapitre 9.

Voici une façon de définir une procédure qui retourne la valeur absolue d'un nombre :

```
POUR ABS :N
SI INFP :N 0 <RT -:N> <RT :N>
FIN
```

STOP

STOP (commande)

La commande STOP arrête l'exécution de la procédure et rend le contrôle à la procédure appelante. Cette commande ne peut être employée qu'à l'intérieur d'une procédure, non au niveau supérieur. Remarquez que la procédure où apparaît STOP est une commande. Comparer à RETOURNE.

Exemples :

```
POUR REBOURS :N
EC :N
SI :N = 0<EC<C'EST PARTI!>STOP>
REBOURS :N - 1
FIN

?REBOURS 4
4
3
2
1
0
C'EST PARTI!
```

Transfert du contrôle d'exécution et instructions d'itération

Cette section fournit une description de chacune des primitives que vous pouvez utiliser pour répéter une liste d'instructions ou pour passer le contrôle d'exécution à d'autres instructions. Ces primitives sont les suivantes :

ATTRAPE
ERREUR
ETIQUETTE
EXECUTE
RENVOIE
REPETE
VA

Quatre primitives servent à passer ou à rendre le contrôle d'exécution à d'autres instructions. Les primitives ETIQUETTE et VA sont utilisées pour rendre le contrôle d'exécution à une instruction apparaissant dans la procédure même ; ATTRAPE et RENVOIE servent à remettre le contrôle à une autre procédure. Vous pouvez utiliser ATTRAPE et RENVOIE pour arrêter l'exécution d'un programme.

Voir sous la rubrique EXECUTE des exemples de procédures d'itération.

On utilise la primitive REPETE ou une procédure récursive pour **répéter** une liste d'instructions. Vous en trouverez des exemples dans des procédures apparaissant au fil des chapitres de ce manuel.

ATTRAPE

ATTRAPE *nom liste* (commande)

La primitive ATTRAPE exécute *liste*. Si la commande RENVOIE *nom* est appelée pendant l'exécution de *liste*, le contrôle revient à la première indication qui suit la commande ATTRAPE. Le *nom* sert à faire correspondre un RENVOIE et un ATTRAPE. Par exemple, ATTRAPE "CHAISE <quelque chose> attrape un RENVOIE "CHAISE, non un RENVOIE "TABLE.

Il existe un cas particulier. ATTRAPE "ERREUR attrape une erreur qui, autrement, provoquerait l'affichage d'un message Logo et un retour au niveau supérieur. Si une erreur est attrapée, Logo n'affiche pas de message alors qu'il devrait normalement le faire. Voir la description de ERREUR apparaissant dans ce chapitre pour savoir comment déterminer quelle était l'erreur.

Exemples :

La procédure SERPENT lit les nombres fournis par l'utilisateur ; ceux-ci déterminent la distance que la Tortue doit parcourir. Cette procédure fait pivoter la Tortue entre ses déplacements. Si vous tapez autre chose qu'un nombre, le programme (au moyen de la sous-procédure LISNOMBRE) affiche un message et continue à fonctionner.

```
POUR SERPENT                                    (superprocédure)
ATTRAPE "NONNOMBRE <RAMPER>
SERPENT
FIN

POUR RAMPER                                     (sous-procédure)
EC <TAPEZ UN NOMBRE, S'IL VOUS PLAIT>
AV LISNOMBRE
DR 10
FIN
```

```
POUR LISNOMBRE                                    (sous-procédure)
LOCALE "NOMBRE
RELIE "NOMBRE LL
SI NON NOMBREP PREMIER :NOMBRE<EC<CEC!
I N'EST PAS UN NOMBRE.>RENVOIE "NONNOM!
BRE>
SI NON VIDEP SP :NOMBRE<EC<UN SEUL NO!
MBRE, S'IL VOUS PLAIT.>RENVOIE "NONNOM!
BRE>
RETOURNE PREMIER :NOMBRE
FIN
```

Remarquez que l'utilisation de la commande STOP (au lieu de RENVOIE "NONNOMBRE) aurait renvoyé à RAMPER, non à SERPENT.

La procédure AGIS exécute les *instructions* fournies par l'utilisateur. Lorsqu'il y a une erreur, Logo n'affiche pas le message usuel et ne revient pas au niveau supérieur ; il affiche plutôt VOS INSTRUCTIONS SONT INCORRECTES et vous laisse taper vos instructions.

```
POUR AGIS
ATTRAPE "ERREUR<AGIS1>
EC<VOS INSTRUCTIONS SONT INCORRECTES>
AGIS
FIN

POUR AGIS1
EXECUTE LISLISTE
AGIS1
FIN

?AGIS
EC 3+5
8
EC12-7
VOS INSTRUCTIONS SONT INCORRECTES
EC 12-7
5
RENVOIE "NIVEAUSUP
```

ERREUR

ERREUR (opération)

Vous trouverez à l'annexe A une liste complète des numéros correspondant aux erreurs ainsi que leur signification.

L'opération ERREUR retourne une liste de quatre éléments ; celle-ci contient des indications sur l'erreur la plus récente pour laquelle aucun message n'a été affiché ou retourné par ERREUR. S'il n'y a pas eu d'erreur, ERREUR retourne la liste vide. Les éléments contenus dans la liste sont les suivants :

- un nombre qui identifie l'erreur ;
- un message expliquant l'erreur ;
- le nom de la primitive qui a causé l'erreur, s'il y a lieu ;
- le nom de la procédure à l'intérieur de laquelle l'erreur s'est produite (la liste vide s'il s'agit du niveau supérieur).

Logo exécute RENVOIE "ERREUR à chaque fois qu'une erreur se produit pendant l'exécution d'une procédure. Le contrôle est rendu au niveau supérieur à moins que ATTRAPE "ERREUR ne soit exécutée. Lorsqu'une erreur est attrapée de cette manière, le message usuel n'est pas affiché ; vous pouvez alors écrire le message de votre choix.

Exemples :

```
POUR CARRE.DE :COTE
ATTRAPE "ERREUR<REPETE 4<AV :COTE DR !
90 >STOP>
EC ERREUR
FIN

?CARRE.DE "15CENTIMETRES
41 <AV N'AIME PAS 15CENTIMETRES COMME D!
ONNEE DANS CARRE.DE> AV CARRE.DE
```

CARRE.DE exécute ATTRAPE "ERREUR et imprime ERREUR si une erreur se produit. Vous pouvez modifier la procédure de sorte que votre propre message s'affiche.

```
POUR CARRE.DE :COTE
ATTRAPE "ERREUR<REPETE 4<AV :COTE DR !
90 >STOP>
EC<ZUT, UNE ERREUR!>
FIN

?CARRE.DE "SIX
ZUT, UNE ERREUR!
```

ETIQUETTE

ETIQUETTE *mot* (commande)

Voir VA.

La commande ETIQUETTE elle-même n'a aucun effet. Mais VA *mot* donne le contrôle à l'instruction qui la suit. Remarquez que *mot* doit toujours être un mot littéral, c'est-à-dire qu'il doit être précédé des guillemets.

EXECUTE

EXECUTE *liste* (commande ou opération)

Cette commande exécute *liste* comme si celle-ci avait été tapée au clavier. Si *liste* est une opération, EXECUTE retourne alors ce que *liste* retourne.

Exemples :

```
POUR CALCULATEUR
EC EXECUTE LL
EC <>
CALCULATEUR
FIN

?CALCULATEUR
2 + 3
5

17.5 * 3
52.5

42 = 8 * 7
FAUX

RESTE 12 5
2
```

La procédure TANTQUE exécute une liste d'instructions tant que la condition indiquée est vraie :

```
POUR TANTQUE :CONDITION :LISTE
TESTE EXECUTE :CONDITION
SIFAUX <STOP>
EXECUTE :LISTE
TANTQUE :CONDITION :LISTE
FIN
```

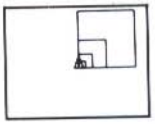
```
?DR 10
?TANTQUE <COORX < 100><AV 25 EC POS>
```

La procédure suivante applique une commande à chacun des éléments d'une liste.

```
POUR DONNE :COMMANDE :LISTE
SI VIDEP :LISTE <STOP>
EXECUTE LISTE :COMMANDE MOT "" PREMIER !
:LISTE
DONNE :COMMANDE SP :LISTE
FIN

POUR CARRE :COTE
REPETE 4 <AV :COTE DR 90>
FIN

?DONNE "CARRE <10 20 40 80>
```

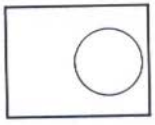


```
?RELIE "PAYS <FRANCE ANGLETERRE CANADA!
BELGIQUE>
?DONNE "EC :PAYS
FRANCE
ANGLETERRE
CANADA
BELGIQUE
```

La procédure SANSFIN répète indéfiniment sa donnée (à moins qu'une erreur ne se produise ou qu'on ne stoppe la procédure au moyen de [ó] - [ESC]) :

```
POUR SANSFIN :LISTE
EXECUTE :LISTE
SANSFIN :LISTE
FIN
```

La commande `SANSFIN <AV 1 DR 1 >` fait tracer un cercle à la Tortue.



La commande SANSFIN <EC EXECUTE LL EC< >> est équivalente à la procédure CALCULATEUR définie auparavant.

La procédure CARRE.SUR trace un carré ; si le crayon était levé avant l'exécution de la procédure, il se replace dans cet état une fois l'exécution terminée.

```
POUR CARRE.SUR
RELIE "GARDERETAT CRAYON
BC
CARRE 100
EXECUTE (PH :GARDERETAT)
FIN

POUR CARRE :LONG
REPETE 4 <AV :LONG DR 90>
FIN

?MONTRE CRAYON
LC
?CARRE.SUR
?MONTRE CRAYON
LC
```

EXECUTE LL	exécute les commandes que vous tapez.
EC EXECUTE LL	affiche le résultat de toute donnée que vous fournissez.

RENVOIE

RENVOIE *nom* (commande)

Voir ATTRAPE.

La commande RENVOIE ne peut être employée qu'avec la commande ATTRAPE. Il y a erreur s'il ne se trouve pas de ATTRAPE *nom* qui corresponde à la commande RENVOIE.

RENVOIE "NIVEAUSUP redonne le contrôle au niveau supérieur. Comparer à STOP.

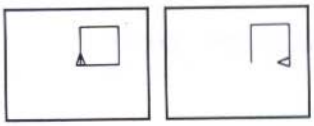
REPETE

REPETE *entier liste* (commande)

La commande REPETE exécute *liste* le nombre de fois indiqué par *entier*. Il y a erreur si *entier* est négatif.

Exemples :

`REPETE 4 <AV 100 DR 90>` trace un carré dont chacun des côtés a 100 pas de longueur. `REPETE 3 <AV 100 DR 90>` trace les trois quarts d'un carré.



VA

VA *mot* (commande)

La commande VA transfère le contrôle à l'instruction qui suit ETIQUETTE *mot* dans la même procédure.

Exemple :

```
POUR REBOURS1 :N
ETIQUETTE "BOUCLE
SI INFP :N 0<STOP>
ECRIS :N
RELIE "N :N - 1
VA "BOUCLE
FIN
```

Mise au point de programmes

Les primitives décrites dans cette section servent à l'analyse et à la mise au point de programmes. Ce sont les suivantes :

NONPAP
NONTRACE
PAP
TRACE

NONPAP

NONPAP *nom(liste)* (commande)

La commande NONPAP (non pas à pas) annule l'effet de PAP (pas à pas) pour la (ou les) procédure(s) *nom(liste)*. Après l'emploi de PAP, vous devez commander NONPAP pour que la procédure s'exécute normalement.

Exemple :

```
?NONPAP "TRIANGLE
?TRIANGLE "IL
IL
I
?
```

NONTRACE

NONTRACE *nom(liste)* (commande)

La commande NONTRACE stoppe le processus débuté au moyen de TRACE ; la procédure s'exécute alors normalement.

Exemple :

```
?NONTRACE "COMPTER
?COMPTER 5
9
8
7
6
5
?
```

PAP

PAP *nom(liste)* (commande)

La commande PAP (pas à pas) vous permet de faire exécuter ligne par ligne la (ou les) procédure(s) *nom(liste)*. Cette commande provoque un arrêt après chaque ligne d'instructions. L'exécution se poursuit seulement lorsque vous pressez une touche du clavier.

Exemples :

```
POUR TRIANGLE :MOT
SI VIDEP :MOT <STOP>
EC :MOT
TRIANGLE SD :MOT
FIN

?PAP "TRIANGLE
?TRIANGLE "IL
SI VIDEP :MOT <STOP>          Pressez une touche.

EC :MOT                       Pressez une touche.
IL
TRIANGLE SD :MOT              Pressez une touche.
SI VIDEP :MOT <STOP>          Pressez une touche.

EC :MOT                       Pressez une touche.
I
TRIANGLE SD :MOT              Pressez une touche.
SI VIDEP :MOT <STOP>          Pressez une touche.
?
```

TRACE

TRACE *nom(liste)* (commande)

Logo affiche la ligne titre et la donnée, s'il y a lieu, de la procédure *nom(liste)* soumise à l'effet de TRACE. Cela n'interrompt pas l'exécution de la procédure mais vous permet de voir ce qu'accomplit la procédure lors de l'exécution. La commande TRACE sert surtout à comprendre le fonctionnement des procédures récursives ou de programmes complexes comprenant plusieurs sous-procédures.

Exemples :

```
?IMPS
POUR COMPTER :N
SI :N = 10 <STOP>
COMPTER :N + 1
EC :N
FIN
?TRACE "COMPTER
?COMPTER 5
 COMPTER 5
  COMPTER 6
   COMPTER 7
    COMPTER 8
     COMPTER 9
      COMPTER 10
      COMPTER finie
9
     COMPTER finie
8
    COMPTER finie
7
   COMPTER finie
6
  COMPTER finie
5
 COMPTER finie
?
```

Touches de fonction

Les touches spéciales décrites dans cette section interrompent le contrôle d'exécution de Logo temporairement ou de façon permanente.

POMME VIDE-ESC

ó - ESC (touche spéciale)

ó - ESC stoppe immédiatement l'exécution en cours puis retourne au niveau supérieur à moins qu'une pause n'ait été commandée.

CONTROL-W

CONTROL - W (touche spéciale)

Cette combinaison de touches interrompt l'exécution en cours. L'exécution normale reprend si l'on presse une touche. CONTROL - W est particulièrement utile lorsque Logo affiche de l'information apparaissant sur plus d'une page d'écran ; cela vous permet de lire ce qui est affiché.

CONTROL-Z

CONTROL - Z (touche spéciale)

CONTROL - Z interrompt l'exécution en cours et provoque une pause. Son effet est équivalent à celui de PAUSE mais son utilisation est différente : vous pressez les touches CONTROL - Z au cours de l'exécution d'une procédure alors que PAUSE fait partie de la définition d'une procédure.

Modification des procédures à l'aide de primitives spéciales

Modification des procédures à l'aide de primitives spéciales

Ce chapitre explique les caractéristiques de Logo qui vous permettent d'écrire des procédures pour définir et modifier d'autres procédures. Les primitives utilisées pour arriver à un tel résultat sont les suivantes :

COPIEDEF
DEFINIEP
DEFINIS
PRIMITIVEP
TEXTE

L'utilisation de DEFINIS et de TEXTE permet de définir ou de modifier des procédures à l'intérieur d'autres procédures. DEFINIS fait d'une liste d'instructions une procédure. TEXTE, à l'inverse de DEFINIS, retourne la définition d'une procédure sous la forme d'une liste. Cette dernière peut être modifiée en utilisant les techniques de traitement de listes décrites au chapitre 7.

Voir le chapitre 10 pour plus de détails au sujet de EXECUTE.

L'utilisation de ces techniques vous permet de créer des listes totalement nouvelles. DEFINIS les transforme ensuite en procédures et les place dans votre espace de travail. Si vous voulez faire exécuter les listes mais ne voulez pas les conserver dans l'espace de travail, employez la primitive EXECUTE plutôt que DEFINIS.

PRIMITIVEP et DEFINIEP vérifient si le nom d'une procédure existe déjà. Elles peuvent être très utiles pour écrire des programmes de mise au point et pour éviter certaines erreurs.

COPIEDEF fait une copie d'une procédure et la place sous un nouveau nom. COPIEDEF vous permet de créer une copie de sauvegarde ; cette copie peut être très utile dans le cas où l'emploi de DEFINIS rend inutilisable la procédure déjà existante.

COPIEDEF

COPIEDEF *nom nouvnom* (commande)

COPIEDEF (copie définition) copie la définition de *nom* et en fait alors la définition de *nouvnom.*

Exemples :

`COPIEDEF "CARRE "NOUVEAUCARRE` donne à `NOUVEAUCARRE` la même définition que `CARRE`.

`COPIE "AVANCE "A` donne à `A` la même définition que `AVANCE`.

DEFINIEP

DEFINIEP *mot* (opération)

Cette opération retourne VRAI si *mot* correspond au nom d'une procédure définie par l'utilisateur ; sinon, retourne FAUX.

DEFINIS

DEFINIS *nom liste* (commande)

DEFINIS fait de *liste* la définition de la procédure *nom.* Le premier élément de *liste* est une liste composée des données de *nom,* sans qu'elles soient précédées du deux points (:).

Si *nom* n'a pas de donnée, le premier élément de *liste* doit être la liste vide. Chacun des éléments de la liste est une liste composée d'une ligne de la définition de la procédure. Cette liste ne contient pas FIN car FIN ne fait pas partie de la définition de la procédure.

La deuxième donnée de DEFINIS a la même forme que ce que TEXTE retourne. DEFINIS peut servir à redéfinir une procédure qui existe déjà.

Exemples :

```
DEFINIS "CARRE <<COTE><REPETE 4 <AV : !
COTE DR 90>>>
```

équivaut à :

```
POUR CARRE : COTE
REPETE 4 <AV : COTE DR 90>
FIN
```

APPRENTISSAGE est un programme qui vous permet de taper successivement les lignes définissant une procédure qui ne comporte pas de données. Chaque fois que vous pressez ↵, Logo exécute l'instruction et l'intègre à la définition de la procédure. Vous pouvez effacer la ligne précédente en tapant EFFACE.

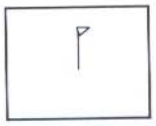
```
POUR APPRENTISSAGE
RELIE "PRO << >>
LISLIGNES
EC <VOULEZ ! - VOUS GARDER CECI COMME DEFI !
NITION D'UNE PROCEDURE ? >
TESTE (PREMIER PREMIER LL) = "O
SIVRAI <TAPE <NOM DE LA PROCEDURE ? > DEFI !
NIS PREMIER LL : PRO>
FIN
```

```
POUR LISLIGNES
RELIE "LIGNESUIV LL
SI : LIGNESUIV = <FIN><STOP>
TESTE : LIGNESUIV = <EFFACE>
SIVRAI <ANNULE>
SIFAUX <EXECUTE : LIGNESUIV RELIE "PRO M !
D : LIGNESUIV : PRO>
LISLIGNES
FIN
```

```
POUR ANNULE
EC PH <JE VAIS EFFACER LA LIGNE> DERNIE !
R : PRO
RELIE "PRO SD : PRO
FIN
```

```
?APPRENTISSAGE
AV 20
DR 36
EFFACE
JE VAIS EFFACER LA LIGNE DR 36
DR 72
FIN
VOULEZ-VOUS GARDER CECI COMME DEFINITIO!
N D'UNE PROCEDURE?
OUI
NOM DE LA PROCEDURE? PATTE

?IM "PATTE
POUR PATTE
AV 20
DR 72
FIN
```



PRIMITIVEP

PRIMITIVEP *nom* (opération)

Cette opération retourne VRAI si *nom* est le nom d'une primitive ; sinon, retourne FAUX.

Exemples :

Opération	**Résultat**
PRIMITIVEP "AVANCE	VRAI
PRIMITIVEP "CARRE	FAUX

TEXTE

TEXTE *nom* (opération)

La primitive TEXTE retourne la définition de *nom* sous la forme d'une liste de listes ; celle-ci peut être utilisée comme donnée de DEFINIS.

Exemples :

```
?MONTRE TEXTE "POLY
<<COTE ANGLE><AV :COTE DR :ANGLE><POL!
Y :COTE :ANGLE>>
```

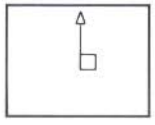
Le premier élément de ce que retourne TEXTE est une liste des données de la procédure. Les autres éléments sont des listes ; chacune correspond à une ligne de la définition de la procédure. Si la procédure *nom* n'est pas définie, TEXTE retourne la liste vide. L'exemple ci-dessus correspond à :

```
?IM "POLY
POUR POLY :COTE :ANGLE
AV :COTE DR :ANGLE
POLY :COTE :ANGLE
FIN
```

TEXTE peut être utilisée avec DEFINIS pour créer des procédures qui modifient d'autres procédures. Par exemple :

```
?IM "CARRE
POUR CARRE
REPETE 4<AV 30 DR 90>
FIN
?DEFINIS "CARRE.AVEC.QUEUE MD<AV 100>!
TEXTE "CARRE
```

```
?IM "CARRE.AVEC.QUEUE
POUR CARRE.AVEC.QUEUE
REPETE 4<AV 30 DR 90>
AV 100
FIN
```



La PRIMITIVE PAP est décrite au chapitre 10.

Exemple plus complexe :

La procédure PASAPAS modifie la définition d'une procédure de sorte qu'elle s'exécute une ligne à la fois. PASAPAS est semblable à la primitive PAP. L'exemple qui suit démontre comment modifier la définition d'une procédure.

Après l'exécution de chaque ligne, Logo attend que vous tapiez ↵ pour continuer. ARRET.PASAPAS rétablit la définition créée au départ.

Le programme :

```
POUR PASAPAS :PRO
COPIEDEF :PRO MOT "." :PRO
RELIE "PREDEF TEXTE :PRO
RELIE "NOUVDEF (LISTE PREMIER :PREDEF)
RELIE "NOUVDEF MD (LISTE "ECRIS (LISTE !
"J'EXECUTE :PRO)) :NOUVDEF
MONTREDONNEES PREMIER :PREDEF
MONTRELIGNES SP :PREDEF
DEFINIS :PRO :NOUVDEF
FIN

POUR IGNORE :DONNEES
FIN

POUR PAS
TAPE "
IGNORE LL
FIN
```

```
POUR MONTRELIGNES :INSTRUCTIONS
SI VIDEP :INSTRUCTIONS <STOP>
RELIE "NOUVDEF MD (LISTE "TAPE PREMIER !
:INSTRUCTIONS) :NOUVDEF
RELIE "NOUVDEF MD <PAS> :NOUVDEF
RELIE "NOUVDEF MD PREMIER :INSTRUCTIONS!
 :NOUVDEF
MONTRELIGNES SP :INSTRUCTIONS
FIN

POUR MONTREDONNEES :LISTEDONNEES
SI VIDEP :LISTEDONNEES <STOP>
RELIE "NOUVDEF MD (LISTE "ECRIS "PHRASE!
 (LISTE (PREMIER :LISTEDONNEES) "EST) (!
MOT " :PREMIER :LISTEDONNEES)) :NOUVDEF
MONTREDONNEES SP :LISTEDONNEES
FIN

POUR ARRET.PASAPAS :PRO
COPIEDEF MOT ". :PRO :PRO
EF MOT ". :PRO
FIN
```

Utilisation du programme

```
POUR TRIANGLE :DONNEE
SI VIDEP :DONNEE <STOP>
EC :DONNEE
TRIANGLE SD :DONNEE
FIN

?PASAPAS "TRIANGLE
?TRIANGLE "IL
J'EXECUTE TRIANGLE
DONNEE EST IL
SI VIDEP :DONNEE <STOP>          Pressez ↵.

EC :DONNEE                       Pressez ↵.
IL
TRIANGLE SD :DONNEE              Pressez ↵.
J'EXECUTE TRIANGLE
DONNEE EST I
SI VIDEP :DONNEE <STOP>          Pressez ↵.

EC :DONNEE                       Pressez ↵.
I
TRIANGLE SD :DONNEE              Pressez ↵.
J'EXECUTE TRIANGLE
DONNEE EST
SI VIDEP :DONNEE <STOP>
?
```

Opérations logiques

Opérations logiques

Les **prédicats** sont des opérations qui retournent uniquement VRAI ou FAUX. La plupart de leurs noms finissent par P (prédicat).

Ce chapitre décrit les **opérations logiques** ET, NON et OU. Une opération logique est un prédicat qui retourne soit VRAI, soit FAUX.

Les données des opérations logiques sont généralement d'autres prédicats. Ils sont décrits dans les chapitres qui suivent.

Prédicat	**Chapitre**
AVANTP	7
BOUTONP	13
DEFINIEP	11
EGALP	7
FICHIERP	15
INFP	9
LISTEP	7
MEMBREP	7
MONTREP	5
MOTP	7
NOMBREP	7
NOMP	8
POINTP	5
PRIMITIVEP	11
SUPP	9
TOUCHEP	13
VIDEP	7
=	9

ET

ET *préd1 préd2* (opération)
(ET *préd1 préd2 préd3...)*

Cette opération retourne VRAI si toutes ses données sont VRAI ; sinon retourne FAUX.

Exemples :

Opération	**Résultat**
ET "VRAI "VRAI	VRAI
ET "VRAI "FAUX	FAUX
ET "FAUX "FAUX	FAUX
(ET "VRAI "VRAI "FAUX "FAUX)	FAUX
ET 5 7	erreur
ET CC = 1 FOND = 0 (à la mise en route de Logo)	VRAI

La procédure DECIMALP indique si sa donnée est un nombre décimal.

```
POUR DECIMALP :OBJ
RETOURNE ET NOMBREP :OBJ MEMBREP ". :OB!
J
FIN

?EC DECIMALP 17
FAUX
?EC DECIMALP 17.
VRAI
?EC DECIMALP "STOP
FAUX
```

La procédure suivante indique si la température est agréable (entre 10 et 30 degrés Celsius).

```
POUR CONFORT :TEMPERATURE
SI ET SUPP :TEMPERATURE 10 INFP :TEMPE!
RATURE 30<EC "AGREABLE><EC "DESAGREAB!
LE>
FIN

?CONFORT 18
AGREABLE
```

NON

NON *préd* (opération)

Cette opération retourne VRAI si *préd* est FAUX ou FAUX si *préd* est VRAI.

Exemples :

Opération	**Résultat**
NON EGALP "A "B	VRAI
NON EGALP "A "A	FAUX
NON "A = PREMIER "CHAT	VRAI
NON "A	erreur

Si MOTP n'était pas une primitive, elle pourrait être définie de la manière suivante :

```
POUR MOTP :OBJ
RETOURNE NON LISTEP :OBJ
FIN
```

La procédure suivante indique si sa donnée est un mot Logo autre qu'un nombre :

```
POUR VRAIMOTP :OBJ
RETOURNE ET MOTP :OBJ NON NOMBREP :OBJ
FIN

?EC VRAIMOTP CAP
FAUX
?EC VRAIMOTP POS
FAUX
?EC VRAIMOTP "KANGOUROU
VRAI
?EC VRAIMOTP "CRAYON
VRAI
```

OU

OU *préd1 préd2* (opération)
(OU *préd1 préd2 préd3...)*

Cette opération retourne FAUX si toutes ses données sont fausses ; sinon, retourne VRAI.

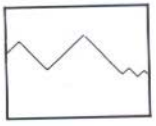
Exemples :

Opération	**Résultat**
OU "VRAI "VRAI	VRAI
OU "VRAI "FAUX	VRAI
OU "FAUX "FAUX	FAUX
(OU "FAUX "FAUX "FAUX "VRAI)	VRAI
OU 5 7	erreur

La procédure MONTAGNES dessine des montagnes :

```
POUR MONTAGNES
FCC 5
DR 45
AV 5
SOUSMONTAGNES
FIN

POUR SOUSMONTAGNES
AV 5 + HASARD 10
SI OU SUPP COORY 50 INFP COORY 0<FCAP !
180 - CAP >
SOUSMONTAGNES
FIN
```



Le monde extérieur

Le monde extérieur

Ce chapitre décrit les primitives qui permettent de communiquer avec l'ordinateur au moyen de périphériques tels le clavier, le téléviseur et les manettes de jeu. Ces primitives sont divisées en quatre groupes :

- celles pour les manettes ;
- celles pour la lecture ;
- celles pour l'écriture ;
- celle pour les effets sonores.

Utilisation des manettes

Cette section décrit les primitives BOUTONP et MANETTE qui communiquent des indications au moyen de manettes ou de boutons de commande.

BOUTONP

BOUTONP *nomanette* (opération)

Cette opération retourne VRAI si le bouton de la manette indiquée est enfoncé ; sinon, retourne FAUX. Le *nomanette* doit être 0, 1, 2 ou 3. ⌥ correspond au bouton 0 et ⌘ au bouton 1.

MANETTE

MANETTE *nomanette* (opération)

Cette opération retourne un nombre compris entre 0 et 255, représentant la rotation du cadran de la manette indiquée.

Exemple :

```
POUR MDESSINE
DR (MANETTE 0) / 25.6
AV (MANETTE 1) / 25.6
MDESSINE
FIN
```

Primitives de lecture

Cette section décrit les primitives que Logo utilise pour lire des indications à partir d'un périphérique, généralement le clavier, ou d'un fichier. Ces primitives sont :

LISCAR
LISCARS
LISLISTE
LISMOT
TOUCHEP

LISCAR, LISCARS, LISLISTE et LISMOT sont aussi employées pour la gestion des fichiers ; celle-ci est décrite aux chapitres 15 et 16.

Les opérations LISCAR, LISCARS, LISLISTE et LISMOT permettent à Logo de lire le texte qui a été tapé au clavier. Le prédicat TOUCHEP a aussi rapport au clavier et est surtout utilisé pour les jeux.

LISCAR

LISCAR (opération)

LISCAR (lis caractère) retourne le premier caractère lu provenant du clavier ou d'un fichier. Si aucun caractère n'est prêt à être lu à partir du clavier, LISCAR attend que l'usager en tape un.

Voir aussi TOUCHEP.

Si la position de fin de fichier est atteinte pendant la lecture, LISCAR retourne la liste vide. Remarquez que lorsque vous utilisez LISCAR au niveau du clavier, vous n'obtenez pas d'écho à ce que vous tapez.

Si LISCAR est utilisée au niveau du clavier, il est possible de mettre à 1 le bit de gauche du caractère ; vous devez enfoncer l'une ou l'autre des touches (⌥ ou ◆) en même temps que vous tapez le caractère. L'utilisation du bit de gauche ajoute 128 au caractère.

La procédure PROMENE permet à l'utilisateur d'exécuter certaines instructions en tapant une seule touche : A exécute AVANCE 5, D DROITE 10 (vous pouvez ajouter d'autres instructions à la liste). Il n'est pas nécessaire de taper ↵.

```
POUR PROMENE
INTERPRETE LISCAR
PROMENE
FIN

POUR INTERPRETE :CAR
SI :CAR = "A <AV 5>
SI :CAR = "D <DR 10>
SI :CAR = "S <RENVOIE "NIVEAUSUP>
FIN
```

LISCARS

LISCARS *entier* (opération)

LISCARS (lis caractères) retourne le nombre de caractères indiqués par *entier*, qu'ils soient lus au niveau du clavier ou dans un fichier. Si aucun caractère n'est prêt à être lu à partir du clavier, LISCARS attend que l'usager tape quelque chose.

Si la position de fin de fichier est atteinte avant que le nombre de caractères indiqués par *entier* ne soient lus, LISCARS retourne les caractères lus à ce stade. LISCARS retourne la liste vide si la fin du fichier est atteinte avant que LISCARS ne soit appelée.

Remarquez que LISCARS utilisée au niveau du clavier ne donne pas d'écho à ce que vous tapez.

Un retour de chariot est lu comme un caractère.

Si LISCARS est utilisée au niveau du clavier, il est possible de mettre à 1 le bit de gauche du caractère ; vous devez enfoncer l'une ou l'autre des touches (⌥ ou ◆) en même temps que vous tapez le caractère. L'utilisation du bit de gauche ajoute 128 au caractère.

Exemple :

```
?ECRIS LISCARS 4
```

Tapez les lettres suivantes :

```
ABC
```

Ne pressez pas ↵.

Rien ne se produit. Maintenant, tapez :

```
D
```

Voici ce qui apparaît à l'écran :

```
ABCD
```

LISLISTE

LISLISTE (LL) (opération)

LISLISTE lit une ligne dans le fichier ouvert pour la lecture et la retourne sous forme de liste. Si aucun fichier n'a été ouvert, LISLISTE lit à partir du clavier. Lorsque les lignes sont tapées au clavier, LISLISTE donne un écho à ce qui est tapé. La commande FLIS vous permet d'aller lire dans d'autres fichiers.

Si vous lisez dans un fichier et que la position de fin de fichier a été atteinte pendant la lecture, LISLISTE retourne le mot vide.

Exemples :

```
?EC COMPTE LISLISTE
JE SUIS PRET A PARTIR
5

POUR IDENTIFIE
EC <COMMENT VOUS APPELEZ!-VOUS?>
RELIE "USAGER LL
EC PH <BIENVENUE A LOGO,> :USAGER
FIN

?IDENTIFIE
COMMENT VOUS APPELEZ-VOUS?
SEBASTIEN
BIENVENUE A LOGO, SEBASTIEN
?IDENTIFIE
COMMENT VOUS APPELEZ-VOUS?
JEAN BAPTISTE MAURIN
BIENVENUE A LOGO, JEAN BAPTISTE MAURIN
```

LISMOT

LISMOT (LM) (opération)

Cette opération lit une ligne dans le fichier ouvert pour la lecture et la retourne sous forme de mot. Si aucun fichier n'a été ouvert, LISMOT lit à partir du clavier. Lorsque les mots sont tapés au clavier, LISMOT donne un écho à ce qui est tapé. Si vous pressez ↵ avant de taper un mot, LISMOT retourne le mot vide.

Voir aussi LISLISTE, LISCAR, LISCARS et FLIS.

Utilisée dans un fichier, LISMOT lit les caractères jusqu'à ce qu'elle atteigne un retour de chariot et retourne ceux-ci sous forme de mot. LISMOT poursuivra avec le premier caractère suivant le retour de chariot. Si vous lisez dans un fichier et que la position de fin de fichier est atteinte, LISMOT retourne la liste vide.

Exemples :

```
?MONTRE LISMOT
MONTREAL QUEBEC
MONTREAL QUEBEC

?EC COMPTE LISMOT
JE PEUX COMPTER LES MOTS
24
```

La procédure qui suit demande l'âge que vous avez et affiche l'âge que vous aurez l'année prochaine.

```
POUR AGE
EC <QUEL AGE AVEZ!-VOUS?>
EC MESSAGE LISMOT
FIN

POUR MESSAGE :AGE
RT (PH <L'ANNEE PROCHAINE, VOUS AUREZ> !
:AGE + 1 "ANS)
FIN

?AGE
QUEL AGE AVEZ-VOUS?
4
L'ANNEE PROCHAINE, VOUS AUREZ 5 ANS
?AGE
QUEL AGE AVEZ-VOUS?
35
L'ANNEE PROCHAINE, VOUS AUREZ 36 ANS
```

TOUCHEP

TOUCHEP (opération)

Cette opération retourne VRAI s'il y a au moins un caractère en attente pour lecture, c'est-à-dire si un caractère a été tapé au clavier et n'a pas encore été lu par LISCAR ou LISLISTE ; s'il n'y en a pas, TOUCHEP retourne FAUX.

Exemples :

```
POUR CONDUIRE
AV 2
SI TOUCHEP<TOURNE LISCAR>
CONDUIRE
FIN

POUR TOURNE :DIR
SI :DIR = "D<DR 10>
SI :DIR = "G<GA 10>
FIN
```

Primitives d'écriture

Cette section décrit les primitives que Logo utilise pour écrire des instructions à un endroit donné, par exemple, à l'écran. Ces primitives sont :

ECRIS
MONTRE
TAPE

ECRIS

ECRIS *obj1* (EC) (commande)
(ECRIS *obj1 obj2 obj3...)*

Cette commande affiche la ou les données à l'écran, suivies d'un retour de chariot, à moins que la destination n'ait été changée au moyen de FEC. Les crochets extérieurs des listes ne sont pas affichés.

Comparer avec MONTRE ET TAPE.

Exemples :

```
?EC "A
A
?EC "A EC <A B C>
A
A B C
?(EC "A <A B C>)
A A B C
?EC <>

?

POUR REECRIS :MESSAGE :NBFOIS
SI INFP :NBFOIS 1 <STOP>
EC :MESSAGE
EC <>
REECRIS :MESSAGE :NBFOIS - 1
FIN

?REECRIS <AUJOURD'HUI C'EST JEUDI> 4
AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

?
```

MONTRE

MONTRE *obj* (commande)

Cette commande écrit *obj* à l'écran suivi d'un retour de chariot, à moins que la destination n'ait été changée au moyen de FEC. Si *obj* est une liste, elle est affichée avec les crochets qui l'enferment.

Exemples :

```
?MONTRE "A
A
?MONTRE "A MONTRE <A B C>
A
<A B C>
```

TAPE

TAPE *obj1* (commande)
(TAPE *obj1 obj2...)*

Cette commande affiche à l'écran la ou les données sans retour de chariot, à moins que la destination n'ait été modifiée au moyen de la commande FEC. S'il s'agit d'une liste, les crochets extérieurs n'apparaissent pas. Comparer avec ECRIS et MONTRE.

Exemples :

```
?TAPE "A
A?TAPE "A TAPE <A B C>
AA B C?(TAPE "A <A B C>
AA B C?
```

La procédure INVITE tape un message suivi d'un espace :

```
POUR INVITE :MESSAGE
TAPE :MESSAGE
TAPE CAR 32
FIN

POUR DEPLACER
INVITE <DE COMBIEN DE PAS?>
AV PREMIER LL
DEPLACER
FIN

?DEPLACER
DE COMBIEN DE PAS? 50
DE COMBIEN DE PAS? 37
DE COMBIEN DE PAS? 2
DE COMBIEN DE PAS? 108
```

Utilisation de SON pour faire des effets sonores

SON

SON *fréq durée* (commande)

Cette commande génère une tonalité par l'entremise d'un haut-parleur. *Fréq* est évaluée en hertz (cycles par seconde). La note LA correspond à 440. *Durée* peut varier entre 0 et 65 535 Hz. Elle est mesurée en multiples de un soixantième de seconde.

Exemple :

```
POUR GLOUGLOU :FREQ
SI SUPP :FREQ 440 <STOP>
SON :FREQ 3
GLOUGLOU :FREQ + 5
SON :FREQ 3
FIN
```

GLOUGLOU produit des sons passant du grave à l'aigu et de l'aigu au grave.

Le tableau 13-1 fournit les fréquences pour environ 7 octaves.

Tableau 13-1. *Fréquences utilisées avec SON*

Note	**Fréquence par octave**							
SI	62	123	247	494	988	1973	3946	
LA #	58	117	233	466	932	1864	3743	
LA	55	110	220	440	881	1761	3510	
SOL #	52	104	208	415	830	1663	3327	
SOL	49	98	196	392	784	1566	3142	
FA #	46	92	185	370	740	1480	2959	
FA	44	87	175	349	698	1398	2797	
MI	41	82	165	330	659	1319	2637	
RE #	39	78	156	311	622	1244	2495	4990
RE	37	73	147	294	587	1176	2346	4713
DO #	35	69	139	277	554	1109	2213	4426
DO	33	65	131	262	523	1047	2095	4172

do diapason

Gestion de l'espace de travail

Gestion de l'espace de travail

Ce chapitre explique comment gérer l'espace de travail de votre ordinateur Apple. L'**espace de travail** est une partie de la mémoire de l'Apple qui renferme les variables, les procédures et les propriétés que vous y avez introduites. Cela n'inclut pas les primitives.

Logo fournit des primitives :

- pour vérifier la capacité de l'espace de travail et libérer des nœuds ;
- pour examiner le contenu de l'espace de travail ;
- pour effacer les variables et les procédures contenues dans votre espace de travail ;
- pour nettoyer et organiser l'espace de travail.

Voir les chapitres 15 et 16 pour plus d'indications sur les fichiers.

L'espace de travail est un site temporaire de stockage. Vos procédures, variables et propriétés sont effacées lorsque vous éteignez votre ordinateur. Si vous voulez les conserver pour un usage ultérieur, vous devez les emmagasiner sur une disquette sous la forme de fichiers.

Les procédures et les variables peuvent être enterrées. Les primitives EFTOUT, EFPS, IMTOUT, IMPS, IMTS et SAUVE n'ont aucun effet sur les procédures et les variables lorsque ces dernières sont enterrées : ces procédures et ces variables sont toujours présentes dans l'espace de travail. Il vous est ainsi possible d'exécuter, d'éditer, d'imprimer ou d'effacer une procédure enterrée dans la mesure où vous spécifiez son nom.

Les caractéristiques des primitives qui comportent le radical ENTERRE ou DETERRE permettent d'organiser votre espace de travail. Vous pouvez les employer pour sauvegarder des procédures dans différents fichiers ou pour transformer vos procédures en primitives. Par exemple, les procédures décrites à l'annexe B, intitulée "Instruments utiles", peuvent être enterrées dans l'espace de travail.

Voici comment vous pouvez organiser votre espace de travail :

```
?IMTS
POUR PLURIEL :NOMS :VERBES
POUR CHOISIR :OBJET
POUR SUPERPLURIEL
POUR POLY :COTE :ANGLE
POUR POLYSPI :COTE :ANGLE :AUG
POUR CA :COTE
POUR TRIANGLE :MOT

?IMNS
RELIE "NOMS <ORDINATEURS MAISONS LITS CHAISE!
S LIVRES>
RELIE "VERBES <TRAITE ASSOIE ALLONGE <B!
ERCE>>
RELIE "DEPART CAP
```

Vous pouvez regrouper vos procédures et vos variables en leur donnant un nom.

```
?RELIE "LANGAGE <PLURIEL CHOISIR SUPERP!
LURIEL>
?RELIE "DISCOURS <NOMS VERBES>
```

Utilisez les primitives ENTERRE et DETERRE pour sauvegarder les procédures et les variables dans un fichier.

```
?ENTERRETOUT
?DETERRE :LANGAGE
?DETERRENOM :DISCOURS
?SAUVE "LANGAGE
```

Economie de l'espace

Les primitives décrites dans cette section vous permettent de vérifier de combien d'espace vous disposez (NŒUDS) et de libérer le plus de nœuds possible (RECYCLE).

NŒUDS

NŒUDS (opération)

Voir l'annexe D intitulée "Espace mémoire".

Cette opération retourne le nombre de nœuds libres. Ceci vous donne une idée de l'espace disponible dans votre espace de travail pour les procédures, les variables et les propriétés, et pour faire exécuter vos procédures. NŒUDS est particulièrement utile si elle est exécutée immédiatement après RECYCLE.

RECYCLE

RECYCLE (commande)

Voir NŒUDS ainsi que l'annexe D intitulée "Espace mémoire".

La commande RECYCLE effectue un *nettoyage* pour libérer le plus de nœuds possible. Si vous n'employez pas RECYCLE, un nettoyage automatique s'effectue lorsque cela est nécessaire ; toutefois, chacun prend au moins une seconde. L'utilisation de RECYCLE avant une activité où il est nécessaire de tenir compte du facteur temps évite qu'un nettoyage automatique ne se fasse à un moment inapproprié.

Imprimer le contenu de l'espace de travail

Cette section décrit les primitives utilisées pour imprimer le contenu de l'espace de travail ; elles apparaissent dans l'ordre suivant :

IM
IMN
IMNS
IMPS
IMT
IMTOUT
IMTS

IM

IM *nom(liste)* (commande)

La commande IM (imprime) affiche la définition de chacune des procédures indiquées par *nom(liste)*.

Exemples :

```
?IM "LONG
POUR LONG :OBJ
SI VIDEP :OBJ <RT 0><RT 1 + LONG SP :0!
BJ>
FIN
?IM <LONG SALUER>
POUR LONG :OBJ
SI VIDEP :OBJ <RT 0><RT 1 + LONG SP :0!
BJ>
FIN

POUR SALUER
EC <BONJOUR, COMMENT ÇA VA?>
FIN
```

IMN

IMN *nom(liste)* (commande)

La commande IMN (imprime nom) affiche le nom et la valeur de la variable ou des variables indiquées par *nom(liste)*.

Exemples :

```
?IMN "LONG
RELIE "LONG 3.98
?IMN :DISCOURS
RELIE "NOMS <ORDINATEURS MAISONS LITS CHAISE!
S LIVRES>
RELIE "VERBES <TRAITE ASSOIE ALLONGE <B!
ERCE>>
```

IMNS

IMNS (commande)

La commande IMNS (imprime noms) affiche le nom et la valeur de toutes les variables contenues dans l'espace de travail.

Exemple :

```
?IMNS
RELIE "F 3
RELIE "LISTE <A B C>
```

IMPS

IMPS (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande IMPS (imprime procédures) affiche la définition de toutes les procédures contenues dans l'espace de travail.

Exemple :

```
?IMPS
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN

POUR SPI :COTE :ANGLE :AUG
AV :COTE DR :ANGLE
SPI :COTE + :AUG :ANGLE :AUG
FIN
```

IMT

IMT *nom(liste)* (commande)

La commande IMT (imprime titre) affiche la ligne titre de la ou des procédures indiquées par *nom(liste)*.

Exemple :

Vous pouvez regrouper quelques procédures en leur donnant un nom de variable.

```
?RELIE "LANGAGE <PLURIEL CHOISIR>
```

Utilisez IMT pour que s'affichent les titres contenus dans la variable LANGAGE.

```
?IMT :LANGAGE
POUR PLURIEL :NOMS :VERBES
POUR CHOISIR :OBJET
```

IMTOUT

IMTOUT (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande IMTOUT (imprime tout) affiche la définition de chaque procédure et la valeur de chaque variable contenues dans l'espace de travail.

Exemple :

```
?IMTOUT
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN

POUR LONG :OBJ
SI VIDEP :OBJ <RT 0> <RT 1 + LONG SP :0!
BJ>
FIN

POUR SALUER
EC <BONJOUR, COMMENT ÇA VA? >
FIN

POUR SPI :COTE :ANGLE :AUG
AV :COTE
DR :ANGLE
SPI :COTE + :AUG :ANGLE :AUG
FIN

RELIE "ANIMAL "ELEPHANT
RELIE "LONG 3.98
RELIE "PRENOM "MARIE
```

IMTS

IMTS (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande IMTS (imprime titres) affiche la ligne titre de chacune des procédures contenues dans l'espace de travail.

Exemples :

```
?IMTS
POUR POLY :COTE :ANGLE
POUR LONG :OBJ
POUR SALUER
POUR SPI :COTE :ANGLE :AUG
```

Effacer le contenu de l'espace de travail

Cette section décrit, dans l'ordre suivant, les primitives qui permettent d'effacer des informations contenues dans l'espace de travail.

EFFACE, EF
EFN
EFNS
EFPS
EFTOUT

EFFACE

EFFACE *nom(liste)* (EF) (commande)

La commande EFFACE efface de l'espace de travail la ou les procédures indiquées par *nom(liste)*.

Exemples :

`EFFACE "TRIANGLE` efface la procédure TRIANGLE.
`EFFACE <TRIANGLE CARRE>` efface les procédures TRIANGLE et CARRE.

EFN

EFN *nom(liste)* (commande)

Voir les exemples utilisant les variables NOMS et VERBES au début de ce chapitre.

La commande EFN (efface nom) efface de l'espace de travail la ou les variables indiquées par *nom(liste)*.

```
EFN "LONG
```
efface la variable LONG.

```
EFN :DISCOURS
```
efface les variables NOMS et VERBES.

EFNS

EFNS (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande EFNS (efface noms) efface toutes les variables contenues dans l'espace de travail.

EFPS

EFPS (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande EFPS (efface procédures) efface toutes les procédures contenues dans l'espace de travail.

EFTOUT

EFTOUT (commande)

Voir les exceptions à cette règle sous la rubrique ENTERRE.

La commande EFTOUT (efface tout) efface toutes les procédures, les variables et les propriétés contenues dans l'espace de travail.

Manipuler et organiser l'espace de travail

Cette section décrit, dans l'ordre suivant, les primitives qui permettent d'organiser l'espace de travail.

DETERRE
DETERRENOM
DETERRETOUT
ENTERRE
ENTERRENOM
ENTERRETOUT

DETERRE

DETERRE *nom(liste)* (commande)

Voir aussi ENTERRE.

Cette commande déterre les procédures indiquées par *nom(liste)*.

DETERRENOM

DETERRENOM *nom(liste)* (commande)

Cette commande déterre la ou les variables indiquées par *nom(liste)*.

Exemple :

```
?IMNS
?_
```

Il n'y a aucune variable visible dans l'espace de travail.

```
?DETERRENOM <LONG NOMS>
?IMNS
RELIE "LONG 3.98
RELIE "NOMS <ORDINATEURS MAISONS LITS C!
HAISES LIVRES>
```

DETERRETOUT

DETERRETOUT (commande)

Cette commande déterre toutes les procédures et les variables qui sont enterrées dans l'espace de travail.

Exemple :

```
?IMTS
?IMNS
```

Aucune variable ou procédure n'est affichée.

```
?DETERRETOUT
?IMTS
POUR POLY :COTE :ANGLE
POUR LONG :OBJ
POUR SPI :COTE :ANGLE :AUG
?IMNS
RELIE "ANIMAL "ELEPHANT
RELIE "LONG 3.98
RELIE "PRENOM "MARIE
```

Lorsque la primitive DETERRETOUT est exécutée, les procédures et les variables deviennent visibles.

ENTERRE

ENTERRE *nom(liste)* (commande)

Cette primitive enterre toutes les procédures indiquées par *nom(liste)*. Certaines commandes telles EFTOUT, EFNS, IMTOUT, IMPS, IMTS et SAUVE agissent sur tout le contenu de l'espace de travail excepté sur les procédures enterrées.

Exemple :

SAUVE "BON sauvegarde dans le fichier BON tout le contenu de l'espace de travail à l'exception des procédures et des variables déjà enterrées.

ENTERRENOM

ENTERRENOM *nom(liste)* (commande)

Voir comment déterrer les variables sous la rubrique DETERRENOM.

Cette commande enterre la ou les variables indiquées par *nom(liste)*.

Exemple :

```
?IMNS
RELIE "ANIMAL "ELEPHANT
RELIE "LONG 3.98
RELIE "PRENOM "MARIE
?ENTERRENOM "PRENOM
?IMNS
RELIE "ANIMAL "ELEPHANT
RELIE "LONG 3.98
```

ENTERRETOUT

ENTERRETOUT (commande)

Voir DETERRETOUT pour savoir comment déterrer le contenu de l'espace de travail.

Cette commande enterre toutes les procédures et les variables contenues dans l'espace de travail.

Exemple :

```
?IMTS
POUR POLY :COTE :ANGLE
POUR LONG :OBJ
POUR SALUER
POUR SPI :COTE :ANGLE :AUG
?IMNS
RELIE "ANIMAL "ELEPHANT
RELIE "LONG 3.98
RELIE "PRENOM "MARIE
?ENTERRETOUT
?IMTS

?IMNS

?_
```

Lorsque la primitive ENTERRETOUT est exécutée, les procédures et les variables deviennent invisibles.

Gestion des fichiers

Gestion des fichiers

La description des primitives utilisées pour des types particuliers de fichiers se trouve au chapitre 16.

Le système de fichiers Logo vous permet d'utiliser des fichiers programmes, des fichiers dessins, des fichiers de données ainsi que des fichiers copies. Dans ce chapitre, vous trouverez des indications d'ordre général sur le système de fichiers Logo ainsi que sur les primitives servant à la gestion de tous les types de fichiers Logo.

Ce chapitre est divisé en deux sections :

- l'une vous donne des renseignements d'ordre général sur le système de fichiers ainsi que des indications sur la terminologie et sur les règles à employer ;
- l'autre fournit les primitives servant à la gestion des fichiers.

Quelques indications sur le système de fichiers Logo

Cette section vous donne les principes de base du système de fichiers Logo et introduit l'exemple utilisé tout au long de ce chapitre qui illustre les caractéristiques de la gestion de fichiers.

Qu'est-ce qu'un fichier ?

Un **fichier** consiste en un ensemble de renseignements mis en réserve sur disquette. Logo vous permet de créer différents types de fichiers sur une disquette selon les caractéristiques de l'information qui y est stockée.

Il existe quatre types de fichiers Logo : les fichiers programmes, les fichiers dessins, les fichiers copies et les fichiers de données. Un **fichier programmes** contient les procédures que vous désirez conserver et utiliser ultérieurement. Un **fichier dessins** peut contenir un dessin que vous avez créé. Un **fichier copies** contient un double du texte affiché à l'écran. Un **fichier de données** renferme des renseignements que vous désirez mettre à jour tels des adresses et des numéros de téléphone.

Les fichiers sont tous organisés de la même façon sur la disquette malgré que le contenu de chacun puisse être différent. La section suivante décrit comment ProDOS gère les fichiers. ProDOS est le système d'exploitation de disque à l'aide duquel Logo fonctionne.

Formatage d'une disquette et dénominations

Référez-vous au chapitre 4 du manuel d'introduction pour plus de détails sur le formatage.

Chaque disquette doit être formatée. Le formatage prépare la disquette que vous utiliserez. Ainsi :

- La surface de la disquette est divisée en zones uniformes appelées **blocs** où ProDOS stocke de l'information.
- Cela permet d'attribuer une dénomination à la disquette.
- Le formatage fournit un index du contenu de la disquette ainsi que d'autres indications dont ProDOS a besoin pour repérer l'emplacement des fichiers.

Vous devez effectuer le formatage de la ou des disquettes avant d'y stocker quoi que ce soit.

La disquette formatée sur laquelle se trouvent les fichiers que vous désirez conserver porte un nom, soit la **dénomination.** En voici quelques exemples :

Dénomination	**Utilisation**
/LOGO/	La disquette utilisée pour mettre Logo en route.
/MESPROJETS/	La disquette qui contient votre travail.
/EXEMPLES.LOGO/	Une disquette utilisée à titre d'exemple tout au long de ce chapitre.

Vous devez utiliser la dénomination pour indiquer à Logo où trouver le fichier que vous désirez et où stocker le fichier que vous voulez sauvegarder.

Contenu de la disquette

Les fichiers peuvent être sauvegardés sur une disquette de différentes façons. La commande IMINDEX vous donne une liste des fichiers qui se trouvent sur la disquette. Cette liste de noms accompagnée de la longueur de chacun des fichiers est appelée un **index.** Avant d'ouvrir un fichier, ProDOS vérifie l'index de la disquette pour être en mesure de trouver le fichier sur celle-ci.

La disquette portant la dénomination /MESPROJETS/ comporte l'index suivant :

```
?IMINDEX
/MESPROJETS/
   DESSINS1          10
   POLYGONES         15
   SPIRALES          10
   JEU               10
   LISTE.TEL         20
   ADRESSES          10
BLOCS LIBRES: 205 BLOCS UTILISES: 75
?
```

Dans l'exemple précédent, les fichiers sont sauvegardés sous l'index principal puisqu'aucun sous-index n'a été créé. Lorsqu'un grand nombre de fichiers se trouvent sur la disquette, cette façon de les stocker peut devenir peu commode.

ProDOS vous permet, au moyen d'un système de sous-index, de classer vos fichiers sur la disquette selon vos besoins. Ainsi EXEMPLES.LOGO utilise ce système de sous-index. Les **sous-index** sont des fichiers contenant des listes d'autres fichiers.

Les sous-index sont très utiles pour bien structurer le contenu de la disquette. Par exemple, la disquette portant le nom / EXEMPLES.LOGO/ comporte trois sous-index : le premier (PROGRAMMES) contient des programmes Logo, le second (DESSINS), des dessins faits à l'aide du graphique Tortue, et le troisième (DONNEES), les données pour vos programmes.

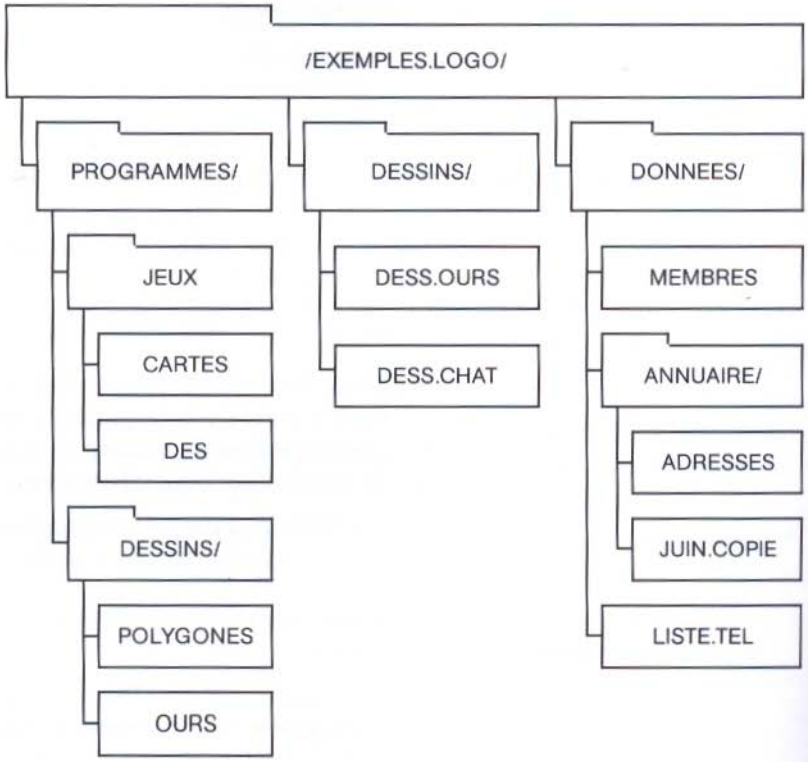
La disquette dont la dénomination est /EXEMPLES.LOGO/ comporte l'index suivant :

```
?IMINDEX
/EXEMPLES.LOGO/
   PROGRAMMES/
   DESSINS/
   DONNEES/
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
```

Remarquez que les noms de fichiers apparaissant ci-dessus se terminent par une barre oblique (/) ; cette dernière indique que ces noms sont des sous-index.

Le tableau 15-1 offre un schéma de l'index de la disquette /EXEMPLES.LOGO/. La structure illustrée dans ce tableau est utilisée dans la plupart des exemples apparaissant dans ce chapitre ainsi qu'au chapitre 16.

Tableau 15-1. *Schéma des fichiers et des sous-index sur une disquette*



Remarquez que les sous-index /EXEMPLES.LOGO /PROGRAMMES/ et /EXEMPLES.LOGO/DONNEES/ contiennent à leur tour des sous-index.

La description de CREEINDEX et de EFFICHIER apparaît plus loin dans ce chapitre.

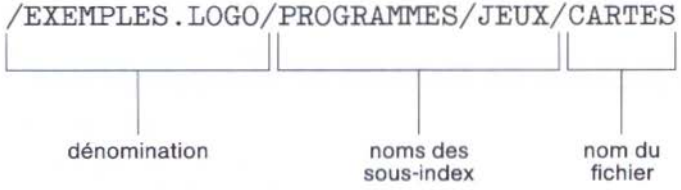
Vous devez utiliser la commande CREEINDEX pour créer un sous- index ; si vous désirez l'effacer, vous devez commander EFFICHIER ou EFF.

Accès au fichier

ProDOS vérifie à chacun des niveaux de l'index de la disquette si un fichier s'y trouve.

Ainsi, pour avoir accès au fichier CARTES sur la disquette /EXEMPLES.LOGO/, Logo doit d'abord avoir *accès* à /EXEMPLES.LOGO/ puis à PROGRAMMES/ ainsi qu'à JEUX/ et finalement à CARTES.

Le nom complet pour accéder au fichier inclut les index et sous-index.



Note : Un nom de fichier a un maximum de 15 caractères et doit commencer par une lettre. Le nom peut comporter des lettres de l'alphabet et des chiffres compris entre 0 et 9 ainsi que des points (.).

Le **préfixe** est composé du nom de la disquette suivi du nom des sous-index. Le préfixe est automatiquement placé avant le nom du fichier ; ce dernier ne doit pas être précédé de la barre oblique (/) une fois qu'on a fixé le préfixe.

Il existe deux façons d'avoir accès au fichier CARTES :

- en fournissant le préfixe suivi du nom des index, sous-index et du fichier ;

```
RAMENE "/EXEMPLES.LOGO/PROGRAMMES/JEUX!
/CARTES
```

- en fixant le préfixe du sous-index qui contient CARTES puis en n'utilisant que le nom du fichier, par exemple :

```
FPREFIXE "/EXEMPLES.LOGO/PROGRAMMES/JE!
UX
RAMENE "CARTES
```

Si vous désirez avoir accès à plusieurs fichiers d'un même sous-index, il est plus commode d'employer cette deuxième méthode.

La commande IMINDEX donne une liste du contenu de l'index et fournit le préfixe à chaque fois qu'elle est utilisée.

```
?IMINDEX
/EXEMPLES.LOGO/PROGRAMMES/JEUX        Voici le préfixe.
   CARTES              12
   DES                  5
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
```

Le système de fichiers et ses primitives

Le reste de ce chapitre donne une description de chacune des primitives servant à la gestion des fichiers, soit créer un sous-index, vérifier quelle disquette est utilisée, etc. Quelle que soit la nature de l'information contenue dans les fichiers, ces primitives peuvent être utilisées avec tous les types de fichiers. Ce sont les suivantes :

CREEINDEX	IMINDEX
EDFICHIER, EDF	NOMSDISQUES, ND
EFFICHIER, EFF	PREFIXE
FICHIERP	RAMENEAIDE
FPREFIXE	RENOMME
IMFICHIER, IMF	

CREEINDEX

CREEINDEX *nomfichier* (commande)

Cette commande crée le sous-index *nomfichier*. Le dernier nom de fichier apparaissant dans *nomfichier* devient donc le sous-index ; les noms qui précèdent ce nom de fichier indiquent où le nouveau sous-index doit être placé.

Exemple :

```
CREEINDEX "/EXEMPLES.LOGO/PROGRAMMES/O!
UTILS
```

crée le sous-index OUTILS faisant partie du sous-index PROGRAMMES. Si le préfixe est déjà fixé à /EXEMPLES.LOGO/PROGRAMMES/,

```
?CREEINDEX "OUTILS
```

a le même effet.

EDFICHIER

EDFICHIER *nomfichier* (EDF) (commande)

La commande EDFICHIER ramène le fichier *nomfichier* dans le tampon d'édition et sauvegarde les modifications apportées au contenu sous le même nom de fichier. Ce que le fichier contenait antérieurement est perdu.

Des indications supplémentaires sur l'utilisation de l'éditeur apparaissent au chapitre 4.

Vous pouvez utiliser EDFICHIER avec n'importe quel fichier, que ce dernier existe ou non. Logo crée un fichier lorsque vous sauvegardez le contenu du tampon d'édition.

Le tampon d'édition peut contenir un maximum de 6144 caractères. Si le fichier que vous désirez éditer en contient plus, Logo affiche un message ; vous ne pouvez donc pas éditer le fichier.

EFFICHIER

EFFICHIER *nomfichier* (EFF) (commande)

La commande EFFICHIER efface de la disquette le fichier *nomfichier.* Si l'on fournit seulement le nom de fichier comme donnée, ce fichier doit se trouver dans l'index auquel vous avez eu accès. Il y a erreur si le fichier n'existe pas.

Exemple :

```
?EFFICHIER "/EXEMPLES.LOGO/PROGRAMMES/D!
ESSINS/OURS
```

efface le fichier OURS du sous-index DESSINS qui, à son tour, se trouve dans le sous-index PROGRAMMES.

La commande EFFICHIER permet aussi d'effacer des sous-index seulement si ces derniers ne contiennent pas de fichiers. Il y a erreur si vous tentez d'effacer un sous-index dans lequel se trouvent des fichiers.

FICHIERP

FICHIERP *nomfichier* (opération)

L'opération FICHIERP retourne VRAI si le fichier indiqué par *nomfichier* se trouve sur la disquette à condition que le préfixe soit bien précisé ; sinon, FICHIERP retourne FAUX. Il y a erreur si vous utilisez FICHIERP et un nombre indiquant un périphérique.

Exemples :

```
?EC FICHIERP "/EXEMPLES.LOGO/PROGRAMMES!
/HANOI
FAUX
```

Le fichier HANOI n'existe pas.

La procédure REMPL.FICHIER vous permet de remplacer un ancien fichier par un nouveau lorsque vous mettez l'information en réserve sur une disquette.

```
POUR REMPL.FICHIER :FICHIER
SI FICHIERP :FICHIER<EFF :FICHIER>
SAUVE :FICHIER
FIN
```

FPREFIXE

FPREFIXE *préfixe* (commande)

La commande FPREFIXE demande à Logo d'utiliser le préfixe indiqué par *préfixe*. Cette commande vous permet d'avoir accès à un fichier se trouvant dans le sous-index portant le nom *préfixe* sans avoir à le taper en entier, c'est-à-dire en fournissant le nom de la disquette ainsi que le nom du sous-index. L'affichage de l'index, au niveau de la commande IMINDEX, s'en trouve légèrement modifié.

Exemples :

```
?FPREFIXE "/EXEMPLES.LOGO/PROGRAMMES
?IMINDEX
/EXEMPLES.LOGO/PROGRAMMES
   JEUX/
   DESSINS/
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
```

Vous pouvez maintenant avoir accès aux fichiers ou aux sous-index JEUX et DESSINS contenus dans le sous-index PROGRAMMES en ne fournissant que les noms de fichiers.

Pour avoir accès aux fichiers contenus dans l'index principal, tapez :

```
?FPREFIXE "/EXEMPLES.LOGO
?IMINDEX
/EXEMPLES.LOGO/
   PROGRAMMES/
   DESSINS/
   DONNEES/
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
```

IMFICHIER

IMFICHIER *nomfichier* (IMF) (commande)

La commande IMFICHIER (imprime fichier) affiche à l'écran le contenu du fichier indiqué par *nomfichier*. Il y a erreur si vous tentez d'utiliser IMFICHIER pour un fichier déjà ouvert.

Exemple :

La procédure suivante vous permet de faire une copie d'un fichier.

```
POUR COPIEF :DE :A
COPIE :DE
IMFICHIER :A
NONCOPIE
FIN
```

Pour obtenir une copie du fichier POLYGONES dans le fichier FIGURES, tapez :

```
?COPIEF "POLYGONES "FIGURES
```

IMINDEX

IMINDEX (commande)

La commande IMINDEX affiche à l'écran les noms des fichiers se trouvant dans l'index ainsi que le nombre de blocs utilisés par chacun. Il s'agit de l'index qui a été désigné à l'aide du préfixe.

Exemple :

```
?IMINDEX
/EXEMPLES.LOGO/                    (préfixe)
PROGRAMMES/                        (sous-index)
DESSINS/                           (sous-index)
DONNEES/                           (sous-index)
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
?FPREFIXE "PROGRAMMES              (fixe le préfixe)
?IMINDEX
/EXEMPLES.LOGO/PROGRAMMES/         (préfixe)
   JEUX/                           (sous-index)
   DESSINS/                        (sous-index)
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?
```

Voici la façon de voir ce qui se trouve dans le sous-index DESSINS :

```
?FPREFIXE "DESSINS
?IMINDEX
/EXEMPLES.LOGO/PROGRAMMES/DESSINS                    (préfixe)
   POLYGONES     2                                   (nom de fichier)
   OURS          3                                   (nom de fichier)
BLOCS LIBRES: 138 BLOCS UTILISES: 142
?FPREFIXE "/EXEMPLES.LOGO/DONNEES/ANNUAIRE           (fixe le préfixe du nouveau
                                                      sous-index)
?IMINDEX
/EXEMPLES.LOGO/DONNEES/ANNUAIRE/
   ADRESSES      10                                  (nom de fichier)
   LISTE.TEL     15                                  (nom de fichier)
```

NOMSDISQUES

NOMSDISQUES (ND) (opération)

L'opération NOMSDISQUES, ou sa forme abrégée ND, retourne la dénomination (ou le nom) de la ou des disquettes utilisées. Par exemple, si vous disposez de deux unités de disquette et qu'une disquette se trouve dans chacune d'elles, NOMSDISQUES retourne la dénomination de chacune.

Exemple :

```
?MONTRE NOMSDISQUES
</EXEMPLES.LOGO/>
```

Vous pouvez utiliser NOMSDISQUES si vous ne vous rappelez pas du nom que vous avez attribué à une disquette. Insérez celle-ci dans l'unité de disquette et tapez EC NOMSDISQUES. Logo affichera le nom de la disquette.

PREFIXE

PREFIXE (opération)

L'opération PREFIXE retourne le préfixe utilisé. Vous devez employer FPREFIXE pour attribuer le préfixe.

Exemple :

```
?EC PREFIXE
/EXEMPLES.LOGO/
?FPREFIXE "DESSINS
?EC PREFIXE
/EXEMPLES.LOGO/DESSINS/
```

RAMENEAIDE

RAMENEAIDE *nomfichier* (commande)

La primitive RAMENEAIDE ramène le fichier *nomfichier* dans la mémoire où se trouve le principal écran d'aide. Cette primitive vous permet d'écrire des programmes Logo qui peuvent aider l'utilisateur.

Pendant que Logo lit des données provenant du clavier, l'utilisateur peut presser [ó] - [?] à n'importe quel moment et l'écran d'aide est affiché.

Le fichier que vous ramenez doit contenir moins de 1023 caractères ; cela inclut les espaces et les retours de chariot. Vous pouvez utiliser la commande EDFICHIER pour écrire le texte de votre propre écran d'aide et l'opération LONGUEURF pour vérifier combien de caractères le fichier contient.

Exemple :

```
?RAMENEAIDE "/EXEMPLES.LOGO/NOUV.AIDE
```

RENOMME

RENOMME *nomfichier nouvnomfichier* (commande)

La commande RENOMME permet de trouver le fichier *nomfichier* sur la disquette et de transformer son nom en *nouvnomfichier*. Le contenu du fichier reste le même qu'auparavant ; *nouvnomfichier* doit désigner un fichier se trouvant dans le même index que *nomfichier*.

Exemple :

```
?RENOMME "/EXEMPLES.LOGO/DONNEES/ADRESS!
ES "/EXEMPLES.LOGO/DONNEES/ANC.ADRESSES
```

transforme le fichier ADRESSES en ANC.ADRESSES.

Gestion des différents types de fichiers

Gestion des différents types de fichiers

La gestion des fichiers est expliquée au chapitre 15.

Ce chapitre donne une description des différents types de fichiers Logo ; il est divisé en cinq sections :

- les primitives utilisées pour les fichiers programmes ;
- les primitives utilisées pour les fichiers dessins ;
- les primitives utilisées pour les fichiers copies ;
- les primitives utilisées pour les fichiers de données ;
- un projet utilisant un fichier de données.

La disquette portant le nom EXEMPLES.LOGO, employée au chapitre précédent, sera aussi utilisée ici. Référez-vous au schéma (Tableau 15-1) illustrant la structure de l'index de cette disquette lorsque des exemples seront fournis.

Logo peut lire de l'information provenant de trois sources : le clavier, les fichiers contenus sur une disquette ainsi que les périphériques reliés à l'ordinateur. Lorsque Logo est mis en route, celui-ci lit les données à partir du clavier.

Parallèlement, Logo écrit des données dans des fichiers contenus sur disquette ainsi qu'à l'écran et permet que ces données soient envoyées à des périphériques. Lorsque Logo est mis en route, celui-ci écrit l'information à l'écran.

Note : Un **périphérique** fait référence au matériel informatique relié à l'ordinateur au moyen d'un logement (dans le cas de l'Apple IIe) ou d'une prise (pour l'Apple IIc). Le contenu de la disquette est organisé en fichiers ; il est important de noter que Logo traite le clavier ou les périphériques, tels l'imprimante ainsi que l'écran, comme s'il s'agissait de fichiers.

Certaines primitives de gestion de fichiers peuvent être utilisées à la fois avec des fichiers se trouvant sur une disquette et des périphériques tels l'imprimante. Ainsi, dans ce chapitre, la donnée *fichier* peut donc désigner un fichier d'une disquette ou un périphérique. On a accès au périphérique par l'intermédiaire du numéro de logement ou d'une prise auquel le périphérique est connecté. Dans la plupart des cas, le périphérique est une imprimante. Vous aurez accès à l'imprimante connectée au logement 1 ou à la prise 1 de l'ordinateur en fournissant le nombre 1 comme donnée pour *fichier.*

Fichiers programmes

Cette section donne une description des primitives servant à sauvegarder et à ramener des fichiers contenant des programmes Logo ; ce sont les suivantes :

RAMENE
SAUVE
SAUVEL

RAMENE

RAMENE *nomfichier* (commande)

La commande RAMENE charge le contenu du fichier indiqué par *nomfichier* dans l'espace de travail, comme si vous aviez tapé le contenu de ce fichier au niveau supérieur. Il y a erreur si le fichier que vous voulez ramener n'existe pas, ou si vous essayez RAMENE suivie d'un nombre désignant un périphérique.

La lecture du contenu du fichier terminée, Logo recherche la variable DEPART. Si cette dernière existe, Logo exécute son contenu.

Exemples :

```
?FPREFIXE "/PROGRAMMES/DESSINS
?RAMENE "OURS
```

Logo ramène le contenu du fichier OURS dans l'espace de travail.

SAUVE

SAUVE *nomfichier* (commande)

La commande SAUVE crée un fichier et y sauvegarde toutes les procédures et toutes les variables déterrées, et toutes les propriétés contenues dans l'espace de travail. Il y a erreur si le fichier que vous désignez existe déjà. Dans ce cas, vous devez d'abord effacer ce fichier à l'aide de EFFICHIER ou lui donner un autre nom au moyen de RENOMME. Il y a erreur si vous commandez SAUVE suivie d'un nombre désignant un périphérique.

Exemple :

```
?SAUVE "/PROGRAMMES/DESSINS/MASQUES
```

sauvegarde le contenu de l'espace de travail dans le fichier MASQUES.

SAUVEL

SAUVEL *nom(liste) nomfichier* (commande)

La commande SAUVEL (sauve liste) sauvegarde dans le fichier *nomfichier* la ou les procédures *nom(liste)*, ainsi que les variables déterrées et les propriétés contenues dans l'espace de travail. Cette commande sert à sauvegarder une partie de votre espace de travail sur disquette. Il y a erreur si vous commandez SAUVEL suivie d'un nombre désignant un périphérique. Comparer avec SAUVE.

Exemple :

```
?IMTS
POUR TRI :OBJET
POUR POLY :COTE :ANGLE
POUR SPI :COTE :ANGLE :AUG
POUR BIENVENUE :NOM

?SAUVEL <POLY SPI > "/EXEMPLES.LOGO/PROG!
RAMMES/DESSINS/POLYGONES
```

Fichiers dessins

Les primitives décrites dans cette section servent à ramener, à sauvegarder ou à imprimer des dessins :

IMIMAGE
RAMENEIMAGE
SAUVEIMAGE

IMIMAGE

IMIMAGE *entier* (commande)

La commande IMIMAGE imprime sur papier le contenu de l'écran graphique ; l'imprimante doit être reliée à la prise ou au logement indiqué par *entier*. L'imprimante Apple Imagewriter vous permet de faire imprimer des dessins. Pour obtenir de bons résultats si vous utilisez un autre type d'imprimante, vous devez d'abord taper les instructions suivantes :

```
.DEPOSE 770 1
.FECHELLE 1
```

puis,

```
IMIMAGE 1
```

Si vous utilisez l'imprimante Apple Imagewriter, vous n'avez qu'à taper :

```
IMIMAGE 1
```

RAMENEIMAGE

RAMENEIMAGE *nomfichier* (commande)

La commande RAMENEIMAGE ramène à l'écran le dessin contenu dans le fichier indiqué par *nomfichier*. Si le contenu du fichier n'est pas un dessin, Logo ne pourra répondre à la demande de ramener un dessin.

Exemple :

```
?RAMENEIMAGE "/EXEMPLES.LOGO/DESSINS/DE!
SS.CHAT
```

ramène à l'écran le dessin contenu dans le fichier DESS.CHAT.

SAUVEIMAGE

SAUVEIMAGE *nomfichier* (commande)

La commande SAUVEIMAGE sauvegarde le contenu de l'écran graphique dans un fichier indiqué par *nomfichier*. En employant RAMENEIMAGE, votre dessin apparaîtra de nouveau à l'écran.

Exemple :

```
?SAUVEIMAGE "/EXEMPLES.LOGO/DESSINS/DES!
S.CHAT
```

Fichiers copies

Cette section vous donne une description de deux primitives servant à une transcription du dialogue entre l'ordinateur Apple et vous. Ces primitives sont COPIE et NONCOPIE.

COPIE

COPIE *fichier* (commande)

La commande COPIE démarre le processus qui consiste à envoyer une copie des caractères affichés à l'écran texte à *fichier*. Cette commande sert à transcrire le dialogue entre l'ordinateur Apple et l'utilisateur. COPIE ouvre automatiquement *fichier*. NONCOPIE stoppe le processus qui débute lorsqu'on commande COPIE.

Vous ne pouvez commander FEC ou FLIS pour le fichier où la copie s'effectue. Une fois que le fichier copies a été fermé au moyen de NONCOPIE, celui-ci peut être manipulé comme tout fichier du système Logo. Ce fichier peut être ouvert de nouveau ; vous pouvez y lire ou y écrire de l'information.

Remarquez qu'un seul fichier copies peut être ouvert à la fois.

Exemples :

```
?COPIE "/DONNEES/ANNUAIRE/JUIN.COPIE
```

crée le fichier JUIN.COPIE et démarre le processus de copie. Chaque ligne d'instructions apparaissant après COPIE est envoyée dans ce fichier.

```
?VE
?AV 100
?DR 80
?AV 50
?NONCOPIE
```

Vous pouvez utiliser COPIE pour que le contenu d'un fichier soit envoyé à l'imprimante : vous obtenez une copie de votre travail sur papier. La donnée de COPIE doit être le numéro de la prise ou du logement auquel l'imprimante est reliée.

```
POUR TRANSCRIS :FICHIER
COPIE 1
IMFICHIER :FICHIER
NONCOPIE
FIN
```

NONCOPIE

NONCOPIE (commande)

La commande NONCOPIE stoppe le processus mis en route au moyen de COPIE ; la copie des caractères affichés à l'écran n'est plus envoyée au fichier ou au périphérique spécifié lorsque la commmande COPIE a été donnée.

Exemples :

```
?COPIE "/EXEMPLES.LOGO/DONNEES/ANNUAIR!
E/COPIERCLASSE
```

crée le fichier COPIERCLASSE et démarre le processus de copie.

```
?REPETE 5 <EC HASARD 10>
8
0
3
3
2
?NONCOPIE
```

Tout ce qui est affiché à l'écran texte après la commande COPIE est envoyé dans le fichier COPIERCLASSE. Si vous faites s'imprimer le contenu du fichier COPIERCLASSE, vous verrez apparaître à l'écran ce que vous venez de taper.

```
?IMFICHIER "EXEMPLES.LOGO/DONNEES/ANNU!
AIRE/COPIERCLASSE
?REPETE 5 <EC HASARD 10>
8
0
3
3
2
? NONCOPIE
```

Fichiers de données

Cette section vous fournit des indications sur :

- la lecture et l'écriture de l'information dans les fichiers de données ;
- la possibilité d'ouvrir et de fermer des fichiers de données ;
- les primitives que vous pouvez utiliser pour les fichiers de données.

Lecture et écriture des données

Dans le système de fichiers Logo, il se trouve toujours un fichier prêt pour la **lecture** et un fichier prêt pour l'**écriture**. Lorsque Logo est mis en route, le clavier sert à la lecture et l'écran à l'écriture. Vous pouvez modifier les fichiers où la lecture et l'écriture s'effectueront au moyen des commandes FLIS et FEC, primitives décrites plus loin dans ce chapitre.

Lorsque vous tentez de lire ou d'écrire dans un fichier, Logo débute la lecture ou l'écriture à des positions spécifiques dans le fichier. Par exemple, lorsque Logo ouvre un fichier, la lecture commence au début de celui-ci, et l'écriture s'effectue à la fin du fichier. Il est possible de modifier les positions où s'effectueront la lecture et l'écriture à l'aide des commandes FPOSLECT et FPOSECRIT aussi décrites dans ce chapitre.

Ouvrir un fichier

Avant de commencer la lecture, ou d'être en mesure d'écrire dans un fichier, vous devez ouvrir un fichier ou un périphérique au moyen de la commande OUVRE. Vous pouvez ouvrir jusqu'à six fichiers à la fois. Cependant, un seul périphérique peut être ouvert. Si un périphérique est ouvert, vous ne pouvez employer une primitive qui sert à ouvrir ou à fermer des périphériques de façon automatique. Vous ne pouvez, par exemple, utiliser la commande COPIE avec le logement 1 ou à la prise 1 si le logement 2 est déjà ouvert.

Les primitives que l'on peut utiliser pour les fichiers de données sont les suivantes :

FEC	FPOSECRIT	OUVRE
FERME	FPOSLECT	POINTECRIT
FERMETOUT	LONGUEURF	POINTELECT
FLIS	OUVERTS	POSECRIT
		POSLECT

FEC

FEC *fichier* (commande)

La commande FEC (fixe écris) prépare le *fichier* où l'écriture s'effectuera. Les primitives ECRIS, TAPE et MONTRE permettent d'écrire dans ce fichier. Si le fichier n'est pas déjà ouvert, vous ne pouvez commander FEC.

Pour que l'écran serve de nouveau à l'écriture, vous devez employer la commande FEC suivie, comme donnée, de la liste vide ou la commande FERME.

Note : Les commandes IM, IMN, IMNS, IMPS, IMT, IMTS, IMINDEX, IMTOUT et IMFICHIER permettent l'affichage à l'écran, non l'écriture dans un fichier.

Exemples :

```
?OUVRE 1
?FEC 1
```

Les commandes permettant l'écriture envoient l'information au périphérique relié au logement 1 ou à la prise 1.

```
?ECRIS <ANNUAIRE TELEPHONIQUE LOGO>
```

Si le périphérique connecté au logement 1 ou à la prise 1 est une imprimante, ANNUAIRE TELEPHONIQUE LOGO y sera imprimé.

```
?FEC<>
```

L'écriture s'effectue de nouveau à l'écran.

```
POUR ACCUMULER :FICHIER :DONNEES
OUVRE :FICHIER
FEC :FICHIER
ECRIS :DONNEES
FERME :FICHIER
FIN

?ACCUMULER "LISTE.TEL<ALAIN TOUGAS: 45!
1-2513>
```

FERME

FERME *fichier* (commande)

Cette commande ferme le *fichier* ou le périphérique. Voir OUVRE pour savoir comment ouvrir un fichier ou un périphérique. Il y a erreur si vous commandez FERME pour un fichier ou un périphérique qui n'est pas ouvert. Il y a aussi erreur si vous utilisez la commande FERME pour un fichier qui a été ouvert au moyen de la commande COPIE.

Attention
Vous ne devez jamais éteindre votre ordinateur lorsque des fichiers sont ouverts. Cela pourrait affecter l'organisation logique de la disquette.

Exemples :

```
?FERME "/EXEMPLES.LOGO/DONNEES/LISTE.TE!
L
```

ferme le fichier LISTE.TEL.

La procédure ACCUMULER ouvre un fichier, y envoie des données puis ferme le fichier.

```
POUR ACCUMULER :FICHIER :DONNEES
OUVRE :FICHIER
FEC :FICHIER
EC :DONNEES
FERME :FICHIER
FIN

?ACCUMULER "/EXEMPLES.LOGO/DONNEES/LIST!
E.TEL<LOUISE: 765-4201>
```

Le nom, et le numéro de téléphone qui l'accompagne, sont écrits dans le fichier LISTE.TEL.

FERMETOUT

FERMETOUT (commande)

Pour fermer des fichiers copies, voir sous la rubrique NONCOPIE.

La commande FERMETOUT ferme tous les fichiers et les périphériques. Si des fichiers copies ont été ouverts, on ne peut les fermer au moyen de FERMETOUT.

Les commandes OUVRE et FERME servent à ouvrir et à fermer un seul fichier à la fois. Lorsque des fichiers ou des périphériques ne sont pas ouverts, la commande FERMETOUT n'a aucun effet.

```
?OUVRE 1
?OUVRE "/EXEMPLES.LOGO/DONNEES/LISTE.TE!
L
```

L'imprimante reliée au logement 1 ou à la prise 1 ainsi que le fichier LISTE.TEL sont ouverts. Après avoir envoyé des données dans le fichier ainsi qu'à l'imprimante, vous les fermez en tapant :

```
?FERMETOUT
```

FLIS

FLIS *fichier* (commande)

La commande FLIS (fixe lis) prépare le *fichier* où la lecture s'effectuera. Une fois cette commande donnée, LISCAR, LISCARS, LISLISTE et LISMOT peuvent lire l'information à partir de ce fichier.

Avant de commander FLIS, vous devez ouvrir le fichier au moyen de la commande OUVRE. Il y a erreur si le fichier n'est pas ouvert. Pour que la lecture se fasse de nouveau à partir du clavier, vous devez donner la commande FLIS suivie de la liste vide comme donnée ou la commande FERME.

Exemples :

```
?FPREFIXE "/EXEMPLES.LOGO/DONNEES
?OUVRE "LISTE.TEL
?FLIS "LISTE.TEL
?ECRIS POSLECT
0
```

Le fichier LISTE.TEL est prêt pour la lecture ; la position de lecture se trouve au début du fichier.

```
?ECRIS LISMOT
RICHARD : 545-2654
```

LISMOT va lire dans le fichier ouvert pour la lecture. Pour que la lecture se fasse de nouveau à partir du clavier, tapez :

```
?FLIS<>
```

FPOSECRIT

FPOSECRIT *entier* (commande)

La commande FPOSECRIT (fixe position écriture) détermine la position où s'effectuera l'écriture dans le fichier. Cette commande sert surtout à modifier l'information contenue dans un fichier. Vous devez fixer la position d'écriture à un *entier* compris entre 0 et la position de fin de fichier. Il y a erreur si vous tentez de fixer la position en dehors de ces limites.

Il y a aussi erreur si vous fixez la position d'écriture lorsque cette dernière se fait à partir de l'écran ou d'un périphérique.

La commande POSECRIT sert à vérifier la position actuelle d'écriture.

Exemples :

```
?OUVRE "LISTE.TEL
?FEC "LISTE.TEL
?FPOSECRIT 0
?ECRIS <ANDREE : 935!-3395>
?FEC<>
```

Ainsi, le fichier LISTE.TEL est ouvert, prêt pour l'écriture ; la position d'écriture est maintenant fixée à 0 (celle-ci était localisée à la fin du fichier au moment de son ouverture). La liste <ANDREE: 935-3395> remplace ce qui se trouvait au début du fichier.

FPOSLECT

FPOSLECT *entier* (commande)

Pour plus de détails au sujet de la commande FPOSLECT, voir sous la rubrique POSLECT.

La commande FPOSLECT (fixe position lecture) détermine la position où s'effectuera la lecture dans le fichier ; *entier* doit être un nombre compris entre 0 et le nombre indiquant la longueur du fichier. Il y a erreur si *entier* n'est pas compris entre ces limites ou si la lecture se fait à partir du clavier ou encore d'un périphérique.

Exemples :

```
?OUVRE "LISTE.TEL
?FLIS "LISTE.TEL
?FPOSLECT 2
?ECRIS LISCAR
S
```

Le fichier LISTE.TEL est ouvert et prêt pour la lecture. La position de lecture est fixée à 2 et le caractère se trouvant à cette position est affiché.

```
POUR LISTE.LL :POS
FPOSLECT :POS
RETOURNE LISMOT
FIN

?ECRIS LISTE.LL 34
RENAUD 734-8374
```

La procédure LISTE.LL retourne la liste se trouvant à la position demandée dans le fichier fourni comme donnée.

LONGUEURF

LONGUEURF *nomfichier* (opération)

L'opération LONGUEURF (longueur fichier) retourne la longueur, en octets, du contenu du fichier indiqué par *nomfichier*. Le fichier doit être ouvert pour que LONGUEURF puisse être utilisée. Il y a erreur si le fichier n'est pas ouvert.

Exemple :

```
?OUVRE "/EXEMPLES.LOGO/DONNEES/ANNUAIRE!
#/ADRESSES
?ECRIS LONGUEURF "/EXEMPLES.LOGO/DONNEE!
S/ANNUAIRE/ADRESSES
128
```

Le contenu du fichier ADRESSES a une longueur de 128 octets.

```
POUR REMPLIS :FICHIER :LONG
OUVRE :FICHIER
FEC :FICHIER
RELIE "ESPACE :LONG - LONGUEURF :FICHIE!
R
SI SUPP :ESPACE Ø<REPETE :ESPACE<TAPE!
Ø>>
FEC<>
FERME :FICHIER
FIN
```

La procédure REMPLIS ouvre le fichier :FICHIER et y place des zéros jusqu'à ce que le fichier ait une longueur en octets de :LONG.

OUVERTS

OUVERTS (opération)

L'opération OUVERTS retourne une liste de tous les fichiers et des périphériques ouverts. La commande OUVRE ouvre un fichier ou un périphérique.

Exemples :

```
?ECRIS OUVERTS

?
```

Aucun fichier ou périphérique n'est ouvert.

```
?ECRIS OUVERTS
1 LISTE.TEL
```

Le périphérique relié au logement 1 ou à la prise 1, c'est-à-dire l'imprimante, ainsi que le fichier LISTE.TEL sont ouverts.

La procédure QUITTER assure que tous les fichiers sont fermés avant que vous n'éteigniez votre ordinateur.

```
POUR QUITTER
SI NON VIDEP OUVERTS <FERMETOUT>
NONCOPIE
EC <L'ORDINATEUR PEUT ETRE ETEINT.>
FIN
```

OUVRE

OUVRE *fichier* (commande)

Cette commande ouvre *fichier*; ce dernier peut ainsi recevoir ou envoyer des caractères.

Un fichier de données doit être ouvert avant que vous puissiez y avoir accès. Remarquez qu'un seul périphérique ne peut être ouvert à la fois.

Voir les commandes FERME et FERMETOUT.

Vous pouvez ouvrir un maximum de six fichiers. Si *fichier* n'existe pas, la commande OUVRE en crée un. Au moment de quitter Logo, tous les fichiers ainsi que les périphériques ouverts à ce moment doivent être fermés.

Exemple :

```
POUR LISFICHIER :FICHIER
FLIS :FICHIER
SI EGALP LONGUEURF :FICHIER POSLECT <FL!
IS <> FERME :FICHIER STOP>
ECRIS LISLISTE
LISFICHIER :FICHIER
FIN

?FPREFIXE "/EXEMPLES.LOGO/DONNEES/ANNUA!
IRE
?OUVRE "ADRESSES
?LISFICHIER "ADRESSES
LOUISE : 191 FLEURY
LOGO : 9960 COTE DE LIESSE
```

La procédure LISFICHIER permet d'aller lire l'information contenue dans un fichier déjà ouvert, jusqu'à ce que la position de fin de fichier soit atteinte (EGALP LONGUEURF :FICHIER POSLECT) ; une fois ceci complété, le fichier est fermé et l'exécution de la procédure prend fin.

POINTECRIT

POINTECRIT (opération)

L'opération POINTECRIT retourne le nom du fichier ou le périphérique ouvert pour l'écriture. Il est possible de changer le fichier ouvert pour l'écriture au moyen de FEC. Si l'écriture se fait au niveau du clavier, POINTECRIT retourne le mot vide. Comparer à l'opération OUVERTS.

Exemples :

```
POUR AJOUTE.INFO :FICHIER :DONNEES
SI NON MEMBREP :FICHIER OUVERTS <OUVRE !
:FICHIER>
RELIE "POINTEUR POINTECRIT
FEC :FICHIER
EC :DONNEES
FEC :POINTEUR
FIN

?AJOUTE.INFO "/EXEMPLES.LOGO/DONNEES/LI!
STE.CLASSE <ALAIN TOUGAS>
```

La procédure AJOUTE.INFO vérifie d'abord si un fichier est ouvert. Si tel n'est pas le cas, AJOUTE.INFO ouvre le fichier ; ce dernier est prêt pour l'écriture, et des données peuvent y être envoyées. La procédure AJOUTE.INFO permet que l'écriture s'effectue dans le fichier déjà ouvert.

POINTELECT

POINTELECT (opération)

L'opération POINTELECT retourne le nom du fichier ou le périphérique ouvert pour la lecture. Il est possible de changer le fichier ouvert pour la lecture au moyen de FLIS. Si la lecture se fait au niveau du clavier, POINTELECT retourne le mot vide. Comparer avec OUVERTS.

Exemples :

```
?ECRIS POINTELECT
/EXEMPLES.LOGO/DONNEES/ANNUAIRE/ADRESSE!
S
```

Le fichier ADRESSES est ouvert et prêt pour la lecture.

```
POUR LISFICHIER :FICH
RELIE "POINTEUR POINTELECT
```

Logo mémorise le fichier déjà ouvert pour la lecture.

```
SI NON EGALP :POINTEUR
:FICH <OUVRE :FICH FLIS
:FICH>
```

Logo vérifie s'il s'agit du même fichier que celui dans lequel on veut lire à ce moment-ci. Si ce n'est pas le cas, Logo ouvre le nouveau fichier et place un pointeur à l'endroit où s'effectuera la lecture.

```
LISLIGNES :FICH
```

Logo lit le fichier ligne par ligne jusqu'à ce qu'il n'y ait plus d'information.

```
FLIS :POINTEUR
```

Logo ramène le pointeur de lecture au fichier originalement ouvert.

```
FIN

POUR LISLIGNES :FICH
SI EGALP LONGUEURF :FICH POSLECT <FERME!
 :FICH STOP>
EC LL
LISLIGNES :FICH
FIN
```

La procédure LISFICHIER demande à Logo de mémoriser dans quel fichier le pointeur est positionné, d'ouvrir un nouveau fichier, de placer le pointeur dans ce nouveau fichier, d'en lire le contenu, puis de replacer le pointeur là où il se trouvait dans le premier fichier ouvert pour la lecture.

POSECRIT

POSECRIT (opération)

L'opération POSECRIT (position écriture) retourne la position où le prochain caractère ira s'inscrire dans le fichier ouvert pour l'écriture. Il y a erreur si l'écriture s'effectue à l'écran ou sur un périphérique.

Exemples :

```
?OUVRE "LISTE.TEL
?FEC "LISTE.TEL
?RELIE "POS POSECRIT
?FEC<>
?EC :POS
33
```

Remarquez que vous ne pouvez pas taper ECRIS POSECRIT car la position où l'écriture s'effectue ira s'inscrire dans le fichier LISTE.TEL.

La procédure VERIFIEPOS affiche la position du fichier ouvert après l'avoir fixé en écriture.

```
POUR VERIFIEPOS
RELIE "POS POSECRIT
RELIE "FICHIER POINTECRIT
FEC<>
EC :POS
FEC :FICHIER
FIN

?VERIFIEPOS
33
```

POSLECT

POSLECT (opération)

L'opération POSLECT (position lecture) retourne la position où s'effectuera la lecture dans le fichier. Il y a erreur si la lecture se fait à ce moment à partir du clavier ou d'un périphérique. Voir la commande FPOSLECT pour fixer la position où la lecture s'effectuera dans le fichier.

Exemples :

```
?FPREFIXE "/EXEMPLES.LOGO/DONNEES/ANNUA!
IRE
?OUVRE "LISTE.TEL
?FLIS "LISTE.TEL
?ECRIS POSLECT
0
```

Si vous venez d'ouvrir un fichier pour la lecture, POSLECT retourne 0.

La procédure ECRISFICH affiche l'information conservée dans le fichier à partir duquel la lecture s'effectue, ainsi qu'un nombre correspondant à chacune des lignes où l'information est stockée.

```
POUR ECRISFICH :FICHIER
SI EGALP LONGUEURF :FICHIER POSLECT <ST!
OP>
ECRIS POSLECT
ECRIS LISMOT
ECRISFICH :FICHIER
FIN

?OUVRE "LISTE.TEL
?FLIS "LISTE.TEL
?ECRISFICH "LISTE.TEL
0
RENAUD 734-8374
16
ANDREE 935-3395
```

Un projet utilisant un fichier de données

Cette section permet d'utiliser le système de fichiers de données. Le projet, un annuaire téléphonique, servira à répertorier les numéros de téléphone des membres d'une association. Les objectifs de ce projet sont les suivants :

1. Stocker les noms des membres et leur numéro de téléphone respectif.
2. Trouver le numéro de téléphone d'un membre.
3. Changer le numéro de téléphone d'un des membres.

Première étape : créer un fichier de données

La procédure qui suit permet de lire le nom d'une personne ainsi que son numéro de téléphone à partir du clavier.

```
POUR DEMANDEINFO
ECRIS <Tapez le nom du membre:>
RELIE "NOM LISMOT
ECRIS <Tapez le numéro de téléphone:>
RELIE "TEL LISMOT
FIN
```

La procédure DEMANDEINFO imprime le message à l'écran, prend la réponse provenant du clavier et donne un nom à cette réponse. Lorsque l'exécution de DEMANDEINFO est terminée, deux variables sont créées : l'une s'appelle NOM, l'autre, TEL. L'étape suivante consiste à placer ces renseignements dans un fichier.

La commande FLIS sert à envoyer de l'information à des fichiers ou à des périphériques.

```
POUR ECRIREINFO
FEC "MEMBRES
ECRIS :NOM
ECRIS :TEL
FEC<>

FIN
```

MEMBRES est le nom du fichier.

L'affichage se fait de nouveau à l'écran.

Il ne vous reste plus qu'à écrire la superprocédure qui ouvre le fichier MEMBRES, à faire exécuter les sous-procédures et à fermer le fichier.

```
POUR SAUVEINFO
OUVRE "MEMBRES
DEMANDEINFO
ECRIREINFO
FERME "MEMBRES
FIN
```

Essayez votre procédure :

```
?SAUVEINFO
Tapez le nom du membre :
Alain Tougas
Tapez le numéro de téléphone :
451-2513
?
```

L'exécution de la procédure est terminée mais vous n'avez pas été en mesure de voir ce qu'il est advenu du fichier de données. Ainsi, tapez :

```
?IMFICHIER "MEMBRES
```

Logo affiche tout ce qui se trouve dans le fichier MEMBRES.

```
Alain Tougas
451-2513
```

Qu'arrivera-t-il si la procédure est exécutée de nouveau ?

```
SAUVEINFO
Tapez le nom du membre:
Renaud Nadeau
Tapez le numéro de téléphone:
734-8374
?
```

La procédure SAUVEINFO a fonctionné comme la première fois où vous l'avez essayée. Voyez maintenant le résultat.

```
?IMFICHIER "MEMBRES
Alain Tougas
451-2513
Renaud Nadeau
734-8374
```

Cette procédure sert à ajouter d'autres noms de personnes accompagnés de numéros de téléphone ainsi qu'à créer un fichier de données.

Deuxième étape : ramener des données

Une fois que vous avez créé un fichier de données qui contient des noms de personnes et des numéros de téléphone, l'étape suivante consiste à écrire un programme qui servira à trouver le numéro de téléphone d'une personne.

```
POUR TROUVEINFO
ECRIS <Tapez le nom du membre:>
RELIE "NOM LISLISTE
OUVRE "MEMBRES
FLIS "MEMBRES
CHERCHETEL :NOM
FEC <>
FERME "MEMBRES
FIN

POUR CHERCHETEL :NOM
SI LISLISTE = :NOM <EC PH <Le numéro de!
 téléphone est> LISMOT STOP>
SI EGALP LONGUEURF "MEMBRES POSLECT <EC!
<Ce nom ne se trouve pas dans la liste!
.> STOP>
CHERCHETEL :NOM
FIN
```

La superprocédure TROUVEINFO fait la lecture, à partir du clavier, du nom de la personne pour laquelle on désire avoir le numéro de téléphone. Puis elle ouvre un fichier de données et informe Logo que la lecture s'effectuera à partir de ce fichier.

La sous-procédure CHERCHETEL démarre la lecture ligne par ligne (au moyen de LISLISTE) à partir du début de ce fichier de données. Après la lecture de chaque ligne, CHERCHETEL compare cette ligne et le nom recherché. Si ce dernier et la ligne lue sont identiques, la procédure lit une autre ligne et affiche, par exemple :

```
Le numéro de téléphone est 734-8374
```

Si tel n'est pas le cas, CHERCHETEL vérifie si LISLISTE a atteint la position de fin de fichier (EGALP LONGUEURF "MEMBRES POSLECT). Si la position de fin de fichier est atteinte, CHERCHETEL affichera le message :

```
Ce nom ne se trouve pas dans la liste.
```

Troisième étape : modifier les données

L'un des membres peut changer de numéro de téléphone ; vous devez donc mettre à jour les données contenues dans le fichier. Pour modifier des données, vous devez savoir à quel endroit elles se trouvent. Les procédures servant à ramener les données (TROUVEINFO et CHERCHETEL) peuvent être utilisées. Une fois l'endroit trouvé, vous pouvez utiliser la procédure MODIFIE ; celle-ci récrira les nouveaux renseignements à cet endroit.

```
POUR MODIFIE :ENDROIT
ECRIS <Tapez le nouveau numéro de télép!
hone:>
FLIS <>
FEC "MEMBRES
FPOSECRIT :ENDROIT
ECRIS LISMOT
FEC <>
FIN
```

FLIS <> indique à Logo que la lecture des données s'effectue à partir du clavier ; FEC "MEMBRES informe Logo que la commande ECRIS effectuera l'écriture de la nouvelle donnée dans le fichier MEMBRES. FPOSECRIT :ENDROIT assure que la donnée est écrite à l'endroit désigné.

Ainsi, au moyen de l'instruction ECRIS LISMOT, les données provenant du clavier sont écrites dans le fichier.

A présent, MODIFIE doit faire partie d'une nouvelle procédure. TROUVENOM effectuera la lecture du fichier ligne par ligne en comparant chacune de ces lignes au nom recherché. Cette procédure appelle MODIFIE qui conserve l'ENDROIT obtenu au moyen de POSLECT. Il s'agit maintenant de créer une superprocédure, CHANGEINFO, qui effectue tout le travail.

```
POUR CHANGEINFO
ECRIS <Tapez le nom du membre : >
RELIE "NOM LISLISTE
OUVRE "MEMBRES
FLIS "MEMBRES
TROUVENOM :NOM
FLIS <>
FERME "MEMBRES
FIN

POUR TROUVENOM :NOM
SI LL = :NOM <MODIFIE POSLECT STOP>
SI EGALP LONGUEURF "MEMBRES POSLECT <EC !
# <Ce nom ne se trouve pas dans la liste !
. > STOP>
TROUVENOM :NOM
FIN
```

Listes de propriétés

Listes de propriétés

A tout mot Logo peut être associée une **liste de propriétés**. Une telle liste comprend un nombre pair d'éléments. Chaque couple d'éléments contient une propriété et sa valeur ; cette dernière peut être un mot ou une liste.

Une liste de propriétés est donc de la forme PROP1 VAL1 PROP2 VAL2. On utilise les primitives décrites dans ce chapitre pour manipuler les listes de propriétés. Ce sont :

ANNULEPROP
DPROP
EFPROPS
IMPROPS
PLISTE
RPROP

SAUVE ET SAUVEL sont décrites au chapitre 16.

Les commandes SAUVE et SAUVEL sauvegardent les listes de propriétés dans des fichiers en même temps qu'elles mettent en réserve les procédures et les variables.

Utilisation des listes de propriétés

Les listes de propriétés vous permettent de conserver différents types de bases de données. L'exemple qui suit explique comment utiliser les primitives ayant trait aux listes de propriétés.

Supposez que vous voulez conserver le numéro de téléphone et la date d'anniversaire de vos amis. Choisissez un mot Logo, disons A1, qui vous permette de repérer les renseignements qui concernent votre premier ami. Puis tapez :

```
DPROP "A1 "NOM <ALAIN TOUGAS>
DPROP "A1 "TEL <514 689 0745>
DPROP "A1 "ANNIVERSAIRE <18 JUIN>
```

Répétez ces étapes pour chacun de vos amis en attribuant au second le mot de repère A2. Par exemple :

```
DPROP "A2 "NOM <JULIEN PERRON>
DPROP "A2 "TEL <514 332 8694>
DPROP "A2 "ANNIVERSAIRE <15 OCTOBRE>

DPROP "A3 "NOM <MARIE!-ANDREE THEORET>
DPROP "A3 "TEL <514 738 0101>
DPROP "A3 "ANNIVERSAIRE <16 JUILLET>
```

Lorsque vous avez terminé, créez une liste des mots de repère comme ceci :

```
RELIE "AMIS <A1 A2 A3>
```

Vous pourrez ensuite utiliser la primitive RPROP pour définir des procédures qui vérifieront dans la liste AMIS la date d'anniversaire de l'un d'entre eux ou qui dresseront la liste de tous vos amis dont le numéro de téléphone comporte le même code régional. La description des primitives, dans les pages qui suivent, fournit des exemples de procédures de ce genre.

ANNULEPROP

ANNULEPROP *nom prop* (commande)

Voir aussi IMPROPS et RPROP.

La commande ANNULEPROP (annule propriétés) retire *prop* de la liste de propriétés *nom*.

Exemple :

```
?MONTRE PLISTE "A1
<NOM <ALAIN TOUGAS> TEL <514 689 0745> A!
NNIVERSAIRE <18 JUIN>
?ANNULEPROP "A1 "TEL
?MONTRE PLISTE "A1
<NOM <ALAIN TOUGAS> ANNIVERSAIRE <18 JU!
IN>>
```

DPROP

DPROP *nom prop obj* (commande)

La commande DPROP (définis propriété) confère à *nom* la propriété *prop* avec la valeur *obj*. Notez que EFTOUT efface les procédures, les variables et les propriétés. Utilisez ANNULEPROP pour effacer les propriétés une à la fois et EFPROPS pour les effacer en bloc.

Exemple :

```
?MONTRE PLISTE "A3
<NOM <MARIE! -ANDREE THEORET> TEL <514 73!
8 0101> ANNIVERSAIRE <16 JUILLET>
?DPROP "A3 "ADRESSE <54 FRIGON>
?MONTRE PLISTE "A3
<NOM <MARIE! -ANDREE THEORET> TEL <514 73!
8 0101> ANNIVERSAIRE <16 JUILLET> ADRESS!
E <54 FRIGON>>
```

EFPROPS

EFPROPS (commande)

La commande EFPROPS (efface propriétés) efface toutes les propriétés contenues dans l'espace de travail. Pour vérifier les propriétés que l'espace contient, utilisez IMPROPS. Employez ANNULEPROP pour retirer les propriétés une à une de l'espace de travail.

IMPROPS

IMPROPS (commande)

La commande IMPROPS (imprime propriétés) affiche toutes les listes de propriétés contenues dans l'espace de travail.

Exemple :

```
?IMPROPS
DPROP "A3 "NOM <MARIE!-ANDREE THEORET>
DPROP "A3 "TEL <514 738 0101>
DPROP "A3 "ANNIVERSAIRE <16 JUILLET>
DPROP "A3 "ADRESSE <54 FRIGON>
DPROP "A2 "NOM <JULIEN PERRON>
DPROP "A2 "TEL <514 332 8694>
DPROP "A2 "ANNIVERSAIRE <15 OCTOBRE>
DPROP "A1 "NOM <ALAIN TOUGAS>
DPROP "A1 "TEL <514 689 0745>
DPROP "A1 "ANNIVERSAIRE <18 JUIN>
```

PLISTE

PLISTE *nom* (opération)

L'opération PLISTE (propriétés, liste de) retourne la liste de propriétés associée à *nom*. Il s'agit d'une liste de noms de propriétés ainsi que de leur valeur affichée sous la forme <PROP1 VAL1 PROP2 VAL2>.

Exemple :

```
?MONTRE PLISTE "A2
<NOM <JULIEN PERRON> TEL <514 332 9694>!
 ANNIVERSAIRE <15 OCTOBRE>
```

La procédure ANNIVERSAIRE? retourne la date d'anniversaire d'un ami donné.

```
POUR ANNIVERSAIRE? :AMI :AMIS
SI VIDEP :AMIS <RT <AUCUNE>>
SI EGALP PREMIER SP PLISTE PREMIER :AMI!
S :AMI <RT RPROP PREMIER :AMIS "ANNIVER!
SAIRE>
RT ANNIVERSAIRE? :AMI SP :AMIS
FIN
```

```
?EC ANNIVERSAIRE? <JULIEN PERRON> :AMIS
15 OCTOBRE
```

RPROP *nom prop* (opération)

L'opération RPROP (retourne propriété) retourne la valeur de la *prop* associée à *nom* ; RPROP retourne la liste vide si cette propriété n'existe pas.

Exemple :

```
?MONTRE RPROP "A1 "NOM
<ALAIN TOUGAS>
```

La procédure LISTETEL affiche le nom et le numéro de téléphone de vos amis.

```
POUR LISTETEL :AMIS
SI VIDEP :AMIS <STOP>
EC PH RPROP PREMIER :AMIS "NOM RPROP PR!
EMIER :AMIS "TEL
LISTETEL SP :AMIS
FIN

?LISTETEL :AMIS
ALAIN TOUGAS 514 689 0745
JULIEN PERRON 514 332 8694
MARIE-ANDREE THEORET 514 738 0101
```

Primitives spéciales

Primitives spéciales

chapitre 18

Ce chapitre décrit les primitives spéciales qui peuvent agir directement sur le système Logo lui-même. Elles vous permettent d'avoir accès à la mémoire de l'ordinateur ou d'en modifier le contenu. Par le fait même, ces primitives doivent être utilisées avec précaution car elles peuvent détruire tout ce que l'espace de travail contient ; dans ce cas, vous devez remettre Logo en route. Les noms de ces primitives sont précédés d'un point pour vous rappeler de les employer avec circonspection. Il est recommandé de sauvegarder votre travail avant de les utiliser.

Les primitives spéciales sont divisées en trois groupes :

- langage d'assemblage et accès direct à la mémoire ;
- graphique ;
- autres.

Langage d'assemblage et primitives

Cette section décrit les primitives qui vous permettent d'utiliser des programmes en langage d'assemblage et d'avoir accès direct à la mémoire. Vous y trouverez aussi d'utiles renseignements ayant trait à la mémoire de l'Apple ; ces indications vous permettent de programmer en langage d'assemblage.

Indications au sujet de la mémoire Apple

Voyez le manuel de référence technique accompagnant votre ordinateur pour plus de détails sur la répartition de la mémoire.

La mémoire de l'Apple IIe et de l'Apple IIc est divisée en deux blocs de 64 Ko chacun, soit le bloc principal et le bloc auxiliaire. Le tableau des correspondances qui suit indique comment Logo utilise ces deux blocs.

Tableau 18-1. *Tableau des correspondances de la mémoire principale*

Mémoire principale	
FFFF	
	ProDOS
D000	
	Espace d'E/S
C000	
	ProDOS
BF00	
	Code Logo
6100	
	Données Logo
6000	
	Tampon fichier 5
5C00	
	Tampon fichier 4
5800	
	Tampon fichier 3
5400	
	Tampon fichier 2
5000	
	Tampon fichier 1
4C00	
	Tampon fichier 0
4800	
	Tampon spécial pour COPIE
4400	
	Tampon spécial pour RAMENE et SAUVE
4000	
	Graphiques haute résolution
2000	
	Tampon d'édition
800	
	Ecran texte 1
400	
	Données Logo
0	

Tableau 18-2. *Tableau des correspondances de la mémoire auxiliaire*

Mémoire auxiliaire

FFFF	ProDOS
E000	Code Logo
D000	Espace d'E/S
C000	ProDOS
BF00	Editeur d'aide
BB00	Editeur d'aide principal
B700	Espace nœuds
800	Ecran texte 2
400	ProDOS
200	Données Logo
0	

Pour écrire des programmes en langage d'assemblage, vous devez connaître certaines adresses que l'on retrouve dans les deux blocs mémoire. Ces adresses sont présentées au tableau 18-3.

Tableau 18-3. *Tableau des adresses mémoire*

	Adresse		Valeur normale	
Information	**Hex**	**Décimal**	**Hex**	**Décimal**
Nombre maximal de fichiers de données (multiplié par 9)	300	768	36	54
Pointeur à la première page après l'espace nœuds	10	16	B7	183
Indicateur pour tampon d'édition invalide	301	769	0	0

Utilisation de l'espace tampon

Vous pouvez utiliser le tampon d'édition, le tampon graphique et les tampons fichiers pour écrire vos programmes dans la mesure où Logo n'emploie pas ces tampons lorsque vos programmes sont exécutés.

Le tampon d'édition et le tampon graphique ne devraient servir qu'à un stockage temporaire ; ce dernier ne vous est nécessaire que lorsque le code d'assemblage est exécuté. Lorsque vous utilisez le tampon d'édition, n'oubliez pas de placer l'indicateur à l'endroit où se trouvent les contenus invalides. Si vous employez le tampon graphique, assurez-vous de bien le vider lorsque vous en avez terminé afin d'éviter tout affichage graphique imprévisible.

Les tampons fichiers peuvent aussi servir au code en langage d'assemblage. Pour vous assurer que Logo n'utilise pas les tampons avec lesquels vous travaillez, il vous faut changer le nombre de fichiers que Logo emploie à l'adresse indiquée au tableau 18-3. Remarquez que le nombre correspond à neuf fois le nombre de fichiers que Logo peut manipuler.

Si vous avez besoin de 2K-octets pour écrire votre code, vous pouvez changer le nombre de fichiers que Logo peut avoir ouverts de 54 à 36. Ainsi les tampons 4 et 5 seront libérés pour vos besoins.

Utilisation de l'espace nœuds

Vous pouvez utiliser l'espace nœuds pour écrire vos programmes en langage d'assemblage. Vous ne pouvez réserver l'espace nœuds qu'à la mise en route de Logo quel que soit le moment où vous avez l'intention de l'employer. Vous réservez l'espace en changeant l'adresse à la fin de l'espace nœuds. Référez-vous au tableau 18-3.

A la mise en route de Logo, l'espace nœuds commence à $800 et se termine à $B6FF. Pour réserver 8K-octets d'espace nœuds, changez le $B7 en $97 à l'adresse indiquée au tableau 18-3. Rappelez-vous de libérer les nœuds en multiples de 5 octets (longueur des nœuds).

.APPELLE

.APPELLE *adr* (commande)

Cette commande transfère le contrôle à la sous-routine en langage machine commençant à l'adresse *adr* (décimal) contenue dans le bloc principal. Une instruction RTS dans votre sous-routine redonne le contrôle à Logo.

.AUXDEPOSE

.AUXDEPOSE *adr octet* (commande)

Cette commande emmagasine la valeur *octet* à l'adresse *adr* dans le bloc mémoire auxiliaire.

.AUXEXAMINE

.AUXEXAMINE *adr* (opération)

Cette opération retourne la valeur emmagasinée à l'adresse *adr* dans le bloc mémoire auxiliaire.

.BRAMENE

.BRAMENE *nomfichier adr* (commande)

Cette commande ramène un fichier composé de données ou d'un code en langage d'assemblage à l'adresse *adr* du bloc mémoire principal.

.BSAUVE

.BSAUVE *nomfichier adr entier* (commande)

Cette commande copie une section du bloc mémoire principal et la place dans un fichier indiqué par *nomfichier*. Cette partie transférée de l'espace mémoire commence à l'adresse *adr* pour une longueur de *entier* octets.

.DEPOSE

.DEPOSE *adr octet* (commande)

Cette commande écrit *octet* à l'adresse-machine *adr* (décimal) de la mémoire principale.

.EXAMINE

.EXAMINE *adr* (opération)

Cette opération retourne le contenu de l'adresse-machine *adr* (décimal) de la mémoire principale.

Le graphique

Les primitives graphiques vous permettent de vérifier et de changer le **rapport d'échelle**, soit le rapport entre un pas vertical et un pas horizontal de la Tortue. A la mise en route de Logo, le rapport d'échelle est fixé à .8.

Si vous désirez modifier le rapport d'échelle lorsque les carrés que vous dessinez ont l'allure de rectangles et les cercles celle d'ellipses, vous devrez utiliser la primitive .FECHELLE.

.ECHELLE

.ECHELLE (opération)

Voir aussi .FECHELLE.

Cette opération retourne le **rapport d'échelle**, c'est-à-dire un nombre décimal qui représente le rapport entre un pas vertical et un pas horizontal de la Tortue. A la mise en route de Logo, le rapport d'échelle est fixé à .8.

.FECHELLE

.FECHELLE *nombre* (commande)

La commande. FECHELLE (fixe échelle) fixe le rapport d'échelle à *nombre*. Ce rapport correspond à la taille d'un pas vertical par rapport à celle d'un pas horizontal. Lorsque vous utilisez .FECHELLE, la valeur réelle des coordonnées demeure la même. Cependant, ce que vous voyez à l'écran est différent.

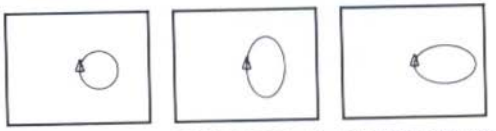
Exemple :

.FECHELLE .5 donne, à la longueur de chaque pas vertical de la Tortue, une valeur ayant la moitié de celle d'un pas horizontal.

.FECHELLE a deux rôles. Premièrement, elle permet de corriger les carrés qui ont l'allure de rectangles et les cercles celle d'ellipses. Le rapport idéal pour la plupart des écrans est .8. Deuxièmement, .FECHELLE permet de comprimer ou d'étirer les dessins de la Tortue. Par exemple, vous pouvez utiliser une procédure de cercle pour dessiner une ellipse.

```
POUR CERCLE :RAYON
REPETE 60<AV :RAYON * 3.14159 / 30 DR !
6>
FIN

POUR ELLIPSE :HORIZ :VERT
.FECHELLE .8 * :VERT / :HORIZ
CERCLE :HORIZ
.FECHELLE .8
FIN
```



CERCLE 25 ELLIPSE 25 40 ELLIPSE 40 25

Primitives diverses

Cette section décrit les primitives .CONTENU et .QUITTE.

.CONTENU

.CONTENU (opération)

Voir l'annexe D pour plus de détails sur l'espace nœuds.

Cette opération retourne une liste de tous les mots que Logo connaît. Cette liste comprend vos variables, vos procédures et vos propriétés, les primitives Logo, la majeure partie de ce que vous avez tapé et quelques autres mots.

.QUITTE

.QUITTE (commande)

L'utilisation de .QUITTE constitue la façon la plus sûre de quitter Logo. Elle assure que tous vos fichiers sont fermés et que votre travail est sauvegardé.

Annexes

Messages

Cette annexe contient tous les messages que Logo peut vous retourner. Les mots *fichier* et *nom* (en minuscules) sont remplacés par le mot précis lorsque le message est affiché.

Tableau A-1. *Messages*

Nombre	Message
1	nom DEJA DEFINIE
2	NOMBRE TROP GRAND
3	DISQUETTE REMPLACEE
6	nom EST UNE PRIMITIVE
7	TROUVE PAS L'ETIQUETTE nom
8	nom IMPOSSIBLE EN EDITION
9	nom NON DEFINIE
10	nom N'A RIEN RETOURNE A nom
11	PROBLEME DE DISQUETTE
12	DISQUETTE REMPLIE
13	NE PEUX DIVISER PAR ZERO
15	FICHIER fichier EXISTE
16	FICHIER fichier PROTEGE
17	FICHIER fichier INEXISTANT
18	fichier PAS UN FICHIER LOGO

Tableau A-1. *Messages*

Nombre	Message
19	PAS ASSEZ D'ELEMENTS DANS nom
20	TROP DE FICHIERS OUVERTS
21	PAS D'ATTRAPE POUR nom
23	MEMOIRE REMPLIE
24	nom NE PEUT ETRE UTILISE
26	PAUSE...
27	PAS AU NIVEAUSUP
28	ARRET!
29	PAS ASSEZ DE DONNEES POUR nom
30	TROP DE DONNEES POUR nom
31	TROP D'ELEMENTS ENTRE ()
33	SEULEMENT DANS UNE PROCEDURE
34	TORTUE HORS LIMITES
35	NE SAIS QUE FAIRE POUR nom
36	nom N'A PAS DE VALEUR
37	")" IMPREVUE
38	NE SAIS QUE FAIRE AVEC nom
40	DISQUETTE PROTEGEE
41	nom N'AIME PAS nom COMME DONNEE
44	AUCUN FICHIER CHOISI
45	FICHIER fichier FERME
46	FICHIER fichier OUVERT
47	POSITION FICHIER HORS PORTEE
48	PERIPHERIQUE PAS DISPONIBLE

Tableau A-1. *Messages*

Nombre	Message
50	DEJA EN MODE COPIE
52	PERIPHERIQUE UTILISE
53	FICHIER fichier TROP GRAND
54	NOM DE DISQUETTE INTROUVABLE POUR fichier
55	SOUS-INDEX INEXISTANT POUR fichier
56	SOUS-INDEX nom VIDE

Instruments utiles

Les procédures définies dans cette annexe vous seront utiles lorsque vous construirez vos propres procédures. Certaines ont déjà été définies à titre d'exemple pour illustrer le fonctionnement de primitives ; d'autres sont définies pour la première fois. Vous trouverez ces procédures sur la disquette Logo dans le fichier OUTILS.

Pour le graphique

L'annexe A du manuel *Introduction à la programmation* fournit d'autres procédures d'arcs et de cercles.

Les procédures regroupées dans cette section peuvent servir à :

- dessiner un arc qui se trace vers la gauche ou vers la droite (ARCG ou ARCD) ;
- dessiner un cercle qui se trace vers la gauche ou vers la droite (CERCLEG ou CERCLED) ;
- dessiner un polygone (POLY).

ARCD et ARCG

ARCD et ARCG dessinent des arcs qui se tracent respectivement vers la droite et vers la gauche. Leurs données sont les suivantes :

:RAYON	le rayon d'un cercle à partir duquel l'arc se dessine ;
:DEGRES	le degré de l'arc (la longueur de la courbe).

```
POUR ARCD :RAYON :DEGRES
LOCALE <PAS EXCEDENT>
RELIE "PAS 2 * :RAYON * 3.1416 / 36
RELIE "EXCEDENT RESTE :DEGRES 10
REPETE :DEGRES / 10 <DR 5 AV :PAS DR 5>
SI SUPP :EXCEDENT 0 <AV :PAS * :EXCEDEN!
T / 10 DR :EXCEDENT>
FIN

POUR ARCG :RAYON :DEGRES
LOCALE <PAS EXCEDENT>
RELIE "PAS 2 * :RAYON * 3.1416 / 36
RELIE "EXCEDENT RESTE :DEGRES 10
REPETE :DEGRES / 10 <GA 5 AV :PAS GA 5>
SI SUPP :EXCEDENT 0 <AV :PAS * :EXCEDEN!
T / 10 GA :EXCEDENT>
FIN
```

CERCLED et CERCLEG

CERCLED et CERCLEG dessinent des cercles qui se tracent vers la droite et vers la gauche. Leur donnée correspond au rayon.

```
POUR CERCLED :RAYON
LOCALE "PAS
RELIE "PAS 2 * :RAYON * 3.1416 / 36
REPETE 36 <DR 5 AV :PAS DR 5>
FIN

POUR CERCLEG :RAYON
LOCALE "PAS
RELIE "PAS 2 * :RAYON * 3.1416 / 36
REPETE 36 <GA 5 AV :PAS GA 5>
FIN
```

POLY

POLY dessine un polygone jusqu'à ce que vous stoppiez la procédure.

```
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN
```

Pour les mathématiques

Les procédures définies dans cette section peuvent servir à :

- trouver la valeur absolue d'un nombre (ABS) ;
- vérifier si un nombre est un diviseur d'un autre nombre (DIVISEURP) ;
- utiliser la fonction exponentielle (EXP) ;
- calculer le logarithme naturel d'un nombre (LN) ;
- calculer le logarithme en base 10 d'un nombre (LOG) ;
- trouver la valeur d'un nombre élevé à une puissance donnée (PUISSANCE) ;
- transformer la valeur d'un nombre d'une base à une autre (TRANSFORME).

ABS

ABS retourne la valeur absolue d'un nombre.

```
POUR ABS :NOMBRE
RT SI INFP :NOMBRE 0<-:NOMBRE><:NOMB!
RE>
FIN
```

DIVISEURP

DIVISEURP retourne VRAI si la deuxième donnée est un diviseur de la première ; sinon, retourne FAUX.

```
POUR DIVISEURP :DIVIDENDE :DIVISEUR
RT 0 = RESTE :DIVIDENDE :DIVISEUR
FIN
```

EXP

EXP correspond à la fonction exponentielle, calculée au moyen de séries de Taylor. La variable E est locale afin de s'assurer qu'elle contient toujours la valeur exacte.

```
POUR EXP :N
LOCALE "E
RELIE "E 2.71828
SI (:N - ENTIER :N) = 0<RT PUISS.ENT :!
E :N>
RT (PUISS.ENT :E ENTIER :N) * (1 + FRAC!
.E (:N - ENTIER :N) 1 1 )
FIN

POUR FRAC.E :FRAC :COMPTE :TERME
SI SUPP :COMPTE 9<RT 0>
RELIE "TERME :TERME * :FRAC / :COMPTE
RT :TERME + FRAC.E :FRAC :COMPTE + 1 :T!
ERME
FIN
```

Essayez ceci :

```
?EC EXP 1
2.71828
?EC EXP 5
148.413
```

LN

LN calcule le logarithme naturel d'un nombre en utilisant les procédures et sous-procédures mathématiques qui suivent.

```
POUR LN :N
LOCALE "LISTE.RAC
SI INFP :N 0<RT <NE PEUX TROUVER LE L!
OG D'UN NOMBRE NEGATIF>>
SI :N = 1<RT 0>
SI INFP :N 1<RELIE "LISTE.RAC RACINE!
 (1 / :X) 1 -1><RELIE "LISTE.RAC RACINE!
 :N 1 1 >
RT (PREMIER SP :LISTE.RAC) * (LN1 PREMI!
ER :LISTE.RAC) / (DERNIER :LISTE.RAC)
FIN

POUR RACINE :N :PUISS.NAT :CONST
SI INFP :N 1.2<RT (LISTE :N :PUISS.NA!
 T :CONST)>
RT RACINE (RC :N) (2 * :PUISS.NAT) :CON!
ST
FIN
```

```
POUR LN1 :N
RELIE "N (:N - 1) / (:N + 1)
RT 2 * (:N + (PUISSANCE :N 3) / 3 + (PU!
ISSANCE :N 5) / 5)
FIN
```

Essayez ceci :

```
EC LN 50
3.91201
?EC LN 2.71828
0.999998
```

LOG

LOG retourne le logarithme en base 10 d'un nombre. Cette procédure utilise LN définie précédemment.

```
POUR LOG :N
RT 0.434294 * LN :N
FIN
```

PUISSANCE

PUISSANCE retourne N élevé à la puissance X. Si X est une fraction et que N n'est pas égal à 1, PUISSANCE utilise les fonctions naturelles EXP et LN. Si N est plus petit que 0 et que X est une fraction, le résultat sera un nombre complexe.

```
POUR PUISSANCE :N :X
SI ET (INFP :N 0) NON (:X = ENTIER :X)!
<EC (PH :N<ELEVE A> :X<EST UN NOMBR!
E COMPLEXE>) STOP>
SI OU :N = 1 :X = ENTIER :X<RT PUISS.ENT!
 :N :X>
RT EXP (LN :N) * :X)
FIN

POUR PUISS.ENT :N :P.ENT
SI OU :N = 1 :P.ENT = 0<RT 1>
SI INFP :P.ENT 0<RT 1 / BOUCLE.PUISS!
 (:N) (-:P.ENT)>
RT BOUCLE.PUISS :N :P.ENT
FIN
```

```
POUR BOUCLE.PUISS :N :P.ENT
SI :P.ENT = 0<RT 1>
RT :N * BOUCLE.PUISS :N :P.ENT - 1
FIN
```

Essayez ceci :

```
?EC PUISSANCE 2 3
8
?EC PUISSANCE 3 2
9
?EC PUISSANCE 3 0
1
```

TRANSFORME

TRANSFORME change la valeur du nombre NB d'une base (:DE.BASE) à une autre (:A.BASE).

```
POUR TRANSFORME :NB :DE.BASE :A.BASE
RT DEC.A.QQBASE QQBASE.A.DEC :NB :DE.BA!
SE 1 :A.BASE
FIN

POUR QQBASE.A.DEC :NB :BASE :PUISSANCE
SI VIDEP :NB<RT 0>
RT (:PUISSANCE * CAR.A.NB DERNIER :NB) !
+ QQBASE.A.DEC SD :NB :BASE :PUISSANCE !
* :BASE
FIN

POUR DEC.A.QQBASE :NB :BASE
SI INFP :NB :BASE<RT NB.A.CAR :NB>
RT MOT DEC.A.QQBASE ENTIER QUOTIENT :NB!
 :BASE :BASE NB.A.CAR RESTE :NB :BASE
FIN

POUR CAR.A.NB :NB
SI NOMBREP :NB<RT :NB>
RT (ASCII :NB) - 55
FIN

POUR NB.A.CAR :NB
SI INF :NB 10<RT :NB>
RT CAR 55 + :NB
FIN
```

Il est aussi possible d'utiliser TRANSFORME pour transformer les décimales en hexadécimales et l'inverse. Remarquez que les représentations hexadécimales composées d'un chiffre et d'une lettre ou d'une lettre seule doivent être précédées des guillemets.

```
POUR DEC.A.HEX :NB
RT TRANSFORME :NB 10 16
FIN

POUR HEX.A.DEC :NB
RT TRANSFORME :NB 16 10
FIN
```

Pour la logique en programmation et la mise au point

Les procédures définies dans cette section vous permettent :

- d'ajouter des commentaires dans un programme (COMMENTAIRE) ;
- de faire en sorte que chaque élément d'une liste réponde à une commande (DONNE) ;
- de répéter des instructions jusqu'à ce que vous stoppiez la procédure (INFINI) ;
- de répéter des instructions jusqu'à ce qu'une condition soit fausse (TANTQUE) ;
- de trier une liste de mots et d'en faire une liste s'inscrivant de gauche à droite (TRI).

COMMENTAIRE

COMMENTAIRE vous permet d'ajouter des commentaires dans un programme sous la forme :

```
; <CECI EST UN COMMENTAIRE>

POUR ; :COMMENTAIRE
FIN
```

DONNE

DONNE fait en sorte que chaque élément d'une liste réponde à une commande.

```
POUR DONNE :COMMANDE :LISTE
SI VIDEP :LISTE <STOP>
EXECUTE LISTE :COMMANDE MOT " " PREMIER !
:LISTE
DONNE :COMMANDE SP :LISTE
FIN
```

INFINI

INFINI répète un groupe d'instructions jusqu'à ce que vous pressiez [Ó] - [ESC] ou que vous éteigniez votre ordinateur.

```
POUR INFINI :LISTEINSTRUCTIONS
EXECUTE :LISTEINSTRUCTIONS
INFINI :LISTEINSTRUCTIONS
FIN
```

TANTQUE

TANTQUE répète un groupe d'instructions jusqu'à ce que :CONDITION soit fausse.

```
POUR TANTQUE :CONDITION :LISTEINSTRUCTI!
ONS
SI EXECUTE :CONDITION <EXECUTE :LISTEIN!
STRUCTIONS><STOP>
TANTQUE :CONDITION :LISTEINSTRUCTIONS
FIN
```

TRI

TRI retourne une liste de mots en ordre alphabétique.

```
POUR TRI :ARG :LISTE
SI VIDEP :ARG <RT :LISTE>
RELIE "LISTE INSERE PREMIER :ARG :LISTE
RT TRI SP :ARG :LISTE
FIN
```

```
POUR INSERE :A :L
SI VIDEP :L<RT (LISTE :A)>
SI AVANTP :A PREMIER :L <RT MP :A :L>
RT MP PREMIER :L INSERE :A SP :L
FIN
```

Essayez ceci :

```
RELIE "TRILISTE TRI <A D E F T C Z><>
```

```
EC :TRILISTE
A C D E F T Z
```

Puis tapez :

```
RELIE "TRILISTE TRI <ZOO BAR BOA> :TRI!
LISTE
EC :TRILISTE
A BAR BOA C D E F T Z ZOO
```

Pour les jeunes utilisateurs

Les procédures décrites dans cette section vous permettent :

- de définir une procédure alors que vous la faites exécuter ligne par ligne (APPRENDRE) ;
- de déplacer la Tortue sur l'écran en appuyant sur une touche (DEPLACE).

APPRENDRE

APPRENDRE vous permet de définir une procédure alors que vous la faites exécuter ligne par ligne. En tapant FIN, vous terminez la procédure. En tapant EFFACE, vous effacez la ligne précédant cette commande. Cette méthode est particulièrement utile pour travailler avec les enfants.

```
POUR APPRENDRE
LOCALE "CETTELIGNE
DEFINIS "PROGRAMME <<>>
VE
RAMENELIGNES
NOMMELIGNES
FIN
```

```
POUR RAMENELIGNES
TAPE "??
RELIE "CETTELIGNE LL
SI :CETTELIGNE = <FIN><STOP>
SI :CETTELIGNE = <EFFACE><DETRUIS><SI (!
 PREMIER :CETTELIGNE) = "POUR<><EXEC !
 .CONTENU>>
RAMENELIGNES
FIN

POUR DETRUIS
DEFINIS "PROGRAMME SD TEXTE "PROGRAMME
VE
EXECUTE <PROGRAMME>
FIN

POUR EXEC.CONTENU
ATTRAPE "ERREUR <EXECUTE :CETTELIGNE :C!
ONTENU STOP>
EC PREMIER SP ERREUR
FIN

POUR CONTENU
DEFINIS "PROGRAMME MD :CETTELIGNE TEXTE!
 "PROGRAMME
FIN

POUR NOMMELIGNES
LOCALE "NOM
EC <COMMENT DOIS!-JE L'APPELER?>
RELIE "NOM LL
SI VIDEP :NOM <EFFACE "PROGRAMME STOP>
SI DEFINIEP PREMIER :NOM <ESSAI><COPIE!
R>
FIN

POUR ESSAI
EC PHRASE PREMIER :NOM <DEJA DEFINIE>
EC <>
NOMMELIGNES
FIN

POUR COPIER
DEFINIS PREMIER :NOM TEXTE "PROGRAMME
EC PHRASE PREMIER :NOM <DEFINIE>
EFFACE "PROGRAMME
FIN
```

DEPLACE

DEPLACE vous permet de déplacer la Tortue sur l'écran en appuyant sur une touche.

```
POUR DEPLACE
SI TOUCHEP<ECOUTE>
AV 1
DEPLACE
FIN

POUR ECOUTE
RELIE "REPONSE LISCAR
SI :REPONSE = "S<RENVOIE "NIVEAUSUP>
SI :REPONSE = "D<DR 10>
SI :REPONSE = "G<GA 10>
FIN
```

Fichiers de départ

Cette annexe décrit la possibilité qu'a Logo, à la mise en route, de ramener automatiquement un fichier appelé DEPART dans votre espace de travail. Il ne peut y avoir qu'un seul fichier DEPART ; ce dernier peut toutefois contenir des commandes qui permettent de ramener d'autres fichiers. La disquette comportant le fichier DEPART doit se trouver dans l'unité de disquette 1 lorsque vous enfoncez ↵ après l'affichage du tout premier message.

Créer un fichier DEPART

Avant qu'une procédure ne soit ajoutée dans le fichier DEPART, celle-ci doit d'abord se trouver dans votre espace de travail. Pour ce faire, il vous faut taper la procédure ou la ramener d'un autre fichier. Par exemple, vous pourriez transférer une procédure du fichier OUTILS au fichier DEPART. Pour vérifier le contenu de votre espace de travail, utilisez la primitive IMTS.

Logo affiche les procédures que vous venez d'ajouter, que ce soit en les tapant au clavier ou en les ramenant d'un fichier, de même que celles qui se trouvaient dans votre espace de travail. A ce stade, vous pouvez sauvegarder un nouveau fichier appelé DEPART.

Toutefois, si certaines procédures sont enterrées lorsque vous ramenez un fichier, leur nom n'apparaîtra pas à l'écran lorsque vous utiliserez IMTS et il vous sera impossible de les sauvegarder ou de les effacer. C'est pourquoi les commandes SAUVE, EFTOUT et EFPS ne sauvegardent ni n'effacent ces procédures (c'est d'ailleurs la raison pour laquelle ces dernières sont enterrées). Ces commandes n'agissent donc que sur des procédures déterrées.

Utilisez l'instruction qui suit pour voir s'afficher les noms des procédures déterrées et enterrées.

```
DETERRETOUT
IMTS
```

Pour effacer ces procédures, employez la primitive EFFACE suivie d'une liste complète des noms de procédures que vous voulez effacer. Pour sauvegarder les procédures, utilisez la primitive SAUVEL suivie d'une liste complète des noms des procédures que vous voulez sauvegarder. Seules ces procédures seront mises en réserve, qu'elles soient enterrées ou non. Les variables déterrées seront également sauvegardées ; aussi, vérifiez le contenu de votre espace de travail à l'aide de IMNS avant d'utiliser SAUVEL.

Avertissement

Si vous avez déjà un fichier DEPART et que vous voulez en créer un autre pour le remplacer, vous risquez de détruire des procédures utiles. Même si c'est là votre intention, il se peut que vous ayez besoin des procédures que vous aurez effacées (pour un utilisateur qui essaie Logo pour la première fois, par exemple).

Aussi, il est recommandé de sauvegarder le fichier DEPART sur une disquette sous le nom DEPART0. Utilisez la commande RENOMME pour changer le nom du fichier. Tapez :

```
RENOMME "DEPART "DEPART0
```

Puis tapez :

```
SAUVE "DEPART
```

Les procédures seront ainsi sauvegardées puis ramenées dans votre espace de travail lorsque vous enfoncerez la touche ↵ après l'affichage du tout premier message.

Variable DEPART

Logo dispose d'une variable spéciale appelée variable DEPART. Tous les fichiers, de même que le fichier DEPART, peuvent contenir une telle variable. Lorsque vous ramenez un fichier DEPART, Logo vérifie d'abord s'il existe une variable DEPART. S'il y en a une, Logo exécute son contenu qui doit toujours être une liste.

Lorsque vous ramenez le fichier DEPART dans votre espace de travail, tapez :

```
RELIE "DEPART <EC <BONJOUR>>
```

Logo sauvegarde le contenu de la variable lorsque vous mettez le fichier DEPART en réserve. A la mise en route de Logo, vous verrez apparaître BONJOUR à l'écran avant que le message BIENVENUE A LOGO ne s'affiche.

Il est plus simple d'utiliser la commande EDFICHIER pour éditer un fichier et lui ajouter une variable telle une variable DEPART. Pour ce faire, tapez :

```
EDFICHIER "DEPART
```

Le contenu du fichier apparaîtra dans l'éditeur Logo. Placez le curseur à la fin du fichier, là où se trouvent les variables, puis ajoutez une ligne comme suit :

```
RELIE "DEPART <BIENVENUE>
```

Puis replacez le curseur dans la section où se trouvent les procédures et tapez :

```
POUR BIENVENUE
LOCALE "REPONSE
EC <Bonjour Marie !>
TAPE <Comment vas-tu aujourd'hui?>
RELIE "REPONSE LM
SI MEMBREP :REPONSE <BIEN OK EXCELLENT>!
 <EC <Très heureux de te l'entendre dir!
e.> STOP>
EC <J'espère que Logo t'aidera à retrou!
ver ta bonne humeur.>
FIN
```

En bref, Logo vérifie s'il y a un fichier DEPART sur la disquette placée dans l'unité de disquette 1. S'il en trouve un, Logo vérifie ensuite si le fichier contient une variable DEPART, puis exécute son contenu.

Espace mémoire

Les procédures et les variables prennent de la place ; il faut encore plus d'espace pour exécuter les procédures.

En général, vous n'avez pas à vous préoccuper de l'économie d'espace. Vous devriez plutôt tenter d'écrire des procédures aussi claires et aussi élégantes que possible. Toutefois, il faut vous rappeler que l'espace mémoire est limité. Vous serez peut-être intéressé de voir comment Logo gère son espace mémoire.

Distribution de l'espace

En Logo, l'espace est distribué en **nœuds** ; chacun a une longueur de cinq octets. Les objets et les procédures Logo sont composés de nœuds. Un mot Logo n'est emmagasiné qu'une fois. Par la suite, l'utilisation de ce mot se fait au moyen de pointeurs vers celui-ci.

Logo distribue les nœuds de la façon suivante :

- Un mot littéral utilise un nœud pour chaque deux caractères.
- Le nom d'une variable et le nom d'une procédure utilisent chacun trois nœuds sans compter la longueur du nom lui-même.
- Une liste de propriétés utilise trois nœuds, et deux nœuds pour chacune des propriétés sans compter la longueur de la liste de propriétés elle-même.
- Un nombre, qu'il soit entier ou décimal, utilise un nœud.
- Une liste occupe un nœud pour chaque élément, plus la longueur de l'élément lui-même.

Le fonctionnement interne de Logo utilise également des nœuds. L'interpréteur Logo sait s'il y a des nœuds libres qui peuvent être utilisés. Lorsqu'il n'y a plus de nœuds libres, Logo met automatiquement en marche son récupérateur de mémoire ; ce dernier cherche parmi tous les nœuds ceux qui ne sont pas utilisés.

Exemple :

```
RELIE "NOMBRE 7
RELIE "NOMBRE 90
```

En exécutant RELIE "NOMBRE 7, Logo assigne à NOMBRE un nœud qui contient la valeur 7. Après avoir exécuté RELIE "NOMBRE 90, Logo peut réutiliser les nœuds qui contenaient la valeur 7, puis les récupérer comme nœuds libres à la prochaine exécution du récupérateur de mémoire. Ce récupérateur entre automatiquement en fonction lorsque c'est nécessaire, mais la commande Logo RECYCLE permet son exécution à tout moment.

La commande RECYCLE est décrite au chapitre 14.

L'opération NŒUDS retourne le nombre de nœuds libres ; toutefois, si vous désirez connaître la quantité réelle d'espace disponible, il est recommandé d'utiliser l'instruction suivante :

```
RECYCLE ECRIS NOEUDS
1259
```

Suggestions pour économiser l'espace

Si vous croyez manquer d'espace, il est recommandé de réécrire vos programmes de sorte qu'ils utilisent moins de nœuds. Voici quelques conseils pratiques :

- Remplacer les sections répétitives du programme par des procédures.
- Vous pouvez économiser de l'espace en évitant de créer des mots nouveaux. Les noms de variables d'une procédure peuvent être les mêmes que les noms de variables d'autres procédures. Les noms des procédures et des primitives peuvent également servir de noms de variables.
- Rappelez-vous que c'est une mauvaise habitude d'utiliser des mots abrégés et obscurs dans vos procédures. Cette méthode vous permet peut-être d'économiser l'espace, mais la lecture des procédures est ainsi plus difficile.

Interprétation

Cette annexe vous explique comment Logo **interprète** les lignes. Lorsque vous tapez une ligne, Logo reconnaît les caractères tels les mots et les listes, et construit une liste qui est la représentation interne d'une ligne Logo. Pour voir l'effet de l'interprétation, tapez une ligne à l'intérieur d'une définition de procédure à l'aide de la commande POUR et voyez sa transcription par l'éditeur Logo.

Délimiteurs

Un mot est habituellement délimité par des espaces. Il y a donc un espace avant le mot et un espace après le mot ; ce dernier est ainsi séparé du reste de la ligne. Il existe quelques autres caractères délimiteurs :

```
<>()=+-*
```

Il n'est pas nécessaire de taper un espace entre un mot et l'un de ces caractères. Par exemple, pour voir de quelle façon cette ligne est interprétée :

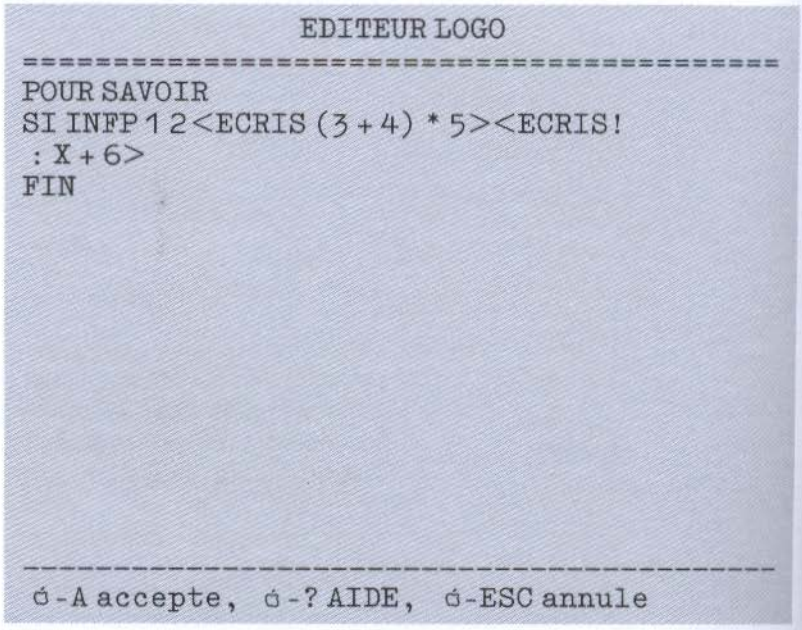
```
SI INFP 1 2<ECRIS(3+4)*5><ECRIS:X+6>
```

tapez :

```
?POUR SAVOIR
>SI INFP 1 2<ECRIS(3+4)*5><ECRIS:X+6>
>FIN

?ED "SAVOIR
```

A l'écran, vous trouverez ceci :



Pour traiter l'un des caractères mentionnés ci-dessus comme un caractère alphabétique normal, il faut le faire précéder d'un point d'exclamation "!". Par exemple :

```
?ECRIS "PASSE!-TEMPS
PASSE-TEMPS
?ECRIS "SAN! FRANCISCO
SAN FRANCISCO
```

Procédures de forme infixée

Les caractères qui suivent sont des procédures de forme infixée. Elles sont traitées comme des procédures à deux données, mais le nom de la procédure est écrit entre ses deux données.

+ - * / =

Crochets et parenthèses

Le crochet ouvrant, <, et le crochet fermant, >, indiquent le début et la fin d'une liste ou d'une sous-liste.

Les parenthèses regroupent les éléments à volonté plutôt que de laisser à Logo le soin de le faire à sa façon. Elles permettent aussi de faire varier le nombre de données dont certaines primitives ont besoin.

Si la fin d'une ligne Logo est atteinte (après l'usage de la touche ↵) et que des crochets ou des parenthèses n'ont pas été fermés, toutes les sous-listes ou expressions sont fermées. Par exemple :

```
?REPETE 4 <ECRIS <CECI <EST <UN <TEST
CECI <EST <UN <TEST>>>
CECI <EST <UN <TEST>>>
CECI <EST <UN <TEST>>>
CECI <EST <UN <TEST>>>
```

Si Logo rencontre un crochet fermant sans trouver le crochet ouvrant correspondant, il arrête l'exécution de la ligne ou de la procédure. Par exemple :

```
?>ECRIS "ABC
?
```

Guillemets et délimiteurs

Habituellement, vous devez faire usage du point d'exclamation devant les caractères <, >, (,), +, −, *, =, et le point d'exclamation lui-même. Toutefois, le premier caractère qui suit les guillemets (") ne requiert pas de point d'exclamation pour être considéré comme tel. Ainsi :

```
?ECRIS " *
*
```

Si un délimiteur occupe une position autre que cette première place, il doit être précédé du point d'exclamation. Par exemple :

```
?ECRIS " * * * *
MANQUE DE DONNEES POUR *
```

Les crochets sont la seule exception à cette règle. Même après des guillemets, les crochets doivent être précédés du point d'exclamation. Par exemple :

```
?ECRIS "<
NE SAIS QUE FAIRE AVEC <>
?ECRIS "!<
<
```

Le signe moins

La façon dont le signe moins "–" est interprété n'est pas évidente. Le problème vient du fait que ce seul caractère cumule trois fonctions :

- Lié à un nombre, il indique que celui-ci est négatif comme dans –3.
- Lié à une donnée, il est appelé le moins unaire et retourne l'inverse additif de la donnée, comme dans –COORX et – :DISTANCE.
- Dans une expression à deux données, il retourne leur différence, comme dans 7 – 3 ou COORX – COORY.

L'interpréteur tend à dénouer cette ambiguïté et détermine de quelle fonction il s'agit en se référant aux règles suivantes :

1. Si "–" précède immédiatement un nombre et suit un délimiteur (incluant l'espace), sauf la parenthèse ")", le nombre est interprété comme une valeur négative. Cela explique les faits suivants :

`ECRIS 3 * -1`	s'interprète comme 3 fois moins 1
`ECRIS 3*-4`	s'interprète comme 3 fois moins 4
`PREMIER <- # 3 4>`	retourne –
`PREMIER <-3 4>`	retourne -3

2. Si "–" est précédé d'une expression numérique, il agit comme une procédure de forme infixée. Par exemple :

```
EC 3-4                                  rend -1
EC COORX - COORY
```

Les exemples suivants sont interprétés de la même façon :

```
RELIE "A PH COORX -COORY 3
RELIE "A PH COORX - COORY 3
RELIE "A PH COORX- COORY 3
```

3. Si "–" n'est pas précédé d'une expression numérique, il agit comme un moins unaire.

```
EC -COORX
EC -(3+4)
```

Codes caractères ASCII

Cette annexe présente un tableau du code ASCII *(American Standard Code for Information Interchange)* en notation décimale pour tous les caractères Logo. Notez que les caractères peuvent être :

- normaux (caractères blancs sur fond noir) ;
- vidéo inverse (caractères noirs sur fond blanc).

Le tableau F-1 présente le code ASCII pour les caractères en mode normal.

Le tableau F-2 présente le code ASCII pour les caractères en mode vidéo inverse.

Pour transformer un caractère normal en un caractère en vidéo inverse, utilisez la procédure suivante :

```
POUR INVERSER :CAR
SI SUPP (ASCII :CAR) 127<RT :CAR>
SI OU INFP (ASCII :CAR) 64 ET SUPP (ASC!
II :CAR) 96 INFP (ASCII :CAR) 128<RT C!
AR 128 + ASCII :CAR><RT CAR 64 + ASCII!
 :CAR>
FIN
```

Tableau F-1. *Code ASCII pour les caractères en mode normal*

Code	Car	Code	Car	Code	Car	Code	Car
0	@	32	espace	64	@	96	`
1	A	33	!	65	A	97	a
2	B	34	"	66	B	98	b
3	C	35	#	67	C	99	c
4	D	36	$	68	D	100	d
5	E	37	%	69	E	101	e
6	F	38	&	70	F	102	f
7	G	39	'	71	G	103	g
8	H	40	(	72	H	104	h
9	I	41	)	73	I	105	i
10	J	42	*	74	J	106	j
11	K	43	+	75	K	107	k
12	L	44	,	76	L	108	l
13	RETURN	45	–	77	M	109	m
14	N	46	.	78	N	110	n
15	O	47	/	79	O	111	o
16	P	48	0	80	P	112	p
17	Q	49	1	81	Q	113	q
18	R	50	2	82	R	114	r
19	S	51	3	83	S	115	s
20	T	52	4	84	T	116	t
21	U	53	5	85	U	117	u
22	V	54	6	86	V	118	v
23	W	55	7	87	W	119	w
24	X	56	8	88	X	120	x
25	Y	57	9	89	Y	121	y
26	Z	58	:	90	Z	122	z
27	[	59	;	91	[	123	{
28	/	60	{	92	/	124	,
29	]	61	=	93	]	125	}
30	^	62	}	94	^	126	~
31	–	63	?	95	–	127	damier

Tableau F-2. *Code ASCII pour les caractères en mode vidéo inverse*

Code	Car	Code	Car	Code	Symboles Souris	Code	Car
128	@	160	espace	192		224	
129	A	161	!	193		225	a
130	B	162	"	194		226	b
131	C	163	#	195		227	c
132	D	164	$	196		228	d
133	E	165	%	197		229	e
134	F	166	&	198		230	f
135	G	167	'	199		231	g
136	H	168	(	200		232	h
137	I	169	)	201		233	i
138	J	170	*	202		234	j
139	K	171	+	203		235	k
140	L	172	,	204		236	l
141	M	173	–	205		237	m
142	N	174	.	206		238	n
143	O	175	/	207		239	o
144	P	176	0	208		240	p
145	Q	177	1	209		241	q
146	R	178	2	210		242	r
147	S	179	3	211		243	s
148	T	180	4	212		244	t
149	U	181	5	213		245	u
150	V	182	6	214		246	v
151	W	183	7	215		247	w
152	X	184	8	216		248	x
153	Y	185	9	217		249	y
154	Z	186	:	218		250	z
155	[	187	;	219		251	{
156	\	188	{	220		252	,
157	]	189	=	221		253	}
158	^	190	}	222		254	~
159	–	191	?	223		255	damier

Résumé des primitives Logo

Les parenthèses entourant une donnée indiquent que cette dernière est facultative. Le signe (*) signifie qu'une primitive peut recevoir un nombre quelconque de données. Si vous attribuez à une telle primitive un nombre de données différent de celui qui est indiqué, il faut mettre toute l'expression entre parenthèses. Les prédicats qui retournent VRAI sous certaines conditions retournent FAUX si ces conditions ne sont pas satisfaites.

AIDE *nom*	Affiche les données requises pour la primitive ou la procédure indiquée par *nom.*
ANNULEPROP *nom prop*	Annule la propriété *prop* de la liste de propriétés *nom.*
.APPELLE *adr*	Appelle la sous-routine en langage machine à l'adresse *adr.*
ARCTAN *nombre*	Retourne l'arc tangente de *nombre* en degrés.
ARRONDIS *nombre*	Retourne *nombre* arrondi à l'entier le plus proche.
ASCII *car*	Retourne le code ASCII de *car.*
ATTENDS *entier*	Arrête l'exécution pendant *entier* soixantièmes de seconde.
ATTRAPE *nom liste*	Exécute *liste* ; rend le contrôle à RENVOIE *nom.*
.AUXDEPOSE *adr octet*	Stocke la valeur *octet* à l'adresse *adr* du bloc mémoire auxiliaire.

.AUXEXAMINE *adr*	Retourne la valeur stockée à l'adresse *adr* du bloc mémoire auxiliaire.
AVANCE, AV *distance*	Déplace la Tortue de *distance* pas vers l'avant.
AVANTP *mot1 mot2*	Retourne VRAI si *mot1* vient avant *mot2* selon le code ASCII.
BARRIERE	Enferme la Tortue dans les limites de l'écran.
BC	Abaisse le crayon.
BOUTONP *nomanette*	Retourne VRAI si le bouton *nomanette* est enfoncé.
.BRAMENE *nomfichier adr*	Ramène un fichier en langage machine dans la mémoire à l'adresse *adr*.
.BSAUVE *nomfichier adr entier*	Sauvegarde une section de la mémoire, commençant à l'adresse *adr* pour *entier octet*, dans un fichier appelé *nomfichier*.
CACHETORTUE, CT	Rend la Tortue invisible.
CAP	Retourne le cap (la direction) de la Tortue en degrés.
CAR *entier*	Retourne le caractère dont le code ASCII est *entier*.
CC	Retourne un nombre représentant la couleur du crayon.
CHOSE *nom*	Retourne la valeur de *nom*.
CO	Reprend l'exécution d'une procédure après une pause.
COMPTE *obj*	Retourne le nombre d'éléments contenus dans sa donnée.
.CONTENU	Retourne une liste composée de tous les noms, des noms de procédures et des autres mots contenus dans l'espace de travail.

CONVERTIS *mot*	Retourne *mot* mis en liste après en avoir effectué l'analyse syntaxique.
COORX	Retourne l'abscisse de la position de la Tortue.
COORY	Retourne l'ordonnée de la position de la Tortue.
COPIE *fichier*	Envoie une copie de ce qui est affiché à l'écran à un fichier ou à un périphérique.
COPIEDEF *nom nouvnom*	Copie la définition de *nom* dans *nouvnom.*
COS *degrés*	Retourne le cosinus de *degrés.*
CRAYON	Retourne l'état du crayon (BC, LC, IC ou GC).
CREEINDEX *nomfichier*	Crée un sous-index portant le nom du dernier élément de *nomfichier.*
CURSEUR	Retourne la position du curseur.
DEFINIEP *nom*	Retourne VRAI si *nom* est le nom d'une procédure.
DEFINIS *nom liste*	Donne *liste* comme définition de *nom.*
.DEPOSE *adr octet*	Stocke la valeur *octet* à l'adresse *adr.*
DERNIER, DE *obj*	Retourne le dernier élément de sa donnée.
DETERRE *nom(liste)*	Déterre les procédures contenues dans *nom(liste).*
DETERRENOM *nom(liste)*	Déterre les variables contenues dans *nom(liste).*
DETERRETOUT	Déterre toutes les procédures et toutes les variables enterrées dans l'espace de travail.
DIFFERENCE *nombre1 nombre2*	Retourne le résultat obtenu en soustrayant *nombre2* de *nombre1.*

DIV *entier1 entier2*	Retourne la partie entière de *entier1* divisé par *entier2*.
DPROP *nom prop obj*	Donne à *nom* la propriété *prop* avec la valeur *obj*.
DROITE, DR *degrés*	Fait tourner la Tortue de *degrés*, vers la droite, c'est-à-dire dans le sens des aiguilles d'une montre.
.ECHELLE	Retourne le rapport d'échelle de l'écran.
ECRAND	Divise l'écran : le haut pour le graphique, le bas pour le texte. Equivaut à CONTROL - S .
ECRANG	Consacre tout l'écran au graphique. Equivaut à CONTROL - L .
ECRANT	Consacre tout l'écran au texte. Equivaut à CONTROL - T .
*ECRIS, EC *obj*	Ecrit *obj* suivi d'un retour de chariot (élimine les crochets des listes).
EDFICHIER, EDF *nomfichier*	Place le contenu du fichier indiqué par *nomfichier* dans l'éditeur Logo.
EDITE, ED *(nom(liste))*	Démarre l'éditeur Logo qui contient alors les procédures nommées.
EDN *nom(liste)*	Ramène l'éditeur Logo qui contient alors les variables données.
EDNS	Ramène l'éditeur Logo et toutes les variables contenues dans l'espace de travail.
EFFACE, EF *nom(liste)*	Efface les procédures nommées.
EFFICHIER, EFF *nomfichier*	Efface de la disquette le fichier indiqué par *nomfichier*.
EFN *nom(liste)*	Efface les variables nommées.

EFNS	Efface toutes les variables contenues dans l'espace de travail.
EFPROPS	Efface toutes les propriétés contenues dans l'espace de travail.
EFPS	Efface toutes les procédures contenues dans l'espace de travail.
EFTOUT	Efface le contenu de l'espace de travail.
EGALP *obj1 obj2*	Retourne VRAI si ses données sont égales.
ELEM *entier obj*	Retourne l'élément dont la position dans *obj* est *entier*.
ENROULE	Fait s'enrouler le champ de la Tortue autour des côtés de l'écran.
ENTERRE *nom(liste)*	Enterre toutes les procédures contenues dans *nom(liste)*.
ENTERRENOM *nom(liste)*	Enterre les variables contenues dans *nom(liste)*.
ENTERRETOUT	Enterre toutes les procédures et toutes les variables contenues dans l'espace de travail.
ENTIER *nombre*	Retourne la partie entière de *nombre*.
ERREUR	Retourne une liste de quatre éléments qui renseigne sur l'erreur la plus récente.
*ET *préd1 préd2*	Retourne VRAI si toutes ses données sont vraies.
ETIQUETTE *nom*	Marque la ligne pour les besoins de VA.
.EXAMINE *adr*	Retourne l'octet stocké à l'adresse *adr*.
EXECUTE *liste*	Exécute *liste* ; retourne ce que *liste* retourne.

FCAP *degrés*	Donne au cap de la Tortue la valeur indiquée par *degrés*.
FCC *nocouleur*	Donne au crayon la valeur indiquée par *nocouleur*.
FCURSEUR *<nocolonne noligne>*	Place le curseur à la position indiquée par *<nocolonne noligne>*.
FEC *fichier*	Fixe la destination des données de ECRIS, TAPE et MONTRE.
.FECHELLE *nombre*	Donne la valeur indiquée par *nombre* au rapport d'échelle.
FENETRE	Supprime les limites du champ de la Tortue.
FERME *fichier*	Ferme le fichier ou le périphérique ouvert à ce moment.
FERMETOUT	Ferme tous les fichiers et les périphériques ouverts à ce moment.
FFOND *nocouleur*	Donne au fond la couleur indiquée par *nocouleur*.
FICHIERP *nomfichier*	Retourne VRAI si le fichier indiqué par *nomfichier* existe.
FLARGEUR *largeur*	Fixe la largeur de l'écran à 40 ou 80 colonnes.
FLIS *fichier*	Prépare le fichier à partir duquel les données de LISCAR, LISCARS, LISLISTE et LISMOT seront lues.
FOND	Retourne le numéro de la couleur du fond.
FORMAT *nombre champ décimale*	Retourne *nombre* dans un nombre d'espaces indiqué par *champ* avec une précision de *décimale* après le point.
FPOS *<coorx coory>*	Place la Tortue aux coordonnées indiquées par *<coorx coory>*.
FPOSECRIT *entier*	Fixe la position où s'effectue l'écriture dans un fichier.

FPOSLECT *entier*	Fixe la position où s'effectue la lecture dans un fichier.
FPREFIXE *préfixe*	Fixe le préfixe ProDOS.
FX *coorx*	Déplace la Tortue horizontalement jusqu'au point d'abscisse *coorx*.
FY *coory*	Déplace la Tortue verticalement jusqu'au point d'ordonnée *coory*.
GAUCHE, GA *degrés*	Fait tourner la Tortue de *degrés* vers la gauche, c'est-à-dire dans le sens inverse des aiguilles d'une montre.
GC	Transforme le crayon en gomme à effacer.
HASARD *entier*	Retourne un entier aléatoire non négatif inférieur à *entier*.
IC	Transforme le crayon en inverseur de couleurs.
IM *nom(liste)*	Affiche les définitions des procédures nommées.
IMFICHIER, IMF *nomfichier*	Affiche le contenu du fichier indiqué par *nomfichier*.
IMIMAGE *entier*	Imprime sur papier le contenu de l'écran graphique. L'imprimante doit être connectée à la prise ou au logement indiqué par *entier*.
IMINDEX	Affiche le nom de tous les fichiers contenus dans l'index ainsi que le nombre de blocs qu'ils utilisent.
IMN *nom(liste)*	Affiche le nom et la valeur des variables nommées.
IMNS	Affiche le nom et la valeur de toutes les variables déterrées contenues dans l'espace de travail.

IMPROPS	Affiche toutes les listes de propriétés contenues dans l'espace de travail.
IMPS	Affiche la définition de toutes les procédures déterrées contenues dans l'espace de travail.
IMT *nom(liste)*	Affiche la ligne titre des procédures nommées.
IMTOUT	Affiche la définition de toutes les procédures et toutes les variables contenues dans l'espace de travail.
IMTS	Affiche la ligne titre de toutes les procédures contenues dans l'espace de travail.
INFP *nombre1 nombre2*	Retourne VRAI si *nombre1* est inférieur à *nombre2*.
LARGEUR	Retourne la largeur d'écran, soit 40 ou 80 caractères.
LC	Lève le crayon.
LISCAR	Retourne le caractère lu à partir d'un fichier ou d'un périphérique (le clavier implicitement). Attend qu'une donnée soit tapée, s'il y a lieu.
LISCARS *entier*	Retourne *entier* caractères lus à partir d'un fichier ou d'un périphérique (le clavier implicitement). Attend qu'une donnée soit tapée, s'il y a lieu.
LISLISTE, LL	Retourne la ligne lue à partir d'un fichier ou d'un périphérique (le clavier implicitement). Attend qu'une donnée soit tapée, s'il y a lieu.
LISMOT, LM	Retourne la ligne lue par le périphérique (le clavier implicitement) après un retour de chariot.
*LISTE *obj1 obj2*	Retourne la liste formée de ses données en lui conservant son ordre.

LISTEP *obj*	Retourne VRAI si *obj* est une liste.
LOCALE *nom(liste)*	Rend la variable *nom(liste)* locale.
LONGUEURF *nomfichier*	Retourne la longueur en octets du fichier indiqué par *nomfichier.*
MAJUSCULE *mot*	Retourne *mot* en majuscules.
MANETTE *nomanette*	Retourne un nombre représentant la rotation du cadran de la manette.
MEMBRE *obj1 obj2*	Retourne la partie de *obj2* qui commence par *obj1.*
MEMBREP *obj1 obj2*	Retourne VRAI si *obj1* est un élément de *obj2.*
METSDERNIER, MD *obj liste*	Retourne la liste formée en plaçant *obj* comme dernier élément de *liste.*
METSPREMIER, MP *obj liste*	Retourne une liste formée en plaçant *obj* comme premier élément de *liste.*
MINUSCULE, *mot*	Retourne *mot* en minuscules.
MONTRE *obj*	Ecrit *obj* suivi d'un retour de chariot (affiche les crochets des listes).
MONTRETORTUE, MT	Rend la Tortue visible.
*MOT *mot1 mot2*	Retourne un mot constitué de ses données.
MOTP *obj*	Retourne VRAI si *obj* est un mot.
NETTOIE	Efface l'écran graphique sans affecter la Tortue.
NŒUDS	Retourne le nombre de nœuds libres.
NOMBREP *obj*	Retourne VRAI si *obj* est un nombre.
NOMME *obj nom*	Donne à *obj* la valeur de *nom.*
NOMP *mot*	Retourne VRAI si *mot* a une valeur.

NOMSDISQUES, ND	Donne la dénomination de la disquette utilisée.
NON *préd*	Retourne VRAI si *préd* est FAUX.
NONCOPIE	Ferme un fichier copies.
NONPAP *nom(liste)*	Stoppe le processus ayant débuté avec PAP.
NONTRACE *nom(liste)*	Stoppe le processus ayant débuté avec TRACE.
ORIGINE	Place la Tortue à <0 0> et fixe le cap à 0.
*OU *préd1 préd2*	Retourne VRAI si une de ses données est vraie.
OUVERTS	Retourne une liste des fichiers ouverts.
OUVRE *fichier*	Ouvre un fichier de sorte qu'on puisse y lire ou y écrire.
PAP *nom(liste)*	Demande à la procédure de s'exécuter une ligne à la fois.
PAUSE	Introduit une pause dans la procédure.
PEINS	Remplit, de la couleur actuelle du crayon, la forme dans laquelle la Tortue se trouve. Si la Tortue ne se trouve pas à l'intérieur d'une forme délimitée, le fond se colore.
*PHRASE, PH *obj1 obj2*	Retourne une liste composée de ses données.
PLISTE *nom*	Retourne la liste de propriétés de *nom.*
POINT <*coorx coory*>	Place un point aux coordonnées indiquées.
POINTECRIT	Retourne le fichier ouvert dans lequel on peut écrire.
POINTELECT	Retourne le fichier ouvert pour la lecture.
POINTP <*coorx coory*>	Retourne VRAI s'il y a un point aux coordonnées indiquées.

POSECRIT	Retourne la position dans le fichier où l'écriture peut s'effectuer.
POSITION, POS	Retourne la position de la Tortue sous la forme de coordonnées.
POSLECT	Retourne la position dans le fichier où la lecture peut s'effectuer.
POUR *nom(donnée(s))*	Commence la définition de la procédure *nom.*
PREFIXE	Retourne le préfixe ProDOS le plus récemment fixé à l'aide de FPREFIXE.
PREMIER, PR *obj*	Retourne le premier élément de sa donnée.
PRIMITIVEP *mot*	Retourne VRAI si *mot* est une primitive.
*PRODUIT *nombre1 nombre2*	Retourne le produit de ses données.
.QUITTE	Abandonne Logo et remet le contrôle à ProDOS.
QUOTIENT *nombre1 nombre2*	Retourne le résultat, sous la forme décimale, de la division de *nombre1* par *nombre2.*
RAMENE *nomfichier*	Ramène le fichier indiqué par *nomfichier* dans l'espace de travail.
RAMENEAIDE *nomfichier*	Ramène le fichier indiqué par *nomfichier* dans la partie de la mémoire contenant l'écran d'aide. Ce dernier peut ainsi apparaître à l'écran lorsque les touches [ó] - [?] sont pressées.
RAMENEIMAGE *nomfichier*	Ramène le dessin contenu dans le fichier indiqué par *nomfichier.*
RC *nombre*	Retourne la racine carrée de *nombre.*
RECULE, RE *distance*	Déplace la Tortue de *distance* pas vers l'arrière.

RECYCLE	Effectue une récupération de la mémoire disponible.
REHASARD	Rend le résultat de HASARD reproductible.
RELIE *nom obj*	Donne la valeur *obj* à la variable *nom.*
RENOMME *nomfichier nouvnomfichier*	Donne à *nomfichier* le *nouvnomfichier* (les deux fichiers doivent être fermés).
RENVOIE *nom*	Transfère le contrôle à la primitive ATTRAPE correspondante.
REPETE *entier liste*	Exécute *liste entier* fois.
RESTE *entier1 entier2*	Retourne le reste de *entier1* divisé par *entier2.*
RETOURNE, RT *obj*	Remet le contrôle à la procédure appelante, avec *obj* comme résultat.
RPROP *nom prop*	Retourne la propriété *prop* de *nom.*
SAUFDERNIER, SD *obj*	Retourne tout *obj* sauf son dernier élément.
SAUFPREMIER, SP *obj*	Retourne tout *obj* sauf son premier élément.
SAUVE *nomfichier*	Sauvegarde le contenu de l'espace de travail dans le fichier indiqué par *nomfichier.*
SAUVEIMAGE *nomfichier*	Sauvegarde le dessin apparaissant à l'écran dans le fichier indiqué par *nomfichier.*
SAUVEL <nom ...> *nomfichier*	Sauvegarde les procédures énumérées ainsi que les variables déterrées dans le fichier indiqué par *nomfichier.*
SI *préd liste1 (liste2)*	Si *préd* est VRAI, exécute *liste1;* sinon, exécute *liste2.*
SIFAUX, SIF *liste*	Exécute *liste* si le plus récent TESTE était FAUX. Si aucun test n'a été effectué, la liste n'est pas exécutée.
SIN *degrés*	Retourne le sinus de *degrés.*

SIVRAI, SIV *liste*	Exécute *liste* si le plus récent TESTE était VRAI. Si aucun test n'a été effectué, la liste n'est pas exécutée.
*SOMME *nombre1 nombre2*	Retourne la somme de ses données.
SON *fréq durée*	Produit un son de *fréq* pendant une *durée* donnée.
STOP	Arrête la procédure et remet le contrôle à la procédure appelante.
SUPP *nombre1 nombre2*	Retourne VRAI si *nombre1* est supérieur à *nombre2*.
*TAPE *obj*	Ecrit *obj*, sans crochets pour les listes.
TESTE *préd*	Vérifie si *préd* est VRAI ou FAUX.
TEXTE *nom*	Retourne la définition de la procédure *nom* sous forme de liste.
TOUCHEP	Retourne VRAI si une touche a été pressée, mais pas encore lue.
TRACE *nom(liste)*	Affiche les instructions une à une au fur et à mesure que la procédure s'exécute.
VA *nom*	Remet le contrôle à ETIQUETTE *nom*.
VE	Efface l'écran, replace la Tortue à <0 0> et fixe son cap à 0.
VERS <*coorx coory*>	Retourne le cap que devrait avoir la Tortue pour être orientée vers les coordonnées indiquées.
VIDEP *obj*	Retourne VRAI si *obj* est la liste vide ou le mot vide.
VISIBLEP	Retourne VRAI si la Tortue est visible.
VT	Vide l'écran de texte.
nombre1 + nombre2	Retourne *nombre1* plus *nombre2*.
nombre1 – nombre2	Retourne *nombre1* moins *nombre2*.
*nombre1 * nombre2*	Retourne *nombre1* multiplié par *nombre2*.
nombre1 / nombre2	Retourne *nombre1* divisé par *nombre2*.
obj1 = obj2	Retourne VRAI si *obj1* est égal à *obj2*.

Utilisation d'une imprimante

Voici quelques indications pouvant vous aider à utiliser une imprimante de façon efficace avec Logo. Si vous obtenez déjà des résultats satisfaisants en employant votre imprimante, ces directives vous sembleront superflues.

Si vous avez des problèmes d'impression, vous devez vérifier trois éléments qui vous aideront à trouver et à corriger ce qui ne va pas :

- le logiciel, c'est-à-dire votre programme ;
- la configuration interne de l'ordinateur, y compris la carte d'interface, ou le point de connexion incorporé ;
- la configuration interne de l'imprimante, ainsi que le fil de raccordement.

Le tableau H-1 décrit certains signes indicateurs d'un mauvais fonctionnement de l'imprimante ainsi que les causes correspondantes.

Tableau H-1. *Problèmes reliés à l'utilisation de l'imprimante et causes probables*

Problème	**Cause probable (voir la section indiquée)**
N'imprime rien du tout	Erreur dans le programme (Le logiciel)
	Installation incorrecte de la carte d'interface, ou configuration interne de l'ordinateur ou de la carte d'interface non conforme (L'ordinateur)

Tableau H-1. *Problèmes reliés à l'utilisation de l'imprimante et causes probables*

Problème	Cause probable (voir la section indiquée)
	Installation incorrecte de l'imprimante ou configuration interne non conforme à celle de l'ordinateur (L'imprimante)
Impression incorrecte	Configuration interne de l'ordinateur ou de la carte d'interface non conforme (L'ordinateur)
	Mauvais fil de raccordement (L'ordinateur)
	Configuration interne de l'imprimante non conforme (L'imprimante)

Après avoir identifié la nature du problème observé, référez-vous à la section de cette annexe qui s'y rapporte ; vous y trouverez des indications supplémentaires et des suggestions pour résoudre le problème.

Si vous suivez toutes les explications et qu'aucune d'elles ne règle le problème, le matériel que vous utilisez peut être défectueux. Dans ce cas, apportez l'ordinateur et l'imprimante chez votre dépositaire pour qu'ils soient vérifiés et réparés, s'il y a lieu.

Le logiciel

Des indications supplémentaires sont fournies au chapitre 16.

Si vous obtenez déjà de bons résultats lorsque vous utilisez votre imprimante avec certains programmes ou langages informatiques autres que Logo, le problème provient vraisemblablement de votre programme Logo. Ce dernier traite toute opération d'entrée et de sortie d'information comme s'il s'agissait d'un fichier. Ainsi, avant de pouvoir transmettre de l'information à l'imprimante, vous devez d'abord ouvrir le logement ou le point de connexion auquel celle-ci est reliée (et que l'on identifie par le numéro de référence interne de ce logement ou point de connexion) ; il faut ensuite désigner ce périphérique comme étant celui auquel vous voulez transmettre de l'information.

En supposant que l'imprimante est reliée au logement 1, le programme suivant enverra le texte à l'imprimante :

```
OUVRE 1 FEC 1
EC <CECI EST UN TEST : >
EC <SI ÇA FONCTIONNE , C'EST GAGNE ! >
FERME 1 FEC <>
```

Ainsi OUVRE 1 ouvre le logement ou point de connexion 1, prêt à être utilisé, alors que FEC 1 fait du périphérique relié à ce logement ou point de connexion celui auquel l'information est transmise. Toute commande, que ce soit ECRIS, TAPE ou MONTRE, apparaissant à la suite de cette première ligne, transmettra automatiquement son information au périphérique choisi, soit, dans ce cas-ci, l'imprimante. La dernière ligne du programme ferme d'abord le logement ou point de connexion auquel l'imprimante est reliée, pour ensuite rétablir l'écran comme périphérique auquel l'information doit dorénavant être transmise.

Note : Si l'imprimante est reliée à un logement autre que 1, vous devez indiquer cet autre numéro de logement dans le programme. Lorsque vous avez fini d'imprimer, vous devez fermer le logement ou point de connexion auquel l'imprimante est reliée puis rétablir l'écran comme périphérique auquel l'information doit être transmise.

Souvenez-vous qu'un maximum de six fichiers peuvent être ouverts à la fois ; *un seul* de ceux-ci peut être un logement ou un point de connexion, puisque l'imprimante est traitée comme un fichier.

L'ordinateur

Reportez-vous au manuel accompagnant votre carte série pour obtenir plus de détails sur sa configuration.

La vérification de votre matériel doit commencer par l'ordinateur et la carte d'interface pour l'imprimante.

Logo traite l'interface de l'imprimante de la même façon que la version 1.1 du Pascal Apple II le fait. Une interface qui ne se conforme pas au protocole Pascal Apple II, telle la carte d'interface parallèle de l'Apple II, ne peut fonctionner avec Logo. Si vous disposez d'une carte d'interface parallèle Apple II, demandez l'aide de votre dépositaire pour la faire fonctionner en utilisant Logo. Si vous avez des questions au sujet d'une autre carte d'interface, vous devriez vous adresser au fabricant de cette carte.

Si vous disposez d'un Apple IIe, assurez-vous que la carte d'interface est bien enfichée dans un des logements de l'ordinateur ; habituellement, il s'agit du logement 1. Si vous disposez d'un Apple IIc, vous devez connecter l'imprimante à la prise marquée d'un 1.

Si vous employez une imprimante à interface série telle l'Imagewriter de Apple, consultez la section "Interfaces série" ; si vous utilisez une imprimante à interface parallèle telle l'imprimante par matrice de points de Apple, consultez la section "Interfaces parallèles".

Interfaces série

Une interface série se définit d'abord selon les caractéristiques suivantes :

- le débit – la vitesse de transmission de l'information mesurée en bauds ;
- la représentation des données – la façon dont l'information est organisée pour la transmission : le nombre de bits par caractère, le mode de parité et le nombre de bits d'arrêt ;
- quelques autres facteurs peuvent affecter le fonctionnement de l'imprimante, selon que :
  - l'information qui s'imprime est aussi affichée à l'écran ;
  - un caractère de changement de ligne s'ajoute automatiquement à la fin de chaque ligne de texte ;
  - le texte transmis se répartit en lignes d'une longueur déterminée.

Lorsque vous allumez l'ordinateur Apple IIc, la prise 1 est automatiquement configurée conformément à la configuration normale de l'imprimante Imagewriter de Apple :

- un débit de 9600 bauds ;
- représentation des données sous forme de caractères de 8-bits, sans parité, avec deux bits d'arrêt ;
- sans interlignage automatique.

Référez-vous au manuel de référence de l'imprimante et de la carte d'interface pour être en mesure d'effectuer les réglages nécessaires et de régler leur configuration respective.

Si vous utilisez un Apple IIe, vous devez configurer votre interface série de la même façon que celle du port série 1 de l'Apple IIc.

Si votre carte d'interface ne peut fonctionner à une vitesse aussi élevée que 9600 bauds, vous devrez la configurer pour qu'elle fonctionne à son débit le plus élevé et rendre la configuration de l'imprimante conforme à celle de la carte d'interface.

Vous pouvez maintenant vérifier le fonctionnement de votre imprimante en faisant exécuter le programme apparaissant dans la section "Le logiciel". Si l'imprimante ne fonctionne toujours pas, passez directement à la section "L'imprimante".

Interfaces parallèles

Si vous disposez d'un Apple IIc, cette section ne vous sera d'aucune utilité.

Si vous utilisez un ordinateur Apple IIe muni d'une carte d'interface parallèle, il est possible d'imprimer à l'aide de Apple Logo II en exécutant les instructions suivantes.

Fixer les commutateurs du boîtier à doubles rangées de connexions de votre carte d'interface tel que recommandé dans votre manuel d'installation. La plupart des imprimantes d'usage courant fonctionneront correctement si la position des commutateurs est telle que présentée ici :

Commutateur :	N01	N02	N03	N04	N05	N06	N07
Position :	OFF	OFF	OFF	ON	ON	OFF	OFF

Le programme suivant fonctionnera à condition que votre carte d'interface parallèle Apple soit installée dans le logement 1 de votre ordinateur Apple.

Chargez Apple Logo II et tapez EDFICHIER "DEPART. Maintenant, définissez les procédures suivantes :

```
POUR CARTE.PARALLELE
.DEPOSE 29694 96
PLACE.OCTETS <234 234 234 234 234> 2965!
6
PLACE.OCTETS <44 193 193 48 251 141 144!
 192 96> 29736
FIN
```

```
POUR PLACE.OCTETS :LISTE :ADRESSE
SI VIDEP :LISTE<STOP>
.DEPOSE :ADRESSE PREMIER :LISTE
PLACE.OCTETS SP :LISTE :ADRESSE + 1
FIN

RELIE "DEPART<CARTE.PARALLELE>
```

Tapez [ó] - [A] pour sauvegarder ces procédures sur votre disquette fichier sous le nom de "DEPART.

Chaque fois que vous chargerez Apple Logo II, les procédures nommées CARTE.PARALLELE et PLACE.OCTETS seront ramenées dans votre espace de travail. De plus, la procédure CARTE.PARALLELE sera exécutée automatiquement, vous permettant ainsi de travailler avec Apple Logo II et votre matériel.

L'imprimante

Assurez-vous que l'imprimante est correctement branchée à la prise murale du réseau électrique et qu'elle est bien reliée à l'ordinateur au moyen du fil de raccordement d'interface. Après avoir réglé les mini-interrupteurs en fonction de la configuration de l'interface utilisée, vous pourrez alors vérifier le fonctionnement de l'imprimante.

Voir le chapitre 16 où se trouve la description de la primitive IMIMAGE.

Il se pourrait que votre imprimante vous permette d'imprimer correctement du texte mais non des dessins lorsque vous utilisez la primitive IMIMAGE. Pour imprimer des dessins au moyen de IMIMAGE, vous devez utiliser une imprimante Imagewriter d'Apple, une imprimante par matrice de points d'Apple ou toute autre imprimante compatible. De plus, vous devez utiliser une carte d'interface telle la "Carte Super Série", par exemple, dont la microprogrammation, c'est-à-dire les programmes contenus dans le ROM, suivent les conventions utilisées par le port série 1 de l'Apple IIc. Si vous disposez d'une imprimante par matrice de points d'Apple ainsi que d'une carte d'interface parallèle, voyez votre dépositaire pour faire fonctionner votre imprimante avec Logo.

Allumez votre ordinateur Apple ainsi que l'imprimante. Essayez d'imprimer du texte en employant le programme de vérification apparaissant à la section "Le logiciel". Si rien ne se passe, vérifiez les éléments suivants :

- Y a-t-il du papier pour l'imprimante ? Le couvercle de l'imprimante est-il placé correctement ? Le ruban de l'imprimante est-il installé correctement ?
- L'imprimante est-elle en circuit et prête à recevoir l'information ? Des imprimantes redeviennent hors circuit lorsque vous remplacez le ruban ou le papier, ou lorsque vous faites avancer le papier. Une fois cela terminé, l'imprimante doit être remise en circuit ou sélectionnée (habituellement en pressant un bouton sur le devant de l'appareil) avant d'être en mesure de continuer à imprimer.
- Les raccordements au réseau électrique et ceux de l'interface ont-ils été correctement branchés ? Le fusible de l'imprimante est-il grillé ?
- La configuration des mini-interrupteurs de la carte d'interface et de l'imprimante est-elle la même ? Reportez-vous au manuel de référence du périphérique concerné pour ce qui est du réglage des mini-interrupteurs.
- La carte d'interface comporte-t-elle un module de configuration ? Est-ce le module approprié ? A-t-il été installé correctement ? Le fil de raccordement aurait-il été placé à l'envers ?
- Avez-vous vérifié chacun des éléments énumérés ci-dessus ? Si l'imprimante ne fonctionne toujours pas, voyez votre dépositaire.

Si l'imprimante hoquète ou n'imprime que des bribes de texte, vérifiez les réglages du débit et de la représentation des données de l'imprimante ainsi que ceux de l'interface. Assurez-vous que le tout est conforme. Vous devriez aussi vous assurer que vous utilisez le bon fil de raccordement.

Si une ligne de texte s'imprime sur une autre, réglez l'imprimante pour qu'elle génère automatiquement un caractère de changement de ligne après chaque ligne. Si le texte s'imprime à interligne double, vous devez régler l'imprimante pour enlever un caractère de changement de ligne.

Vous pouvez aussi obtenir un interligne simple en utilisant la commande Logo

```
.DEPOSE 770 1
```

Pour ramener l'interligne double, vous devez taper :

```
.DEPOSE 770 0
```

Si vous obtenez des caractères trop larges ou beaucoup trop petits, cela est peut-être dû à un mauvais réglage des mini-interrupteurs de l'imprimante.

Pour tout autre problème, reportez-vous à la section appropriée du manuel de référence qui accompagne votre imprimante.

Glossaire

Accès à un fichier : Pour avoir accès à un fichier contenu sur une disquette, on doit fournir le nom de cette dernière (la dénomination), suivi du ou des noms des sous-index et du nom du fichier lui-même.

L'imprimante étant considérée comme un type spécial de fichier, le numéro de prise ou de logement auquel l'imprimante est reliée doit être spécifié.

Adresse : L'emplacement d'un registre, une section spécifique de la mémoire, ou toute autre source d'émission ou une destination.

American Standard Code for Information Interchange (ASCII) : Code normalisé utilisé pour permettre l'échange d'information pour les systèmes de traitement de données et l'équipement associé.

Appel : Fait exécuter un programme informatique, une procédure ou une sous-procédure.

Appel de procédure : Une demande d'exécuter la procédure indiquée. Un appel de procédure se fait au niveau supérieur ou à l'intérieur d'une autre procédure.

ASCII : Voir *American Standard Code for Information Interchange*.

Binaire : Qui comporte deux valeurs ou deux états. Représente aussi le système de numérotation de base deux.

Bit : Un chiffre binaire.

Caractère : Une lettre, un chiffre ou tout autre symbole qui permet l'organisation, la manipulation ou la représentation de données.

Chaîne : Une suite de caractères.

Commande : Une procédure Logo, soit une primitive, soit une procédure que vous définissez vous-même, qui ne retourne rien. VE, AVANCE et ECRIS sont des commandes. Voir **opération.**

Curseur : Un trait clignotant qui indique une position à l'écran.

Défilement : Déplace le contenu de l'écran en tout ou en partie soit horizontalement, soit verticalement, de sorte que les nouvelles données apparaissent d'un côté de l'écran alors que les anciennes disparaissent de l'autre.

Démarrage : Processus qui consiste à charger un langage ou un programme d'application dans la mémoire de l'ordinateur à la mise en route de Logo, par exemple.

Dénomination : Une disquette formatée. La dénomination correspond aussi au nom de l'index lorsque l'utilisateur se trouve au niveau supérieur.

Donnée : L'information que nécessite une primitive ou une procédure pour être exécutée.

Echo : Répercussion des données vers l'émetteur. Par exemple, une touche enfoncée est généralement représentée par un caractère à l'écran.

Ecriture : Enregistrement d'une information dans une mémoire ou sur un support magnétique quelconque.

Editer : Introduire, modifier ou effacer des données.

Effacer : Retirer des indications de l'espace de travail ou d'un fichier de façon permanente.

Elément : Une partie d'un tout, plus particulièrement une composante d'un ensemble.

Emplacement : Tout endroit où des données peuvent être stockées.

Entier : Un nombre positif ou négatif qui n'est pas fractionnaire.

Erreur : Une erreur dans un programme.

Espace de travail : Partie de la mémoire de l'ordinateur qui contient les variables, les procédures et les propriétés aussi longtemps que l'ordinateur est allumé.

Exécuter : Faire accomplir une instruction ou un programme informatique.

Fichier : Ensemble structuré d'informations qui peut être stocké de façon permanente pour un usage ultérieur.

Fichier ASCII : Fichier de texte dont les caractères sont convertis en code ASCII.

Formatage : Disposition ou agencement des données sur un support d'information tel l'écran ou la disquette.

Implicitement : Une valeur ou une notion fournie par le programme lorsqu'aucune n'est spécifiée.

Index : Répertoire de tous les noms de fichiers contenus sur la disquette ainsi que des indications permettant à ProDOS de retrouver ceux-ci sur la disquette.

Instruction : En langage de programmation, toute expression qui réfère à une commande et aux données de cette dernière.

Instruction conditionnelle : Instruction qui amène Logo à exécuter différentes opérations selon qu'une condition est satisfaite ou pas.

Interactif : Programme qui permet d'établir un dialogue entre l'utilisateur et l'ordinateur.

Interprétation : Processus par lequel Logo effectue l'analyse syntaxique d'une ligne, lui conférant ainsi un sens logique.

K : En parlant de la capacité de stokage, notation décimale qui vaut 2^{10} ou 1024.

Lecture : Sélectionner de l'information dans une mémoire ou sur un support magnétique quelconque.

Liste : Un ensemble d'objets Logo, une suite de mots ou de listes qui débutent et se terminent par des crochets.

Liste de propriétés : Une liste composée d'un nombre pair d'éléments. Chaque couple d'éléments comporte une propriété, telle PRENOM, et sa valeur qui peut être un mot ou une liste, tel EMILIE.

Liste vide : Une liste qui ne contient aucun élément ; < > représente la liste vide.

Mise au point : Processus qui consiste à détecter et à éliminer les erreurs dans un programme.

Mot : Un ensemble de caractères traités comme un tout.

Mot littéral : Représentation explicite d'une valeur, plus particulièrement de la valeur d'un mot ou d'une liste. Un mot littéral est précédé des guillemets (").

Mot vide : Un mot qui ne contient aucun caractère ; " représente le mot vide.

Niveau supérieur : Mode dans lequel les commandes peuvent être exécutées sans faire partie d'un programme.

Nœud : Elément de l'espace de travail. Chaque nœud a une longueur de cinq octets.

Nom : Un mot servant de contenant à une valeur dans l'espace de travail.

Nombre réel : Un nombre décimal, positif ou négatif.

Notation infixée : Façon d'exprimer une opération mathématique où le symbole arithmétique se place entre les données. Voir **notation préfixée.**

Notation préfixée : Façon d'exprimer une opération mathématique où le symbole arithmétique ou la primitive se place avant les données. Voir **notation infixée.**

Notation scientifique : Représentation des nombres sous la forme exponentielle.

Objet : Un mot ou une liste.

Octet : Huit bits.

Opération : Une procédure Logo, soit une primitive, soit une procédure que vous définissez vous-même, qui retourne quelque chose. SOMME, NOMSDISQUES, et POSITION sont des opérations. Voir **commande.**

Opération logique : Un prédicat dont la donnée doit être VRAI ou FAUX.

Périphérique : Tout équipement relié à l'ordinateur, tels l'imprimante, le moniteur ou l'unité de disquette.

Pile : Méthode de stockage temporaire où le dernier élément emmagasiné est le premier élément traité.

Pixel : Un point. Correspond aussi aux octets qui contiennent des indications sur ce point.

Prédicat : Une procédure qui retourne soit VRAI, soit FAUX.

Préfixe : Le nom de la disquette suivi du nom d'un sous-index automatiquement placés devant un nom de fichier ; ce dernier n'est pas précédé d'une barre oblique.

Primitive : Une procédure qui fait partie intégrante de Logo.

Procédure : Une instruction ou une série d'instructions adressées à Logo, portant un nom et qui peuvent être stockées de façon permanente.

Procédure récursive : Une procédure qui s'appelle elle-même. Par exemple :

```
POUR TIRAGE
.
.
.
TIRAGE
NOM
```

ProDOS : Système d'exploitation de l'Apple IIe et de l'Apple IIc qui permet à Logo de fonctionner.

Programme : Un ensemble de procédures qui fonctionnent de façon interactive.

Rapport d'échelle : Un nombre décimal qui correspond à la taille d'un pas vertical de Tortue par rapport à la taille d'un pas horizontal.

Récupération de mémoire : Processus qui consiste à libérer des nœuds et ainsi accroître l'espace nécessaire au stockage.

Sortie : La ou les données d'une primitive ou d'une procédure que Logo donne à une autre primitive ou procédure.

Sous-index : Un ensemble de fichiers regroupés sur une disquette.

Sous-procédure : Une procédure utilisée dans la définition d'une autre procédure. Par exemple :

```
POUR A
B
FIN
```

A appelle B ; B est donc une sous-procédure de A.

Stockage : Processus par lequel un périphérique, ou une partie de périphérique, garde des données en mémoire.

Superprocédure : Une procédure qui en appelle une autre. Par exemple :

```
POUR A
B
FIN
```

A appelle B ; A est donc une superprocédure de B.

Suppression : Processus qui consiste à retirer les derniers éléments d'un mot ou les éléments fractionnaires d'un nombre.

Symbole d'invite : Une question que pose l'ordinateur ou un signal qu'il envoie lorsque l'utilisateur doit fournir des instructions.

Syntaxe : Les règles qui régissent un langage.

Tampon : Section de la mémoire où les données sont temporairement stockées lorsque celles-ci sont transférées d'un périphérique à un autre. Les tampons, par l'entremise de périphériques, permettent d'effectuer des opérations de lecture et d'écriture.

Tampon d'édition : Section de la mémoire contenant tout le texte qui se trouve dans l'éditeur Logo.

Tortue : Forme triangulaire apparaissant à l'écran qui utilise un crayon pour tracer des lignes.

Valeur : Le contenu d'une variable.

Variable : Un contenant portant un nom et qui comporte une valeur.

Variable globale : Une variable qui se trouve toujours dans l'espace de travail, et qui est créée à l'aide de la primitive RELIE. Voir **variable locale.**

Variable locale : Une variable qui n'a d'effet que lorsqu'une procédure est exécutée. Voir **variable globale.**

Index

Index

B

C

D

E

F

G

H

I

J, K, L

M

N

O

P

Q

R

S

T

U, V

APPLE COMPUTER FRANCE
Avenue de l'Océanie
Z.A. de Courtabœuf - B.P. 131
91944 LES ULIS CEDEX

Imprimé en Irlande

F 030-0866-A