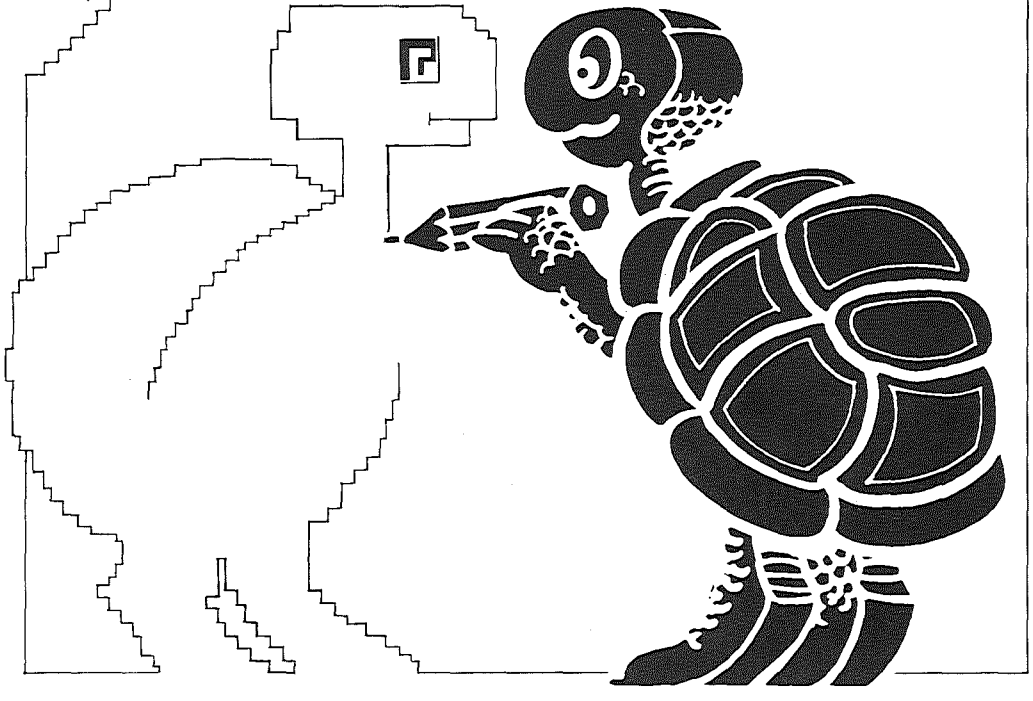
EDICIEL
MATRA ET HACHETTE

EDI•LOGO



Cet ouvrage a été préparé avec l'aide de la subvention n° SED-7919033 de la Fondation Nationale des Sciences (NSF). Toutefois toutes les opinions, conclusions et recommandations figurant dans cet ouvrage sont celles des auteurs et ne reflètent pas nécessairement les vues de la Fondation.

Ce classeur comprend :

- Une introduction à EDI-LOGO à l'intention des parents
- *Premiers pas avec EDI-LOGO,* manuel d'apprentissage
- EDI-LOGO, manuel de référence
- Trois fiches-mémoires reprenant les commandes principales
- Trois autocollants destinés aux touches de votre APPLE II
- Deux copies de la disquette langage EDI-LOGO
- Une copie non protégée de la disquette utilitaire (faites-en tout de suite une copie, elle n'est pas garantie)

La version française d'EDI-LOGO et ces documents ont été réalisés par Catherine BERDONNEAU.

Certifiée de Mathématiques et titulaire d'un Doctorat de Didactique des Mathématiques, Catherine BERDONNEAU a participé aux études menées sur LOGO en France depuis le début de leur structuration en Recherche Coopérative sur Programme.

Membre fondateur de l'Association GREPACIFIC qui regroupe les chercheurs et les utilisateurs de LOGO en France, elle poursuit des expérimentations principalement au niveau de l'école primaire et participe à la formation des enseignants à l'approche LOGO.

Nous remercions pour leur assistance :

- Harold ABELSON et son équipe
- Nicholas NEGROPONTE et le Centre Mondial Informatique et Ressource Humaine, en particulier Alain et Pascal CHESNAIS
- Gérard BOSSUET et les membres du GREPACIFIC
- Philippe AUPHELLE, Eric CHEVAL, Maurice LEMOINE et Pierre MORNET

EDI-LOGO

INTRODUCTION A EDI-LOGO A L'INTENTION DES PARENTS

EDI-LOGO est un langage de programmation, au même titre que BASIC, Pascal ou FORTRAN. Mais c'est aussi toute une approche éducative, développée au cours de plus de quinze années d'études et d'expérimentations dans divers pays (dont près de dix ans en France).

A l'inverse de l'Enseignement Assisté par Ordinateur au sens strict du terme, il n'est pas question ici que la machine transmette à l'élève un certain nombre de savoirs pré-déterminés. Au contraire, la puissance de l'ordinateur est mise au service de l'utilisateur, pour lui permettre une libre exploration d'un certain nombre de domaines de connaissance.

• *Caractéristiques du langage EDI-LOGO*

EDI-LOGO est un langage de programmation, à la fois puissant, sophistiqué et d'une très grande simplicité. Facile à comprendre et d'une manipulation particulièrement agréable, il constitue l'introduction idéale à une programmation structurée.

1 . *EDI-LOGO est interactif :* Vous pouvez faire exécuter immédiatement vos commandes et en voir ainsi aussitôt le résultat. Ce dialogue avec l'ordinateur transforme l'ordinateur en un ami toujours prêt à rendre service. Les messages d'erreur sont courtois et explicites : pas besoin d'avoir recours au manuel pour cerner ce qui ne va pas.

2 . *EDI-LOGO est procédural :* Très vite, vous aurez envie de réutiliser quelque chose que vous venez de faire faire par la machine, et de compléter ou de compliquer votre premier essai. Les procédures sont le moyen idéal, induisant des méthodes efficaces de résolution de problèmes qui s'adaptent avec souplesse à votre propre raisonnement et ne vous imposent jamais un mode de pensée rigide.

3 . *EDI-LOGO est extensible :* Les procédures que vous définissez constituent autant de mots nouveaux qui viennent enrichir le vocabulaire de base de votre micro-ordinateur. Elles ont exactement le même statut que les commandes primitives, et s'emploient indifféremment.

4 . *EDI-LOGO est récursif :* Beaucoup plus sophistiquée que l'itération par boucles, la récursion offre une puissance et une élégance inattendues pour bien des problèmes de répétition.

5 . *Manipulation du texte :* EDI-LOGO travaille sur des données structurées, les listes, les mots et les caractères. Les programmeurs amateurs de jeux de langage, ou désireux de produire des textes aléatoires, trouveront une réponse agréable à leurs projets.

6 . *EDI-LOGO comporte le graphisme tortue :* Avoir une tortue pour réaliser des tracés, c'est pour le jeune enfant projeter son propre corps sur le monde qui l'entoure. Inutile de passer par les complications d'un repère cartésien avec des coordonnées en *x* et en *y* pour obtenir des figures éventuellement complexes (spirales par exemple). LOGO a été le premier langage à introduire le graphisme tortue, repris ensuite par Pascal et maintenant par bien d'autres langages.

- ***Avec les plus jeunes***

Dès trois ou quatre ans, un enfant appréciera de pianoter sur un ordinateur. Utilisez le programme PICOLO qui est fourni sur la disquette utilitaire. Pour cela, chargez LOGO puis, retirez la disquette langage du lecteur, insérez alors la disquette utilitaire et tapez

RAMENE "PICOLO.PARLE RETURN

puis tapez

DEBUT

Si vous disposez de la carte de synthèse de la parole "Porte-Parole" d'EDICIEL, les explications seront données de vive voix par la machine. Sinon, il vous faudra donner à l'enfant les explications suivantes :

Le petit triangle que tu vois au centre de l'écran est la tortue. Elle regarde vers le haut de l'écran.

Si tu lui donnes des ordres pour la déplacer, tu pourras lui faire tracer de jolis dessins.

Pour la faire avancer, appuie sur la touche **A**. *Essaie (non, ce n'est pas la bonne touche). Voilà : tu vois, elle a avancé. Fais-la avancer encore plusieurs fois. (Attention, pas trop, sinon elle va se cogner contre les bords de l'écran).*

Maintenant, tu peux la faire tourner vers la droite en appuyant sur la touche **D**. *Essaie. (Non, ce n'est pas la bonne touche). Tu vois, elle a tourné vers la droite. Fais-la avancer maintenant : elle avance dans la direction où elle regarde. Tu peux aussi la faire reculer, en appuyant sur la touche* **R**. *Essaie (non, ce n'est pas la bonne touche). Tu vois, elle revient sur ces pas, à reculons. Appuie encore plusieurs fois sur cette touche ... Elle continue à reculer. Si tu veux la faire tourner à gauche, appuie sur la touche* **G**. *Essaie (non, ce n'est pas la bonne touche). Tu vois, elle a tourné vers la gauche ; si tu appuies encore sur la touche* **G**, *elle va continuer à pivoter vers la gauche.*

Pour nettoyer l'écran et commencer un nouveau dessin, appuie sur la touche **N**. *Il y a d'autres touches pour donner d'autres ordres. Si tu veux tous les connaître, appuie sur la touche* **Z**.

Laissez-le alors prendre l'initiative : les enfants ont beaucoup plus d'idées que nous, ne les limitons pas sur les choses à faire avec une tortue d'écran.

Si vous trouvez que les touches avec des lettres sont trop tristes, pourquoi ne pas les masquer avec des pastilles auto-collantes :
vert pour AVANCE
rouge pour RECULE
jaune pour DROITE etc...

Mieux encore, vous pouvez explorer avec l'enfant, sans lui donner d'explications initiales, les diverses touches du clavier. Quand une touche produit une action, identifier cette action et choisir un codage qui sera collé sur la touche concernée. Cela vous obligera peut-être à une certaine gymnastique, si vous devez vous souvenir qu'une fleur fait tourner à gauche, un oiseau à droite, un chien avancer, etc...

- ***Dès que la lecture est acquise***

Laissez l'enfant explorer les premiers chapitres du manuel d'apprentissage PREMIERS PAS AVEC EDI-LOGO. Les chapitres 2 à 5 sont tout à fait abordables par des élèves de l'école primaire.

Le texte du chapitre convient à un lecteur adulte, mais les enfants ont leur manière d'en tirer parti. Une lecture "en diagonale", leur permet de retenir quelques termes, et au moment où un exemple est présenté, cet exemple va être recopié sur le clavier. L'objet réalisé apporte l'information qui était contenue dans le texte, et qui a été ainsi acquise d'une autre manière.

Les fiches-mémoires sont très utiles. Il ne faut pas s'étonner (et surtout ne pas freiner) d'une exploration à tâtons des primitifs, pour voir ce qu'ils font. Les messages de la machine sont aussi là pour guider l'enfant.

Offrez à chacun des enfants un joli cahier, ou un classeur, où il ou elle pourra prendre l'habitude de garder un suivi de ses activités avec EDI-LOGO. Evitez le quadrillage Séyès (les traditionnels cahiers d'écolier) et préférez le quadrillage au centimètre ou au demi-centimètre qui peuvent être utilisés indifféremment dans l'un ou l'autre sens.

- ***Pour les plus grands***

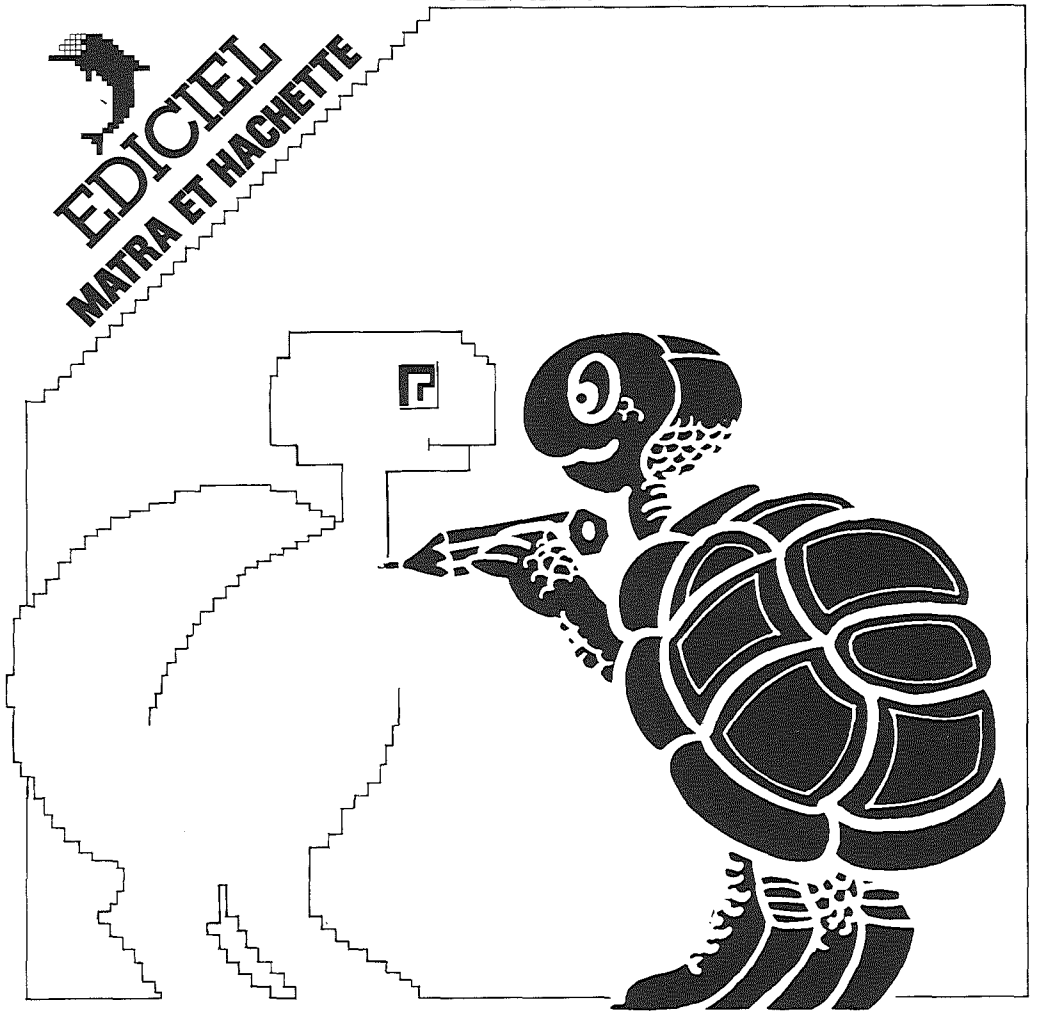
Le manuel d'apprentissage couvre la plupart des primitifs du langage. La liste complète est fournie avec description et exemple dans le manuel de référence, qui peut aussi conduire à l'utilisation de l'assembleur.

Alors, **BIENVENUE A EDI-LOGO ...**

mais n'oubliez pas de rappeler de temps en temps à chacun qu'il est bon aussi de consacrer quelques moments au patin à roulettes, à la lecture ou au bricolage.

CATHERINE BERDONNEAU

PREMIERS PAS AVEC EDI-LOGO



BIBLIOGRAPHIE

EN FRANÇAIS

- Seymour PAPERT — Jaillissement de l'esprit *Flammarion 1981*
- Bertrand SCHWARTZ — L'informatique et l'éducation *La Documentation Française 1981*
- Gérard BOSSUET — L'ordinateur à l'école *Presses Universitaires de France 1982*
- Catherine BERDONNEAU, Rose-Marie DUMAS — Une tortue dans une classe : une année d'expérimentation en Cours Moyen 2e année *Cahier PACIFIC n° 1 1982*
- Catherine BERDONNEAU — Recueil des Pratiques Pédagogiques autour de LOGO *A.D.I. - I.N.R.P. 1983*
- Harold ABELSON — LOGO for the APPLE II *Traduction française par SOGIDES (Montréal) (à paraître)*

EN ANGLAIS

- Paul GOLDENBERG — Special Technology for Special Children *University Park Press 1979*
- Harold ABELSON, Andrea di SESSA — Turtle Geometry *MIT Press 1981*

SOMMAIRE

SOMMAIRE

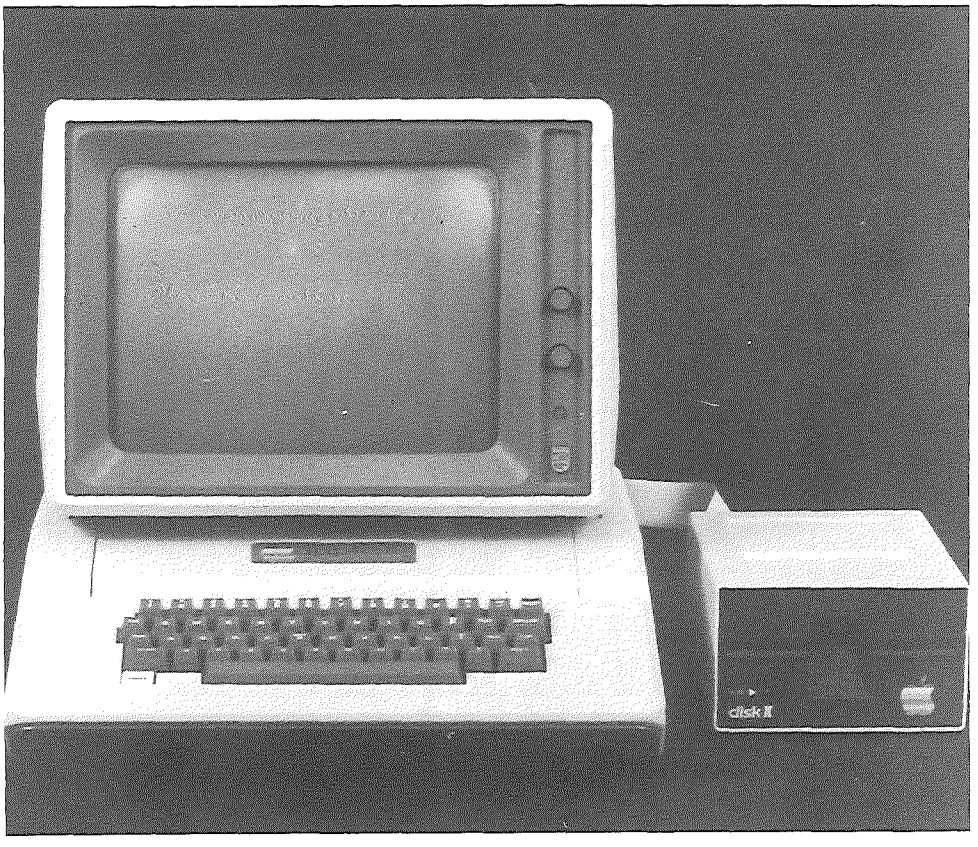
CHAPITRE I

MISE EN ROUTE

La version de LOGO dont vous disposez a été conçue pour le micro-ordinateur Apple II (ou Apple II plus) : celui-ci doit comporter 48 K de mémoire vive, et être équipé d'une carte langage 16 K ; il vous faut aussi un lecteur-enregistreur de disquettes, qui sert à la fois pour charger le langage LOGO à partir de la disquette maître, et également pour garder une partie de votre travail pour des séances ultérieures. Un seul lecteur de disquettes suffit.
Si vous avez un Apple IIe, voir page 185.

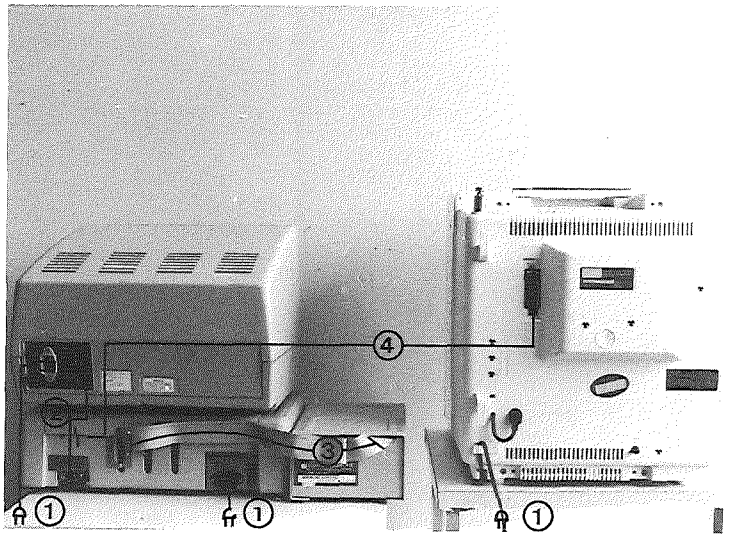
• Chargement du langage

Vérifiez que votre micro-ordinateur Apple (photo 1) est correctement mis en place (photo 2) : la console doit être reliée soit au moniteur par un câble rond (câble n° 2) qui va de "VIDEO IN" sur le moniteur à "VIDEO OUT" sur la console, soit à une TV couleur par un câble (câble n°4) qui s'enfiche sur la prise PERITEL ; la console doit aussi être reliée au lecteur de disquettes par un cable plat (câble n° 3). La console et le moniteur doivent être alimentés (câbles n° 1).

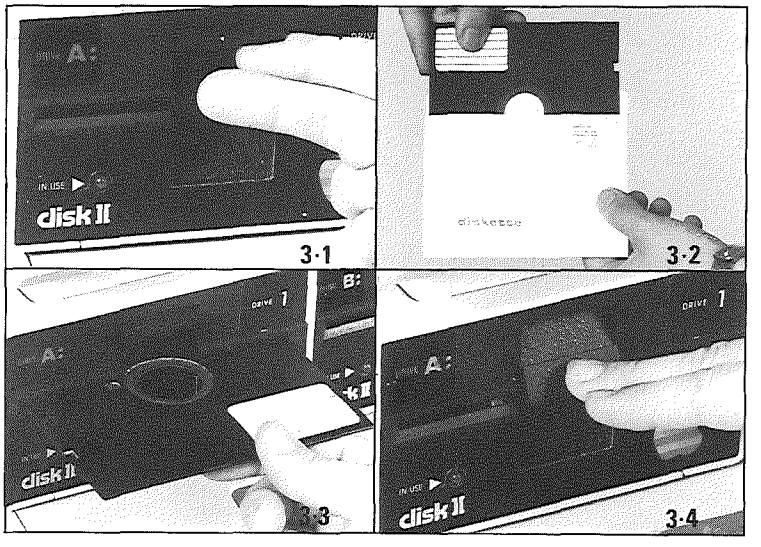


CHAPITRE I

MISE EN ROUTE



Soulevez le clapet de fermeture du lecteur-enregistreur de disquettes (photo 3-1). Sortez votre disquette LOGO de son enveloppe de protection, en la tenant par l'étiquette (photo 3-2) ; insérez-la dans le lecteur (photo 3-3) et refermez le clapet (photo 3-4).



Maintenant, allumez le moniteur, puis la console (le commutateur se trouve à l'arrière de la console, vers votre gauche) : vous devez voir s'allumer la lampe témoin (blanche) située en bas à gauche du clavier, ainsi que la lampe témoin (rouge) du lecteur-enregistreur de disquette.

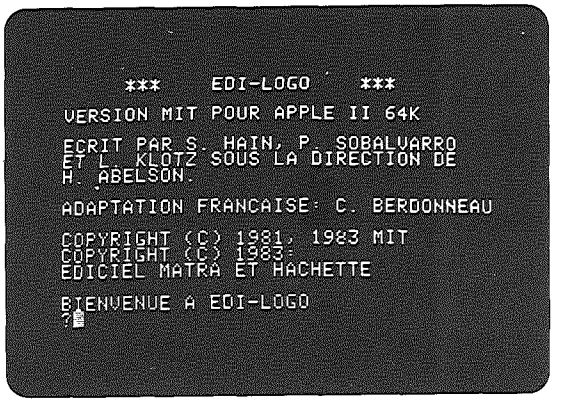
Sur votre écran s'affiche

APPLE][*(Ce que l'ordinateur affiche à l'écran est imprimé dans ce manuel en blanc sur fond noir)*

puis le message

UN MOMENT..., MERCI

Au bout d'un certain temps, cette petite lumière rouge va s'éteindre, et sur l'écran de votre moniteur va s'inscrire un message de bienvenue



Sous le message de bienvenue, vous voyez un point d'interrogation, et un carré qui clignote. Ceci vous indique que LOGO est prêt, et que vous pouvez maintenant taper quelque chose. Ce carré qui clignote est appelé *le curseur.*

CHAPITRE I

MISE EN ROUTE

• *Le clavier de l'Apple II*

Le clavier de votre micro-ordinateur Apple ressemble à celui d'une machine à écrire.

La rangée supérieure de touches, appelée aussi rangée des chiffres, comporte treize touches : les dix premières sont les chiffres, les deux suivantes sont le signe de ponctuation : et le signe opératoire – . La touche la plus à droite est la touche [RESET] . C'est la seule touche qu'il faut absolument éviter de toucher ; en effet, si vous appuyez sur [RESET] , vous allez voir s'afficher à l'écran une ligne qui ressemble à ceci :

```
5E7B- A=ØØ X=7B Y=27 P=37 S=D1
```

Il vous faudra alors éteindre la console et reprendre le chargement de LOGO.

La deuxième rangée de touches est parfois appelée rangée des voyelles : en effet, toutes les voyelles y figurent, sauf le **A** qui est à gauche de la rangée suivante. Faites bien attention de ne pas taper un **I**

au lieu d'un **1**, ou un **O** au lieu d'un **Ø**, ou l'inverse : il n'y a guère de risque de se tromper puisque les chiffres sont sur une rangée à part. De plus, l'affichage à l'écran ne permet aucune confusion : tapez un **l** et un **1**, on voit nettement la différence ; de même **O** et **Ø** sont bien distincts, puisque le **Ø** est barré. Dans tout ce qui va suivre, pour faciliter la lecture et vous habituer à cette notation, nous noterons toujours le **Ø** avec une barre.
Tapez plusieurs touches de ces rangées pour vous familiariser avec les positions de chaque caractère. Essayez maintenant la touche située le plus à gauche de cette rangée, et qui est marquée [ESC]* : elle vous permet d'effacer le caractère qui est situé immédiatement à gauche du curseur. Appuyez deux ou trois fois sur la touche [ESC] pour effacer autant de caractères. Maintenant, tout en gardant la touche [ESC] enfoncée avec un doigt de votre main gauche, appuyez sur la touche [REPT] qui est située en avant-dernière place de cette même rangée. Les caractères situés à gauche du curseur s'effacent successivement, jusqu'à ce que la ligne soit vide.

Si vous appuyez sur la touche **Q**, *au lieu de la touche* ESC *avec la main gauche, et sur la touche* [REPT] *avec la main droite, vous voyez une série de lettres* **Q** *s'inscrire sur l'écran : en maintenant ces deux touches appuyées, vous produirez environ six lignes et demie de ce même caractère* **Q**. *Si vous comptez bien, vous remarquerez que chaque ligne d'affichage du moniteur comporte quarante signes (le* **?** *et trente-neuf fois le caractère* **Q** *pour la première ligne), et qu'en tout vous avez affiché 255 fois le caractère* **Q**. *Nous aurons l'occasion de reparler de cette différence entre le nombre de caractères d'une ligne d'affichage sur l'écran du moniteur et la longueur de la "ligne d'ordinateur" (la longueur du* tampon). *Effacez à nouveau tous les caractères en appuyant sur* [REPT] *pendant que vous maintenez la touche* [ESC] *enfoncée.*

La ligne suivante commence, à part la touche [CTRL] , par la voyelle **A**, puis des consonnes, et

* *ESC est le début du mot anglais ESCAPE, qui signifie s'échapper. REPT est une abréviation de REPETE. Vous trouverez un autocollant EFF à poser sur la touche ESC.*

une touche qui affiche le signe de ponctuation **;** . Tapez quelques-unes de ces lettres, par exemple

ASDFGHJKL *(ce que vous tapez est imprimé dans ce manuel en noir sur fond blanc)*

puis appuyez sur la touche [←]. Le curseur est maintenant sur la lettre **L**. Si vous appuyez maintenant sur [ESC], c'est le caractère **K** qui est effacé. Si vous appuyez sur **QW**, ces deux lettres s'insèrent après le **J**, et la lettre **L** se décale à chaque fois vers la droite. Là encore, vous pouvez utiliser la touche [REPT], en maintenant la touche [←] enfoncée, pour amener le curseur en début de ligne. Remarquez que la touche [←] *déplace* le curseur vers la gauche sans *effacer* aucun caractère.

De la même manière, la flèche [→] déplace le curseur vers la droite sans effacer non plus. Si vous gardez [→] et [REPT] enfoncés, le curseur va probablement dépasser les caractères déjà affichés, et continuer jusqu'à ce que le curseur arrive en bout de ligne d'écran, passe à la ligne suivante, et se déplace encore jusqu'à s'arrêter en bout de tampon.

A gauche de cette ligne se trouve la touche [CTRL] (contrôle). Elle s'utilise conjointement avec une autre touche, comme la touche [REPT] que nous avons vue précédemment. Par exemple, pendant que vous maintenez la touche [CTRL], tapez sur **G** : sur l'écran s'imprime le message :

ARRET !

et le curseur est passé à la ligne suivante. La commande [CTRL] G indique à LOGO d'arrêter immédiatement ce qui était en cours. C'est une commande très utile.

Tapez à nouveau quelques caractères, par exemple

QWERTY

puis, à l'aide de la touche ←, positionnez le curseur sur le E. Enfoncez alors la touche CTRL , et pendant que vous la maintenez, tapez sur D : le E disparaît, et le curseur est maintenant sur le R. Déplacez-le maintenant vers la droite, pour l'amener sur le Y, et utilisez à nouveau CTRL D. Vous avez maintenant le curseur en bout de ligne, à droite de QWRT. Si vous essayez encore une fois CTRL D, rien ne se produit à l'affichage il n'y avait en effet rien à effacer sous le curseur.

La touche ESC efface ce qui est à gauche du curseur ; la combinaison CTRL D efface ce qui est sous le curseur : un bon moyen pour s'en souvenir, c'est de se rappeler que D est le commencement de Détruis Dessous.

La fin de l'alphabet et quelques autres signes se trouvent sur la rangée inférieure. A gauche et à droite, une touche marquée SHIFT joue à peu près le rôle de la touche de majuscules sur une machine à écrire ; bien sûr, pas pour les lettres, qui sont toujours affichées en majuscules avec LOGO ! La touche SHIFT sert pour les touches qui comportent deux caractères. Par exemple ⟨, sur la dernière rangée : si vous frappez cette touche, une virgule , s'affiche sur l'écran ; si maintenant vous appuyez sur l'une des deux touches SHIFT en la maintenant enfoncée, et que vous frappiez cette même touche ⟨, vous voyez le signe < qui s'affiche à l'écran.

De la même façon, il faut appuyer sur SHIFT pour afficher le signe + (juste à gauche des deux touches ← et →), ou pour avoir = (à droite de la rangée des chiffres).

En appuyant sur SHIFT M, vous affichez un crochet fermant] , à ne pas confondre avec la parenthèse

fermante) qui s'obtient avec SHIFT 9. En appuyant sur SHIFT N , vous affichez un crochet ouvrant [; la parenthèse ouvrante s'obtient avec SHIFT 8.

Vous trouverez dans votre coffret LOGO une série d'étiquettes autocollantes pour mettre sur les touches ESC (étiquette EFF) , M (étiquette] M) et N (étiquette [N).

La grande touche en bas du clavier permet d'afficher un espace blanc : c'est ainsi, comme quand on écrit en français, qu'on sépare les mots en LOGO.

Pour bien vous familiariser avec les touches spéciales (ESC , ← , → , CTRL **D**, SHIFT , RETP) et le reste du clavier, essayez de taper le texte suivant :

VOULEZ-VOUS DANSER, GRAND-MERE

puis modifiez-le en

VOULEZ-VOUS CHANTER, GRAND-PERE

et ensuite

VENEZ CHANTER AVEC MICKEY

ou encore

ALLONS DANSER AVEC MINNIE

- ***Dialoguer avec l'ordinateur en LOGO***

Pour le moment nous avons tapé un certain nombre de choses au clavier, et tout cela s'est affiché à l'écran. Mais c'est un peu comme si nous avions écrit une lettre pour un ami et que nous l'ayons gardée dans notre poche au lieu de la mettre dans une boite à lettre : l'information n'est pas arrivée à son destinataire. Pour que l'information arrive à l'ordinateur, il faut appuyer sur la touche RETURN:

c'est un peu comme si on lui disait ''fais-le !''.

Remarquez que les touches [RETURN] , [ESC] , [CTRL] , [REPT] *ou* [SHIFT] *ne produisent aucun affichage spécial à l'écran. La touche* [RETURN] *déplace le curseur, de la fin de ligne où il était, en début de ligne suivante.*

Vous avez appuyé sur la touche [RETURN] alors que le curseur était à la fin de la ligne

[ALLONS DANSER AVEC MINNIE]

Vous voyez maintenant un message, qui vous indique

ALLONS N'A PAS ETE DEFINI

C'est un message qui s'affichera à chaque fois que votre ordre n'aura pas été compris par l'ordinateur. Essayons quelques ordres :

[34 + 5] (n'oubliez pas [RETURN])

Vous devez avoir un nouveau message :

QUE DOIS-JE FAIRE AVEC 39

Essayons une soustraction (pensez à appuyer sur [RETURN] en fin de message, nous ne le mentionnerons plus)

[1Ø28 – 15]

là encore

QUE DOIS-JE FAIRE AVEC 1Ø13

(si vous obtenez un message qui commence par APPRENDS-MOI..., c'est probablement parce que vous avez mélangé des 1 et I, O ou Ø).

[AFFICHE [LOGO EST UN LANGAGE]]

Là, vous n'avez pas de message, mais votre ordre est exécuté.

AFFICHE 183 + 4

(nous verrons plus tard tous les calculs qu'on peut lui demander). Si vous obtenez le message
AFFICHE183 N'A PAS ETE DEFINI

c'est que vous avez oublié l'espace (utilisez la grande touche du bas du clavier) entre **AFFICHE** et **183**. C'est un oubli fréquent : vous en êtes quitte pour retaper toute votre ligne !
Essayez encore de taper quelques lignes pour manipuler le clavier et les diverses touches. Si vous ne voulez pas avoir de message d'erreur en réponse, au lieu de la touche RETURN , appuyez sur CTRL **G**. N'hésitez pas à faire des essais, rien de ce que vous pouvez taper au clavier ne risque d'abîmer votre Apple, ni LOGO. Le pire qu'il puisse vous arriver, c'est de heurter par mégarde la touche RESET , et alors vous n'aurez plus qu'à charger LOGO à nouveau.

- ***Initialiser une disquette***

Avant d'aller plus loin, il serait bon que vous vous prépariez une copie de la disquette *utilitaire*, et quelques disquettes de travail qui vous serviront à partir du chapitre 3.
Eteignez votre micro-ordinateur Apple, et insérez la disquette utilitaire (pas la disquette *langage LOGO* dont vous venez de vous servir) dans le lecteur, puis allumez l'appareil.
Tapez

LOAD HELLO RETURN

La lampe témoin rouge du lecteur s'allume. Quand elle s'éteint, retirez la disquette utilitaire, et insérez une disquette vierge. (Attention : cette disquette va être *initialisée*, c'est-à-dire que toutes les informations qui auraient pu y être stockées seront détruites : il est donc vital de ne pas laisser votre disquette utilitaire dans le lecteur !) Quand la disquette vierge est mise en place dans le lecteur, et la porte du lecteur refermée, tapez

INIT HELLO RETURN

Là encore, la lampe témoin s'allume, puis, au bout d'environ une minute, s'éteint : la disquette est alors prête à l'emploi, et vous permettra de sauvegarder des fichiers LOGO.

Une même disquette peut recevoir à la fois des fichiers LOGO et des fichiers BASIC. Le catalogue vous indique sans confusion possible la nature de chacun des fichiers.

Pour faire une copie de votre disquette utilitaire, servez-vous du programme de copie qui figure sur la disquette "System Master", qui vous a été fournie en même temps que votre Apple. Pour cela, conformez-vous aux indications de la documentation générale de votre micro-ordinateur Apple.

CHAPITRE II

PILOTER UNE TORTUE

Le pilotage d'une tortue peut se faire d'une manière très simplifiée grâce au programme "PICOLO", dont vous trouverez les détails au chapitre 9 p. 156. Comme son nom l'indique, ce programme est plus particulièrement destiné aux très jeunes enfants, même s'il n'est pas dénué d'intérêt pour les non-débutants.

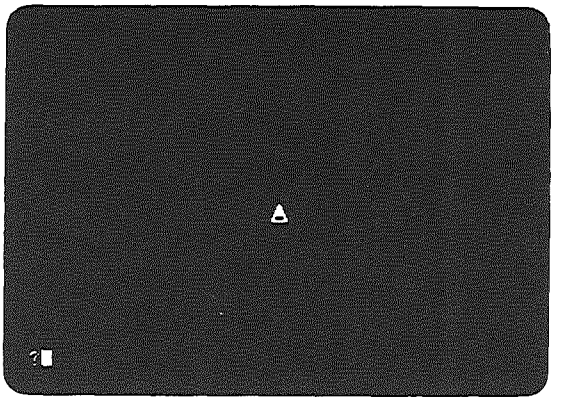
Dans ce chapitre, nous allons *piloter une tortue* : nous introduisons petit à petit des commandes nouvelles, en montrant des exemples et en proposant des idées d'exercices. Il est conseillé de taper tous les exemples donnés, pour mieux se familiariser avec les commandes. Quant aux exercices, ils ne sont là qu'à titre de suggestions. Si à partir des exemples vous avez d'autres idées pour prolonger l'activité, suivez votre propre chemin jusqu'à l'endroit où il vous emmènera : on apprend toujours mieux et plus à partir de projets que l'on se fixe soi-même que par des exercices fournis de l'extérieur. Le chapitre 12 fournit un certain nombre d'indications pour mener à bien les exercices proposés, et des pistes pour poursuivre encore plus loin.

• *Dessiner avec une tortue*

La première commande pour utiliser une tortue et réaliser des dessins est

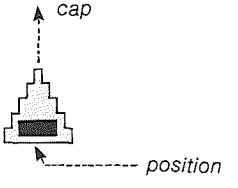
DESSINE DESSINE

(N'oubliez pas de taper RETURN après chaque commande, nous ne vous le rappellerons pas systématiquement).
Là il se produit un changement important sur l'écran de votre moniteur : toutes les commandes ou les textes que vous aviez inscrits précédemment disparaissent, y compris l'ordre DESSINE. Au centre de l'écran, on voit un petit triangle. De plus, le point d'interrogation et le curseur sont maintenant presque en bas de l'écran.



Ce petit triangle est appelé *la tortue*. Cette tortue a une *position* (là où elle se trouve), et un *cap* (vers où elle regarde).

tortue

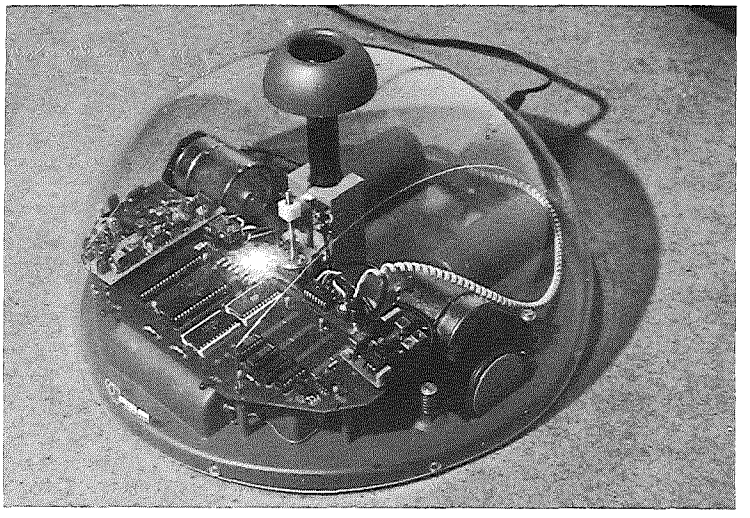


Quand on est en mode DESSINE, l'écran est séparé en deux ''pages'' qui se chevauchent : une page sur laquelle on va déplacer la tortue et réaliser les dessins, et une page sur laquelle s'affichent les commandes. Normalement, on voit les quatre lignes du bas de l'écran pour l'affichage des commandes, la partie supérieure étant réservée au dessin. Mais on peut récupérer les parties cachées...

Les premières tortues étaient de petits robots, munis de deux roues et protégées par une coupole. Ces tortues étaient munies d'un traceur (un stylo-feutre, qu'on continue à appeler la *plume*), et certaines avaient aussi un phare, un avertisseur sonore, et

CHAPITRE II

PILOTER UNE TORTUE



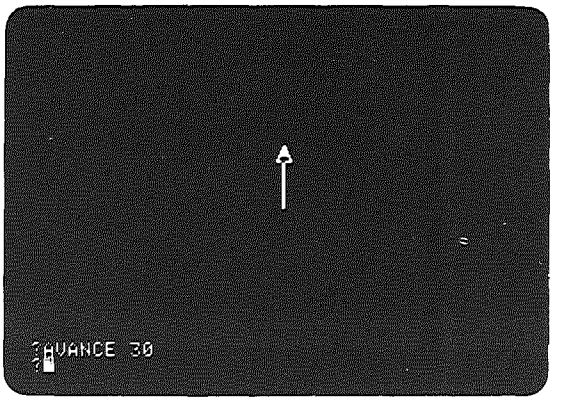
des senseurs pour prévenir si elles se heurtaient à un obstacle. Elles se déplaçaient par terre, et pouvaient dessiner sur de grandes feuilles de papier posées sur le sol.
Notre tortue d'écran est beaucoup plus schématique, mais elle ''possède'' toujours une plume, qui au départ est en position baissée : un déplacement de la tortue produit donc un trait.

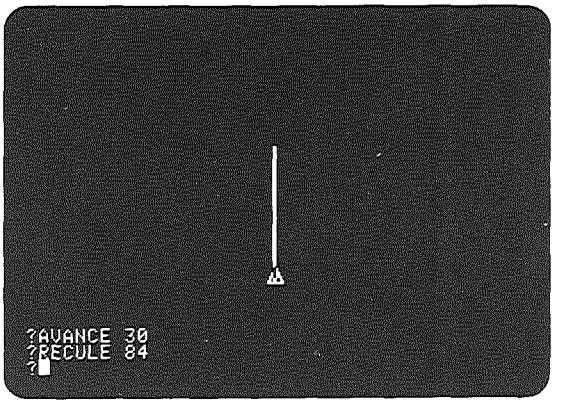
• *Déplacer la tortue*

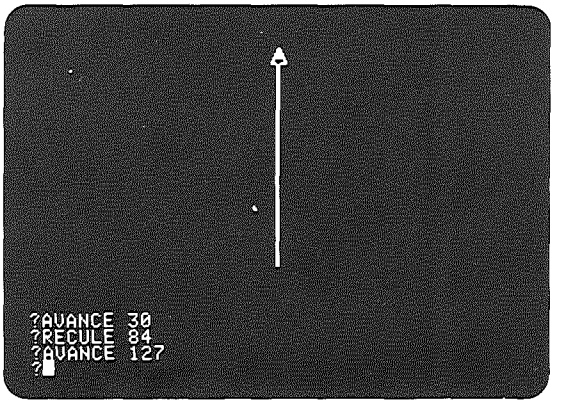
Pour faire avancer ou reculer la tortue, il suffit de le lui indiquer en précisant le nombre de pas qu'elle doit effectuer : un pas de tortue est une unité assez petite. Essayez quelques commandes pour déplacer la tortue :

```
AVANCE
RECULE
```

```
AVANCE 30
RECULE 84
AVANCE 127
```


?AVANCE 30
?


?AVANCE 30
?RECULE 84
?


?AVANCE 30
?RECULE 84
?AVANCE 127
?

CTRL F

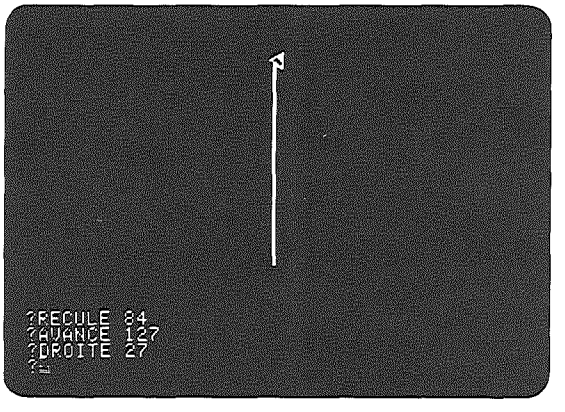
CTRL S

CTRL T

Si vous utilisez un nombre un peu grand pour RECULE, le trait va se poursuivre sous la page de texte, et on ne voit plus ni cette partie du trait, ni le petit triangle de la tortue. Il est possible de rendre visible toute la page de dessin, en tapant CTRL **F** *: les quatre lignes de texte disparaissent alors, ce qui n'empêche nullement de continuer à entrer des ordres. La partie texte redevient automatiquement visible (cachant à nouveau le bas du dessin) si vous tapez quelque chose que la machine ne comprend pas et qu'elle vous retourne un message. Mais vous pouvez aussi retrouver vous-même l'écran mixte en tapant* CTRL **S**. *Par ailleurs, vous aurez peut-être envie de relire des commandes que vous avez tapées antérieurement, et qui sont "montées" sous la page de dessin : pour cela, tapez* CTRL **T**. *On revient ensuite à la page mixte ou à la pleine page de dessin par* CTRL **S** *ou* CTRL **F** *respectivement. (Pour vous souvenir de ces commandes, pensez que* **T** *est le début de* Texte, **F** *le commencement de* Figure, *et* **S** *la première lettre de* Séparé).
Si vous utilisez des nombres trop grands pour AVANCE ou RECULE, il se produit un effet d'enroulement. (voir p. 24-25)

Pour faire pivoter la tortue vers la droite ou vers la gauche, il suffit de le lui indiquer, en précisant de combien elle doit tourner : l'amplitude de la rotation se mesure en degrés, Essayez par exemple

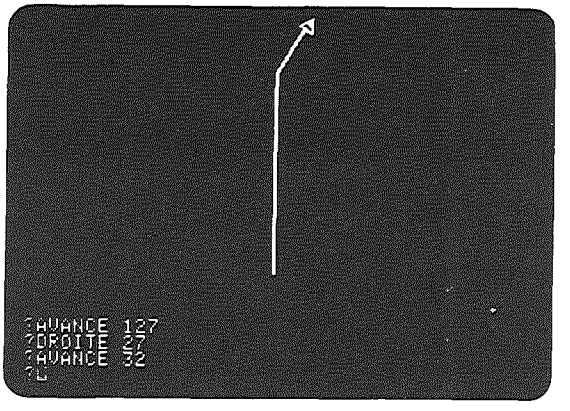
DROITE **DROITE 27**



La tortue regarde maintenant un peu vers la droite

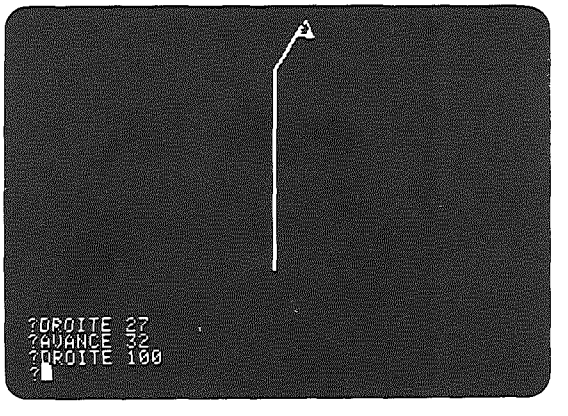
AVANCE 32

elle se déplace "en biais" : AVANCE déplace



toujours la tortue dans le sens où elle regarde. Vous remarquerez que le trait qu'elle laisse cette fois-ci est un peu irrégulier et ressemble à un escalier. Cela est dû à la manière dont sont fabriqués les postes de télévision et les moniteurs : seules les droites ''verticales'' et ''horizontales'' sont bien droites, les autres ont toujours l'air de marches.

DROITE 100



La tortue regarde maintenant en bas à droite...

Il est indispensable d'indiquer le nombre de pas dont la tortue doit avancer ou reculer, et le nombre de degrés dont elle doit pivoter vers la droite ou vers la gauche. Si vous oubliez cette précision, LOGO vous le signalera :

```
AVANCE
IL MANQUE QUELQUECHOSE APRES AVANCE
```

Il vous faut alors retaper AVANCE suivi d'un nombre (de pas).

GAUCHE

```
GAUCHE
IL MANQUE QUELQUECHOSE APRES GAUCHE
```

Il vous arrivera fréquemment de voir LOGO vous retourner un message indiquant que votre commande n'a pas été comprise : ... N'A PAS ETE DEFINI. Regardez attentivement le mot qui n'a pas pu être interprété : vous y trouverez presque toujours une erreur de frappe (AVAMCE, DRØITE), ou l'absence d'un espace séparateur (RECULE3Ø). Un ordinateur ne sait pas se débrouiller avec des fautes d'orthographe ou des mots qui ne lui ont pas été enseignés. Les messages qui s'affichent à l'écran sont en général assez explicites pour être compris dès la lecture ; en cas de doute, reportez-vous au manuel de référence.
Si vous obtenez systématiquement ce message avec un Apple IIe c'est que la touche majuscules (en bas à gauche du clavier) n'est pas enfoncée.

• *Dessinons un escalier*

Maintenant que vous connaissez les premiers ordres pour commander la tortue, si nous dessinions un escalier ? Mais avant, il nous faut une page propre, que nous obtenons avec

```
DESSINE
```

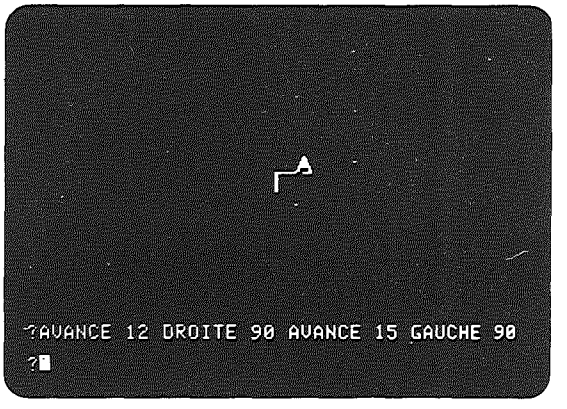
Cette commande efface tout ce qui était sur la page de dessin, et ramène la tortue au centre de l'écran, pointe vers le haut.
Chaque marche de l'escalier se fait ainsi : on avance la tortue de la hauteur de la contremarche, puis on tourne la tortue d'un angle droit, mettons vers la droite, on avance alors de la largeur de la marche, et on tourne la tortue d'un angle droit, vers la gauche cette fois-ci pour qu'elle regarde à nouveau vers le

haut de l'écran. Voyons cela : décidons par exemple que la hauteur de la contremarche est 12 et la largeur de la marche 15. On sait qu'un angle droit mesure 90 degrés.

Nous avons maintenant tout ce qu'il nous faut. Pour aller un peu plus vite, au lieu d'appuyer sur [RETURN] après chaque commande, nous allons taper sur une seule ligne les quatre commandes qui permettent de tracer la marche, en séparant comme toujours les mots par un espace.

AVANCE 12 DROITE 90 AVANCE 15 GAUCHE 90

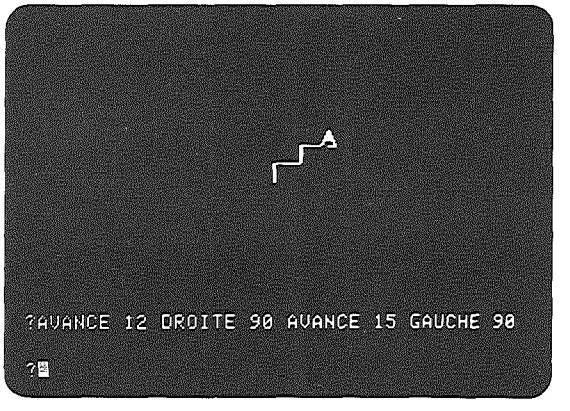
Appuyez sur [RETURN] maintenant, et voilà la marche tracée.

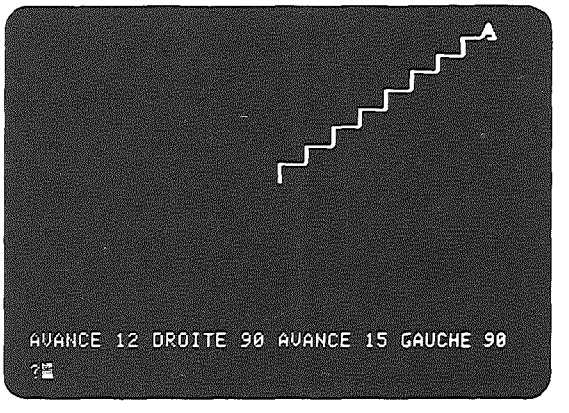


Pour avoir une deuxième marche tracée, il suffit de refaire exactement la même chose. Mais au lieu de le taper à nouveau, nous allons ramener le contenu du *tampon*, c'est-à-dire ce qui a été écrit sur la ligne précédente, en tapant [CTRL] **P** : la ligne s'affiche à nouveau, et le curseur se trouve maintenant à la droite de cette ligne ; appuyez sur

tampon

RETURN et voilà une deuxième marche. Vous pouvez recommencer ainsi autant de marches que vous voulez... dans les limites de l'écran.





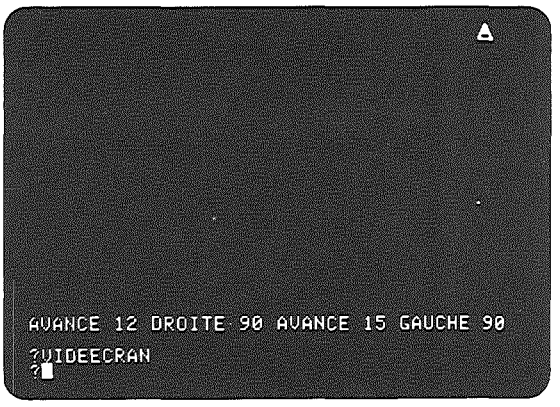
Attention, si vous tapez par mégarde plusieurs fois de suite sur RETURN, *le tampon est vide, et* CTRL **P** *ne ramène plus rien, puisqu'il n'y a plus rien sur la ligne précédente !*
Il n'y a aucun inccnvénient à écrire soit un ordre par ligne (RETURN *après chaque commande), soit plusieurs ordres sur la même ligne, séparés par des espaces. Chacune des commandes est exécutée, en commençant par le début de la ligne, à gauche sur l'écran. Si une commande sur la ligne produit une erreur et ne peut être exécutée, LOGO exécute toutes les commandes qui précèdent, et, en arrivant à la commande incorrecte, fournit un message d'erreur : les commandes qui suivent ne sont pas exécutées.*

Une ligne contenant beaucoup de commandes peut être plus difficile à relire, et donc source de confusion : par conséquent, il vaut mieux ne pas écrire des lignes trop longues. En particulier, il est conseillé d'éviter des suites d'ordres qui dépassent la ligne d'affichage à l'écran (quarante caractères), de manière à conserver une bonne lisibilité. Mais en cas de besoin, on peut dépasser cette longueur : il faut alors entrer ses commandes sans se préoccuper de la mise en page (LOGO le fait pour vous), et surtout ne pas mettre de trait d'union en fin de ligne ! Le tampon d'édition peut contenir jusqu'à 255 caractères ; si vous essayez d'en entrer davantage, les caractères que vous essaierez de rentrer en plus ne s'afficheront plus.

Pas trop mal cet escalier, mais c'est un peu dommage qu'il commence ainsi au centre de l'écran. Si nous partions de l'endroit où nous sommes (en haut à droite de l'écran, la tortue regardant vers le haut), pour tracer les marches en descendant ?

Bien entendu, il n'est pas question cette fois-ci de taper DESSINE, qui nous remettrait au centre. Il faut seulement se débarrasser des tracés précédents, sans modifier ni la position, ni la direction de la tortue. La commande qui réalise cela est VIDEECRAN, *en un seul mot.*

VIDEECRAN



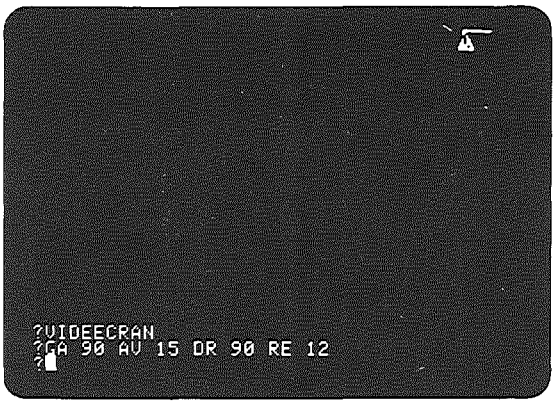
Encore une indication : beaucoup de mots en LOGO ont une forme courte, une abréviation. Ainsi, il suffira de taper **VE** au lieu de VIDEECRAN, **AV**

abréviation

pour AVANCE, et de même **RE**, **DR**, **GA**, à la place de RECULE, DROITE et GAUCHE respectivement.

La marche qui descend s'obtient par exemple avec

GA 90 AV 15 DR 90 RE 12



REPETE

Et puisqu'il nous a fallu 8 marches pour aller du centre au bord supérieur de l'écran, parions qu'il nous faudra 16 marches pour avoir l'escalier complet. REPETE est un mot utilisé par LOGO (on dit aussi un *primitif*), qui doit être suivi du nombre de répétitions et de la liste des commandes à répéter indiquée *entre crochets :*

REPETE 16 [GA 90 AV 15 DR 90 RE 12]

SHIFT **N** *(Apple II)* SHIFT **M**

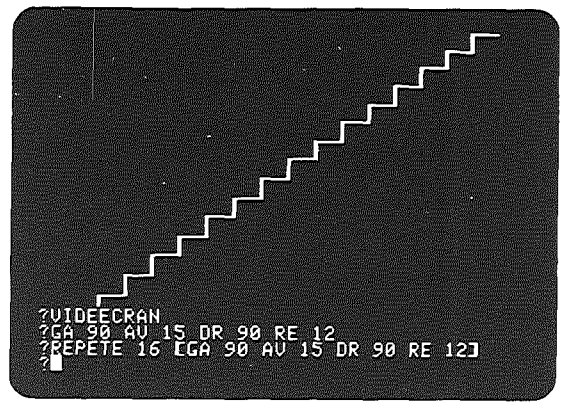
Maintenant RETURN ,
et pour voir tout l'escalier, CTRL **F**.

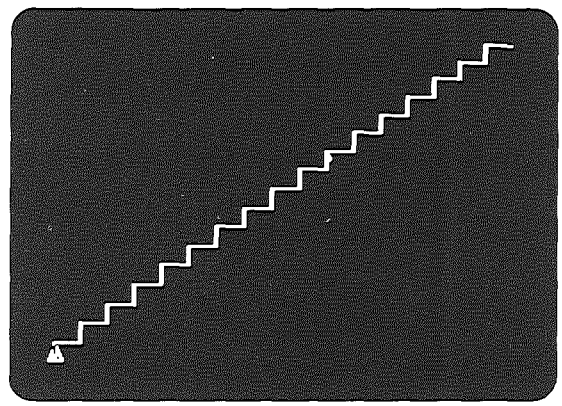
Si on veut n'avoir que le tracé de l'escalier, sans le triangle de la tortue au pied de la marche inférieure, CACHETORTUE (en abrégé CT) la fait disparaître.

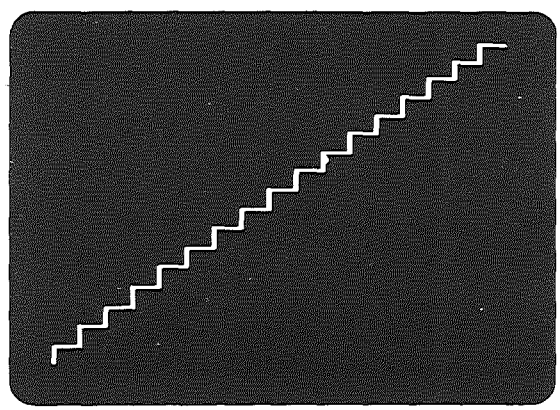
CACHETORTUE

MONTRETORTUE

Pour la faire reparaître, MONTRETORTUE (en abrégé MT).


?VIDEECRAN
?GA 90 AV 15 DR 90 RE 12
?REPETE 16 [GA 90 AV 15 DR 90 RE 12]
?





- ***L'effet d'enroulement***

Reprenons une page propre, grâce à l'ordre DESSINE.
Puis tournons un peu la tortue :

DROITE 20

Faisons-la avancer d'un très grand nombre de pas :

AVANCE 1000

Il se produit un phénomène curieux : plusieurs lignes apparaissent sur l'écran, toutes parallèles.

Essayons de voir un peu mieux ce qui s'est passé.

DESSINE

DROITE 20

AVANCE 200

CTRL **F.**

C'est un peu comme si la feuille de dessin, au lieu d'être collée sur l'écran, était légèrement décalée vers l'avant : quand la tortue arrive au bord de la

feuille , elle glisse derrière (elle ne compte pas de pas, comme quand on dérape sur la glace en hiver), et se rattrape à l'autre bord de la feuille, où elle continue à compter ses pas.

Ce phénomène se produit aussi avec les bords gauche et droite de l'écran.

Vous pouvez aussi penser à ce qui se passe quand vous ficelez un paquet avec un ruban, qui passe tout autour de votre paquet (donc alternativement dessus et dessous) : on appelle cette propriété *l'effet d'enroulement.* Remarquez que tout en glissant derrière la feuille, la tortue reste les yeux fixés vers la même direction, et qu'elle n'a pas changé de direction en réapparaissant sur le bord opposé : c'est pour cela que nos droites tracées avec AVANCE 1000 étaient toutes parallèles.

Si vous trouvez l'effet d'enroulement peu pratique, vous pouvez garder la tortue sur une page bien définie, grâce au primitif PAGE. Lorsqu'elle arrivera près des bords de la feuille, l'ordinateur vous avertira que la tortue ne peut plus continuer, en vous envoyant le message :

```
TORTUE HORS ECRAN
```

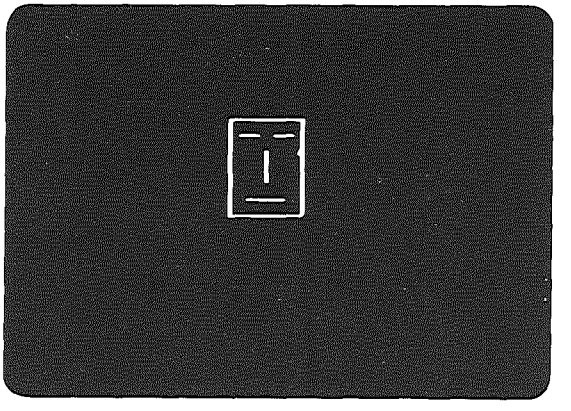
et en refusant d'exécuter l'instruction qui ferait sortir des limites de la page. On revient à l'effet d'enroulement grâce au primitif ENROULE.

• *Quelques primitifs complémentaires*

Comment dessiner la tête ci-dessous ?
Partons du bas du nez :

```
AVANCE 2Ø
```

paraît un bon début. Il faut ensuite tracer les yeux :



mais entre le nez et les yeux, il ne doit pas y avoir de trait : pour cela, il suffit de lever la plume (cf. page 13) de la tortue :

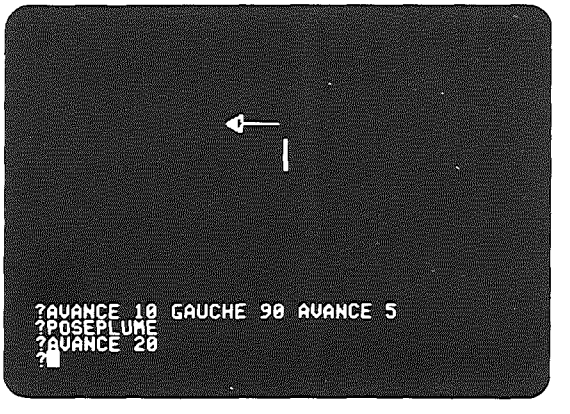
LEVEPLUME

```
LEVEPLUME
AVANCE 1Ø GAUCHE 9Ø AVANCE 5
```

Maintenant nous sommes en bonne position : en reposant la plume de la tortue, nous allons pouvoir tracer les yeux :

POSEPLUME

```
POSEPLUME
AVANCE 2Ø
```

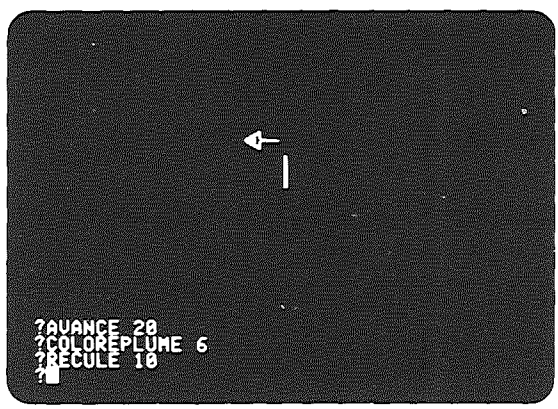


Zut : cet œil, aussi grand que le nez, a vraiment l'air démesuré. Faut-il tout effacer et repartir à zéro ? Non : nous pouvons remplacer la plume traçante de notre tortue par un effaceur, en tapant :

COLOREPLUME 6

Maintenant, repassons sur la partie à enlever :

RECULE 10



puis remettons notre plume en état de tracer,

COLOREPLUME 1 **COLOREPLUME**

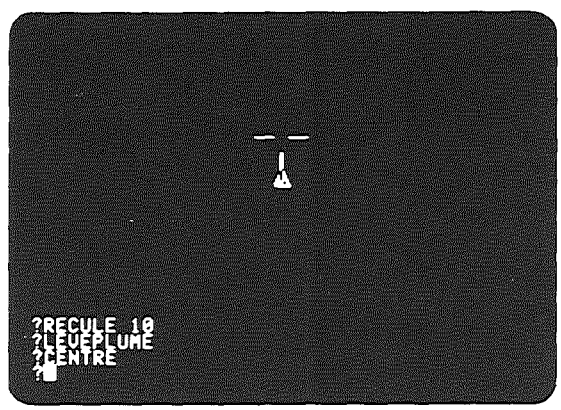
et reprenons la suite de notre dessin :

LEVEPLUME RECULE 20
POSEPLUME
RECULE 10

Les yeux étant faits, passons à la bouche ; nous allons, sans tracer, revenir à notre point de départ, au bas du nez :

LEVEPLUME

CENTRE CENTRE



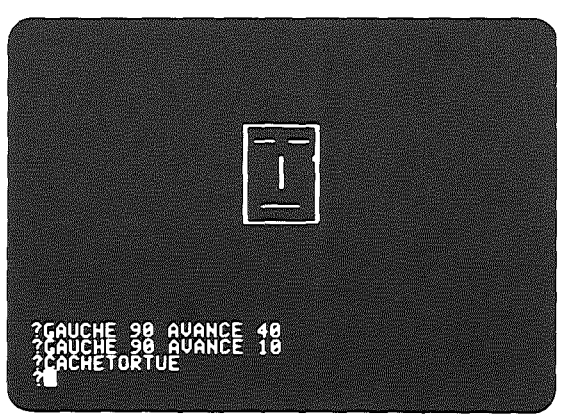
Remarquons que le primitif CENTRE remet la tortue le nez en l'air, regardant vers le haut de l'écran.

RECULE 10 GAUCHE 90 POSEPLUME

AVANCE 10 RECULE 20

Il ne reste plus à faire que le tour de la tête :

LEVEPLUME RECULE 10 DROITE 90 POSEPLUME



Nous voici au bon endroit :

```
AVANCE 50 GAUCHE 90 AVANCE 40
GAUCHE 90 AVANCE 60
GAUCHE 90 AVANCE 40
GAUCHE 90 AVANCE 10
CACHETORTUE
```

Le primitif CENTRE n'a pas de forme courte. Par contre, LEVEPLUME s'abrège en LP, et POSEPLUME en PP. COLOREPLUME devient CP. A l'inverse, il existe des primitifs qui font la même chose que CTRL **F** *et* CTRL **S** *: ce sont PLEINECRAN et ECRANMIXTE qui n'ont pas d'abréviation.*

• *Pour aller plus loin*

1. Combien y a-t-il de pas de tortue du centre de l'écran au bord supérieur ? Du centre au bord de la partie texte sur l'écran mixte ? Du centre à la partie inférieure de PLEINECRAN ? Du centre au bord gauche ? Du centre au bord droit ?

2. Combien y a-t-il de pas de tortue du coin inférieur gauche (en ECRANMIXTE) au coin supérieur droit ? Du coin inférieur gauche (en PLEINECRAN) au coin supérieur droit ?

3. Dessiner un carré, un rectangle.

4. Essayer un triangle.

5. Modifier une ligne rappelée par CTRL **P**, puis taper à nouveau CTRL **P** avant RETURN tapez encore CTRL **P** après RETURN (attention aux espaces).

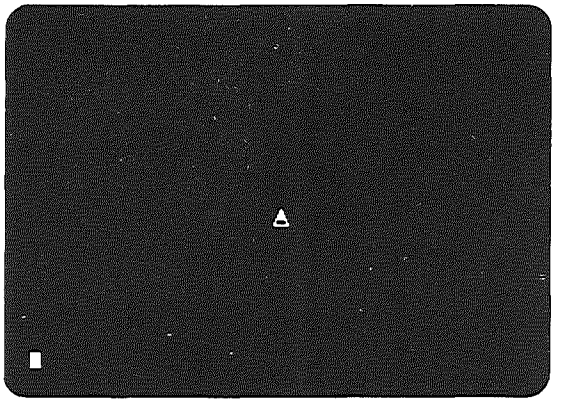
6. Dessiner des lettres, des chiffres.

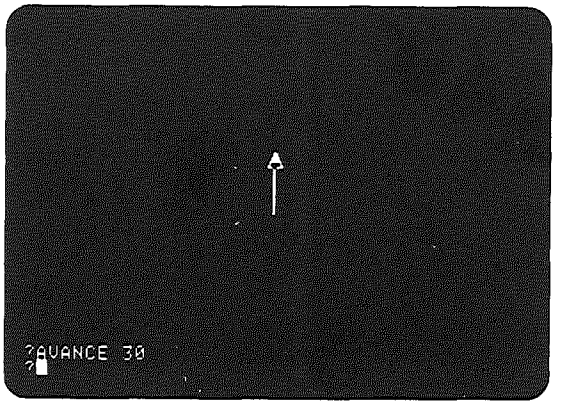
CHAPITRE III

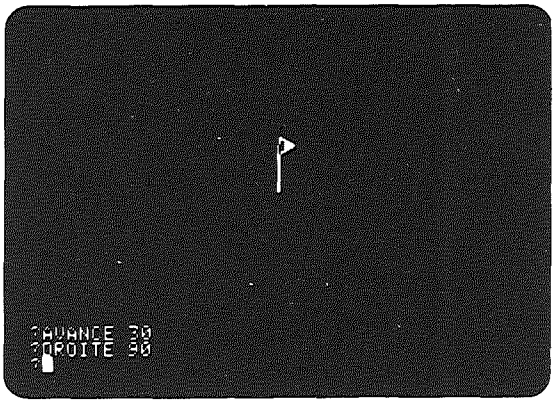
ENSEIGNER A LOGO DES MOTS NOUVEAUX

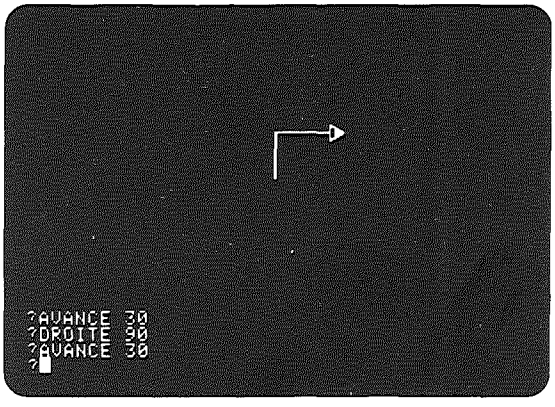
Voyons ensemble un des exercices proposés à la fin du chapitre précédent : dessiner un carré, disons de trente pas de côté pour se fixer les idées.

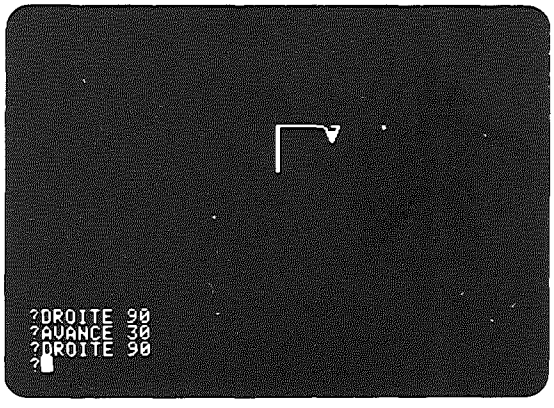
```
DESSINE
AVANCE 30 DROITE 90
AVANCE 30 DROITE 90
AVANCE 30 DROITE 90
AVANCE 30
```





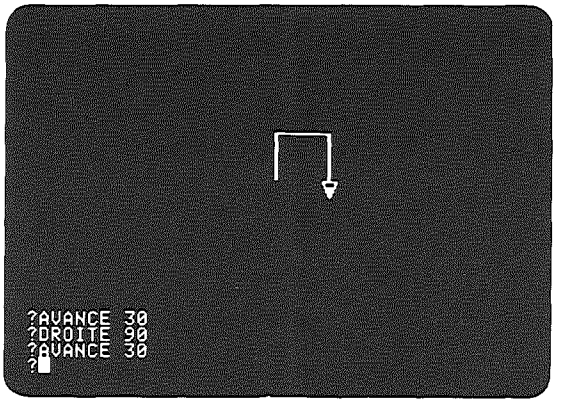

?AVANCE 30
?DROITE 90
?

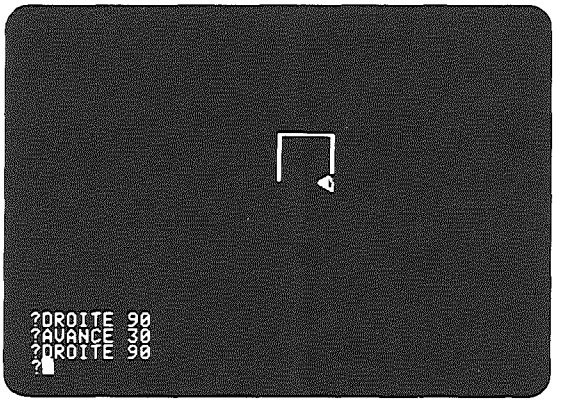

?AVANCE 30
?DROITE 90
?AVANCE 30
?

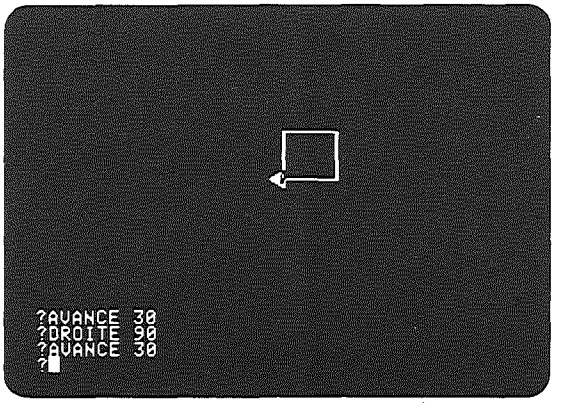

?DROITE 90
?AVANCE 30
?DROITE 90
?

CHAPITRE III

ENSEIGNER A LOGO DES MOTS NOUVEAUX







Donc la tortue peut tracer un carré, puisque nous venons de le lui faire faire. Pourtant, si vous effacez l'écran et écrivez

DESSINE CARRE

elle ne trace rien du tout, mais retourne le message

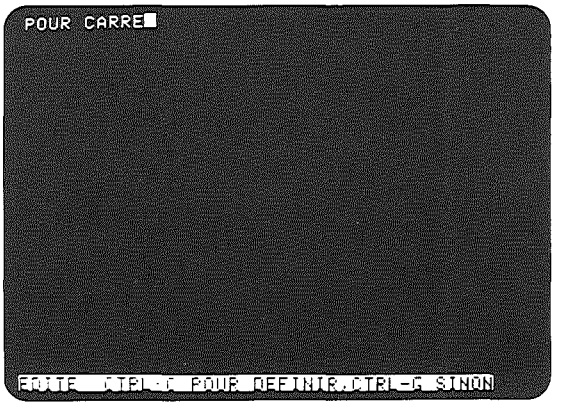
CARRE N'A PAS ETE DEFINI

- ***Enseigner la procédure CARRE***

Nous allons maintenant enrichir le vocabulaire de l'ordinateur, augmenter son langage. En effet, tous les mots que vous tapez au clavier sont analysés par la machine, qui cherche chacun des termes que vous avez employés dans une liste de choses qu'elle sait déjà faire (ces mots sont appelés les *primitifs* du langage LOGO). Si vous utilisez un terme qui ne figure pas dans cette liste, l'ordinateur ne peut pas deviner ce que vous attendez de lui ; aussi, il faut le lui expliquer. tapez

primitif

POUR CARRE (suivi de la touche RETURN)



CHAPITRE III

ENSEIGNER A LOGO DES MOTS NOUVEAUX

Aussitôt que vous appuyez sur la touche RETURN, l'écran mixte disparaît : il n'y a plus de tortue, les commandes précédemment affichées sont invisibles, il n'y a même plus de signe ? mais seulement, en haut de l'écran, la ligne

```
POUR CARRE
```

avec le curseur qui clignote, et en bas de l'écran un bandeau avec ces mots

```
EDITE : CTRL-C POUR DEFINIR, CTRL-G SINON
```

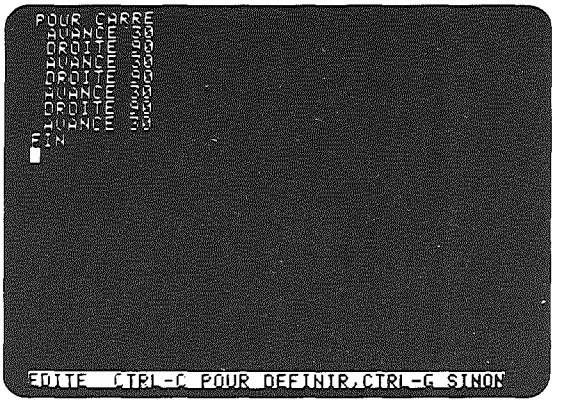
Vous êtes maintenant en mode éditeur *(on dit aussi* mode programme *ou* mode procédure*) et non plus en* mode direct *(ou* mode pilotage, mode conversationnel, mode immédiat*), comme jusqu'à présent.*

procédure

A chaque fois que vous enseignerez un mot nouveau à LOGO, vous retrouverez cet écran d'édition avec le même bandeau en bas de page.
Appuyez sur la touche RETURN pour aller à la ligne, et taper les commandes qui permettront à la tortue de tracer le carré. Quand vous avez fini, allez à la ligne et tapez le mot FIN.
L'écran se présente ainsi :

```
POUR CARRE
AVANCE 30
DROITE 90
AVANCE 30
DROITE 90
AVANCE 30
DROITE 90
AVANCE 30
FIN
```

□



Appuyez maintenant, comme vous y invite l'inscription sur le bandeau, sur CTRL **C** (C comme... Compris ?), l'écran d'édition disparaît, et en haut de l'écran vous voyez

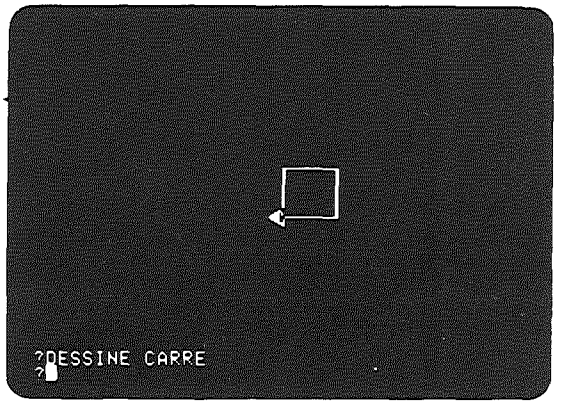
UN MOMENT, MERCI...

et au bout d'un très court instant

CARRE EST DEFINI

Si maintenant nous lui demandons

DESSINE CARRE

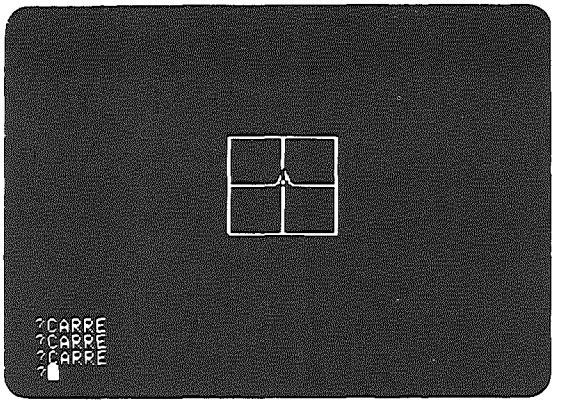


plus de message d'erreur, mais en un rien de temps le carré est tracé. Le mot CARRE est maintenant devenu partie intégrante de son vocabulaire, au même titre que LEVEPLUME ou AVANCE, et peut être réutilisé :

CARRE

CARRE

CARRE



Nous voici avec une fenêtre : on peut d'ailleurs lui apprendre ce mot :

POUR FENETRE (n'oubliez pas RETURN)

REPETE 4 [CARRE]

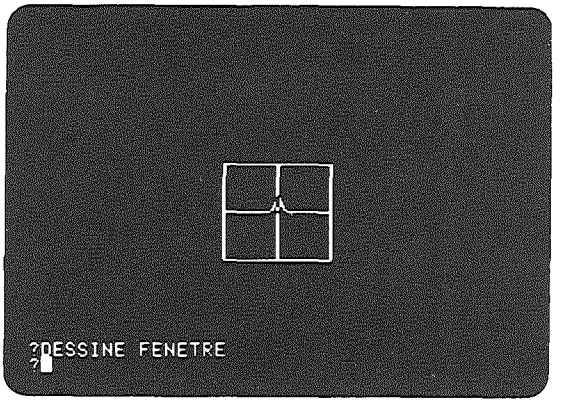
FIN (n'oubliez pas CTRL **C**)

Là encore, un message signale

FENETRE EST DEFINI

Remarquons que, pendant que nous avons défini cette procédure, le dessin du carré a disparu. Il en est toujours ainsi, l'entrée en mode éditeur efface tout ce qui se trouve sur l'écran du mode direct.

DESSINE FENETRE



A nouveau, nous pouvons utiliser ce mot dans un autre tracé :

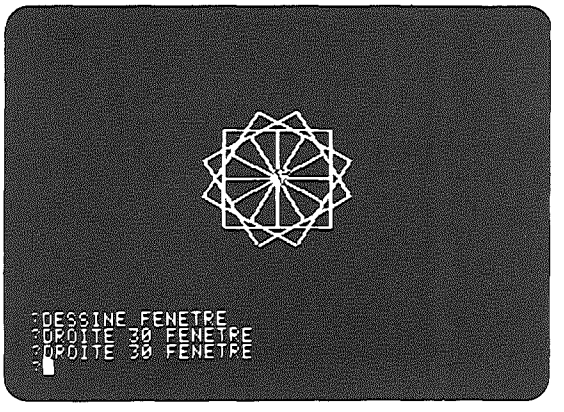
DROITE 3Ø FENETRE

DROITE 3Ø FENETRE

Et voici une fleur

POUR FLEUR

FENETRE DROITE 3Ø FENETRE



DROITE 3Ø FENETRE

FIN

super-procédure *Enseigner un mot nouveau à LOGO se dit aussi* définir une procédure. *Dans le cas d'une procédure comme ici FENETRE, qui utilise une autre procédure précédemment définie (CARRE), on dit parfois que FENETRE est une* super-procédure.

• *Utilisation de l'éditeur*

Maintenant que nous avons un carré, si nous changions un peu de forme, pour avoir un rectangle de vingt pas sur quarante ? La procédure est presque la même.

POUR RECTENGLE RETURN

Ah, mais non ! Si nous lui apprenons le mot rectangle avec une faute, c'est cette orthographe-là qui va être retenue, et ensuite nous risquons de ne plus nous y retrouver. Le bandeau de bas de page nous indique quoi faire : nous ne voulons pas définir RECTENGLE, alors il suffit d'appuyer sur CTRL **G** au lieu de CTRL **C**.

Cette fois-ci le retour au mode direct est accompagné du message

ARRET !

Reprenons :

POUR RECTANGLE RETURN

AVANCE 2Ø

DROITE 9Ø

AVANCE 4Ø

DROITE 9Ø

AVANCE 2Ø

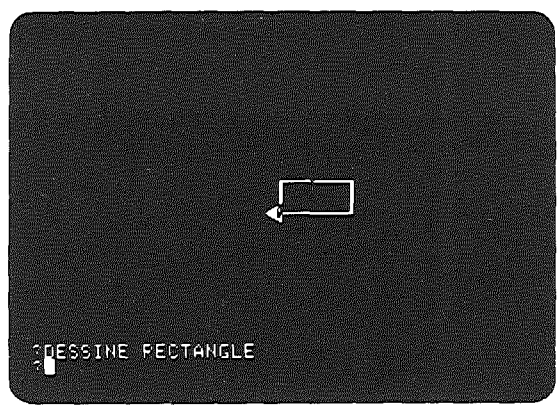
DROITE 9Ø

AVANCE 4Ø

FIN CTRL **C**

Maintenant :

DESSINE RECTANGLE



Réflexion faite, ce rectangle est bien petit : il aurait été mieux avec une taille de quarante pas sur soixante. Ce n'est pas difficile à modifier : le primitif qui permet cela est EDITE , suivi du nom (on dit plutôt du *titre*) de la procédure à modifier :

EDITE RECTANGLE RETURN

Nous voici encore une fois en mode éditeur, et tout le texte de la procédure est affiché. Nous allons pouvoir nous déplacer dans ce texte en utilisant les touches fléchées → et ← et la touche REPT. Remplaçons par exemple dans la première ligne le **2** de **2Ø** par un **4** : nous avons notre **4Ø**. Maintenant, appuyez sur CTRL **N** : le curseur descend d'une ligne ; encore une fois CTRL **N**, et vous êtes en bonne position pour corriger le **4** en **6** : bien sûr,

les touches [ESC] et [CTRL] **D** vous fournissent le moyen d'effacer les caractères que vous voulez changer. Continuez à utiliser [CTRL] **N** pour descendre de ligne en ligne. Si par hasard vous êtes descendu trop bas, [CTRL] **P** vous permet de remonter.

Signalons encore deux touches de contrôle qui sont souvent bien pratiques : [CTRL] **A** qui ramène le curseur au début de la ligne, et [CTRL] **E** qui ramène le curseur en fin de ligne.

Vous trouverez la description complète des touches de contrôle de l'éditeur dans le manuel de référence, mais celles que vous venez de voir sont de loin les plus utiles.

Quand vos corrections sont terminées,

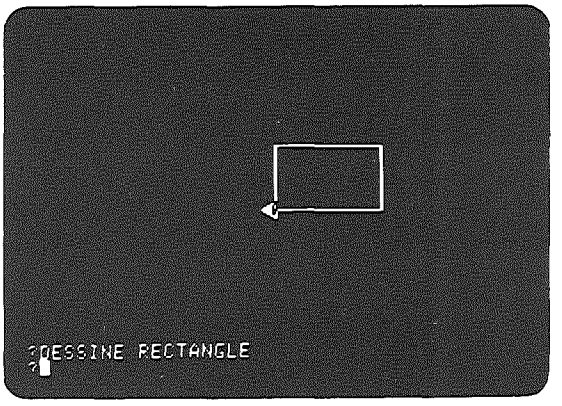
```
POUR RECTANGLE
AVANCE 40
DROITE 90
AVANCE 60
DROITE 90
AVANCE 40
DROITE 90
AVANCE 60
FIN
```

tapez [CTRL] **C**.

*Vous pouvez taper [CTRL] **C** de n'importe quel endroit de l'éditeur, quand vous êtes en train de modifier une procédure précédemment définie. Il n'est nul besoin de positionner le curseur après le mot FIN, ni même en bout de ligne, avant de taper [CTRL] **C**.*

Et maintenant :

```
DESSINE RECTANGLE
```



C'est bien, mais finalement c'est un peu dommage de ne pas avoir gardé aussi le petit rectangle. Il aurait fallu y penser plus tôt pour s'éviter un peu de travail, mais rien n'est perdu : tapez à nouveau

EDITE RECTANGLE **EDITE**

faites les modifications inverses, et, avant de taper CTRL **C**, remontez jusqu'à la ligne de titre et, par exemple, modifiez le **A** de RECTANGLE en un **E** : nous avons maintenant :

```
POUR RECTENGLE
AVANCE 2Ø
DROITE 9Ø
AVANCE 4Ø
DROITE 9Ø
AVANCE 2Ø
DROITE 9Ø
AVANCE 4Ø
FIN
```

Tapez CTRL **C** : le message affiché est

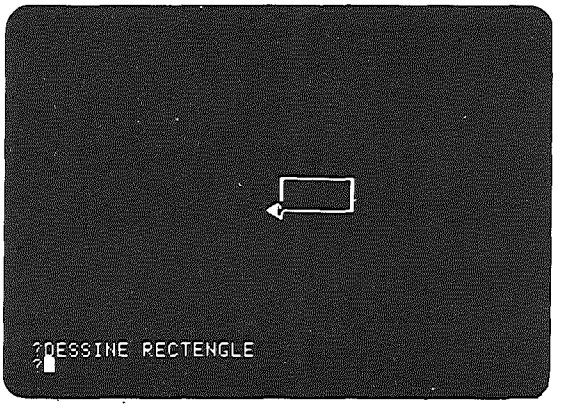
UN MOMENT, MERCI...

RECTENGLE EST DEFINI

Elle a donc appris un nouveau mot.

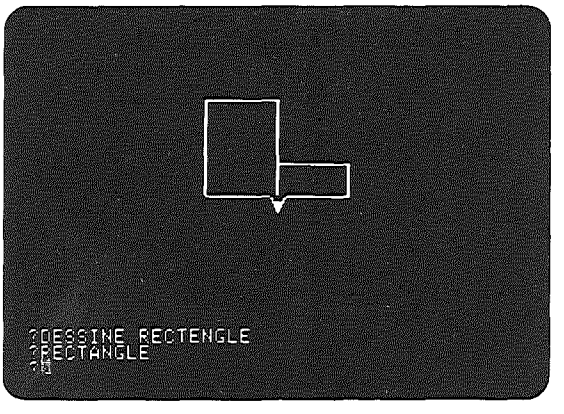
DESSINE RECTENGLE

voilà le petit rectangle,



RECTANGLE

voilà le grand.
Nous avons ainsi deux copies légèrement différentes.



• *Procédures paramétrées*

Et si nous voulions changer à nouveau les dimensions des côtés ? Il y a un moyen de ne pas modifier le nom de la procédure pour chaque valeur, tout en se gardant la possibilité de choisir cette valeur au moment de l'exécution.

Définissons une nouvelle procédure, que nous allons appeler CARREAU, par exemple, et qui va nous laisser la possibilité de remettre à plus tard la détermination des valeurs des deux côtés. Pour celà, nous allons remplacer les nombres 2Ø et 4Ø (ou 4Ø et 6Ø) par deux *paramètres* que nous nommerons COTE1 et COTE2.

paramètre

Nous aurions pu désigner ces paramètres par les lettres X et Y, ou par toute autre dénomination, pourvu qu'elle soit écrite en un seul mot, c'est-à-dire sans contenir d'espace. Ainsi, PREMIER.COTE et DEUXIEME.COTE feraient aussi bien l'affaire.

Un paramètre, c'est un peu comme une boite, dans laquelle on mettra une valeur pour permettre l'exécution de la procédure. Bien sûr, il faut indiquer que les mots COTE1 et COTE2 ne sont pas des procédures, c'est pourquoi on les fait précéder immédiatement (pas d'espace) du signe **:** Ce signe permet à LOGO de distinguer un mot qui désigne une procédure, d'un mot qui désigne un paramètre. Il faut indiquer dès la ligne de titre que l'on va utiliser des paramètres, en donnant leurs noms :

```
POUR CARREAU :COTE1 :COTE2
AVANCE :COTE1 DROITE 9Ø
AVANCE :COTE2 DROITE 9Ø
AVANCE :COTE1 DROITE 9Ø
AVANCE :COTE2 DROITE 9Ø
FIN
```

CTRL **C** nous ramène au mode direct, où nous trouvons le message

UN MOMENT, MERCI...
CARREAU EST DEFINI

Cependant

DESSINE CARREAU

ne produit pas de dessin, mais l'ordinateur nous indique

IL MANQUE QUELQUECHOSE APRES CARREAU

exactement de la même façon qu'écrire simplement

DROITE

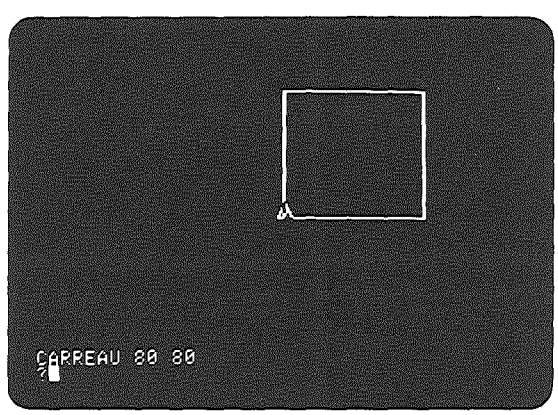
amène la remarque

IL MANQUE QUELQUECHOSE APRES DROITE

En effet, il faut indiquer les valeurs à donner à COTE1 et COTE2

CARREAU 2Ø 4Ø

trace maintenant le petit RECTENGLE de tout-à-l'heure, et



CARREAU 4Ø 6Ø

le RECTANGLE précédent.

Rien ne nous interdit de donner la même valeur aux deux côtés :

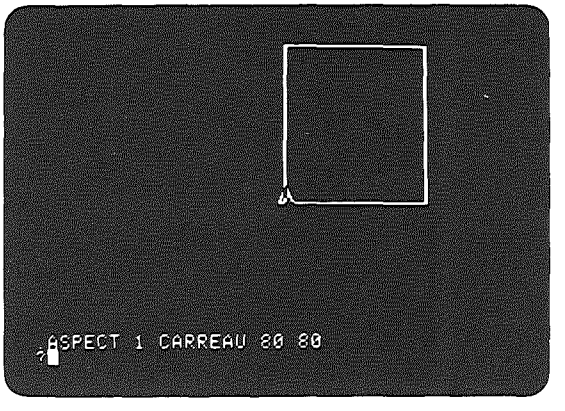
CARREAU 8Ø 8Ø

dessine un carré.

Vous trouverez peut-être que ce carré a plutôt l'air d'un rectangle que d'un carré. Le problème vient de votre téléviseur : en général, en effet, le balayage ne donne pas des unités égales en hauteur et en largeur. Un primitif peut vous permettre de changer l'aspect de vos tracés.
Tapez

.ASPECT 1 CARREAU 8Ø 8Ø

.ASPECT



Si l'allongement est trop grand maintenant, essayez

.ASPECT .9 CARREAU 8Ø 8Ø

(voir page suivante)

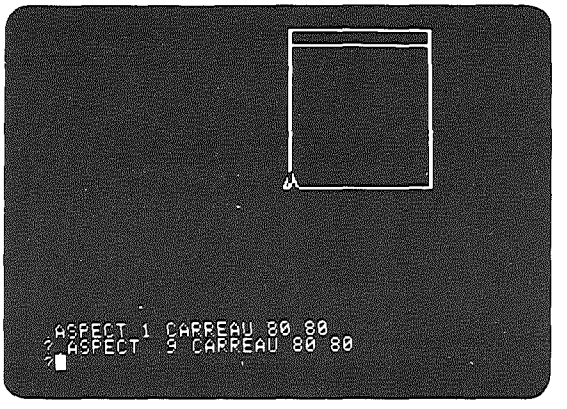
si c'est au contraire trop aplati, essayez

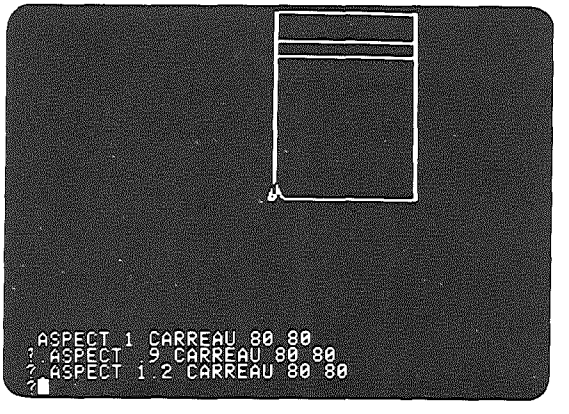
.ASPECT 1.2 CARREAU 8Ø 8Ø

Quand vous aurez trouvé la valeur qui vous convient, notez-la pour la réutiliser à chaque fois que vous reprendrez des activités avec LOGO.

CHAPITRE III

ENSEIGNER A LOGO DES MOTS NOUVEAUX





- ***L'espace de travail***

Les procédures que vous avez créées restent à votre disposition tant que vous n'éteignez pas votre micro-ordinateur. (Vous pouvez aussi les sauvegarder si vous pensez vous en resservir ultérieurement, voir chapitre 9)
Mais quand on a écrit un assez grand nombre de procédures, on finit par ne plus très bien savoir de quoi on dispose. Il est prudent de tenir un cahier de bord, et d'y noter tous les renseignements utiles :

la valeur qu'il faut donner à .ASPECT pour que les carrés n'aient pas l'air de rectangles, les textes des procédures avec un petit croquis qui rappelle ce que la tortue dessine, etc...
Vous pouvez aussi faire afficher sur l'écran un certain nombre de renseignements :

IMPRIME TITRES **IMPRIMETITRES**

(en deux mots, la forme courte est IMTS en un seul mot), affiche la liste des titres de procédures. Ainsi, en ce moment, nous avons :

POUR CARREAU
POUR RECTENGLE
POUR RECTANGLE
POUR FLEUR
POUR FENETRE
POUR CARRE

Pour inspecter le contenu d'une procédure, il y a bien sûr la possibilité d'éditer cette procédure. Mais quand on revient au mode direct avec CTRL **C**, on constate que les dessins qu'on pouvait avoir sur l'écran ont disparu. On évite cet inconvénient en faisant imprimer la procédure. Ainsi, pour revoir CARREAU

IMPRIME CARREAU **IMPRIME**

Bien sûr, l'énoncé de la procédure est probablement trop long pour tenir sur les quatre lignes du bas de l'écran mixte, si vous utilisez une tortue en ce moment : mais grâce à CTRL **T**, on peut rendre visible le reste de la page de texte sans pour autant perdre les dessins, et on pourra les récupérer par CTRL **S** ou CTRL **F**.

CHAPITRE III

ENSEIGNER A LOGO DES MOTS NOUVEAUX

Parmi les procédures disponibles dans votre espace de travail, certaines peuvent vous paraître désormais sans intérêt : en particulier quand une procédure paramétrée peut couvrir des procédures non paramétrées, comme ici CARREAU couvre RECTANGLE (que l'on obtient en tapant CARREAU 4Ø 6Ø), RECTENGLE, et même CARRE. Il est possible de se débarrasser de ces procédures désormais inutiles :

```
EFFACE
EFFACE RECTANGLE
EFFACE RECTENGLE
```

Maintenant, si nous faisons afficher les titres

```
IMPRIME TITRES
POUR CARREAU
POUR FLEUR
POUR FENETRE
POUR CARRE
```

Attention à ne pas effacer une procédure qui est utilisée comme une sous-procédure dans une autre procédure. C'est la raison pour laquelle nous n'avons pas effacé CARRE, qui est utilisé dans FLEUR et dans FENETRE. En effet, si CARRE peut être remplacé, pour une exécution directe, par CARREAU 3Ø 3Ø, l'éliminer de l'espace de travail provoquerait un message d'erreur lors de l'exécution de FENETRE ou de FLEUR (à moins de modifier, dans ces deux procédures, CARRE par CARREAU 3Ø 3Ø à chaque fois que le mot CARRE figure).

Lorsque nous avons pour la première fois commencé la procédure RECTENGLE, nous l'avions détruite en tapant CTRL **G** *au lieu de* CTRL **C** *pour revenir au mode direct. Si maintenant nous éditons CARRE, par exemple pour changer la longueur du côté de 3Ø à 5Ø pas de tortue, et qu'après ces modifications nous quittons le mode éditeur en tapant CTRL G, la procédure CARRE n'est pas détruite, mais les modifications que nous venons d'y apporter sont ignorées. Ainsi, le côté de CARRE est encore de 3Ø pas de tortue*

- ***Choix des noms de procédures***

On peut donner à une procédure un nom absolument arbitraire : rien n'empêche d'appeler votre CARRE du nom de HENRI, ou ZWXKHT, ou même CERCLE. Toutefois, l'expérience montre qu'on a tout intérêt, ne serait-ce que pour soulager sa propre mémoire, à donner aux procédures des titres descriptifs : même après un arrêt de trois mois, vous retrouverez sans peine ce que fait la procédure CARRE, alors que les trois autres titres vous paraîtront beaucoup plus énigmatiques.

Si une procédure doit beaucoup servir pour elle-même, c'est-à-dire si on pense avoir souvent à taper son titre, il vaut mieux que ce terme soit court. Au contraire, si elle est surtout destinée à être incorporée dans une superprocédure il est préférable de lui donner un titre aussi explicite que possible, même s'il est un peu long.

Le titre doit toujours être *un seul mot*. On pourra rendre plus lisible un terme composé en intercalant des points entre les différentes parties : par exemple

CARRESEMBOITES

est moins parlant que

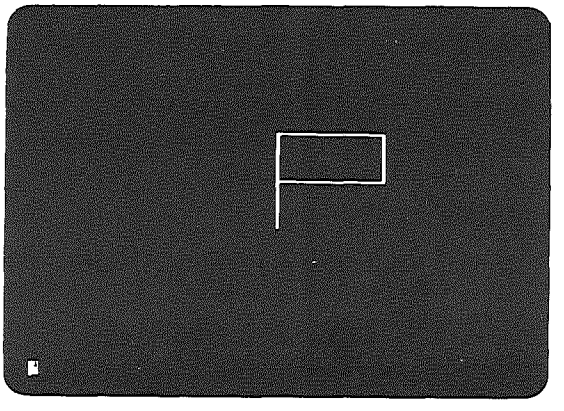
CARRES.EMBOITES

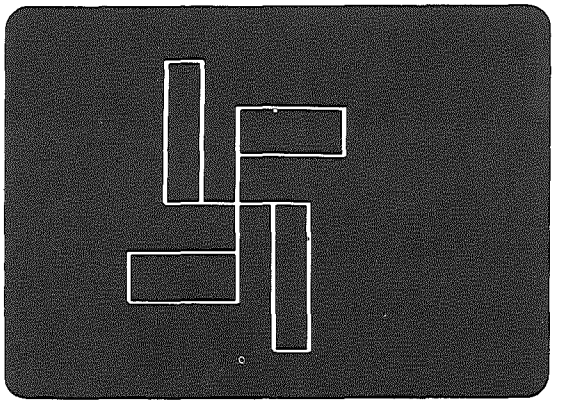
Toutes les remarques précédentes sont également valables pour les noms de paramètres. Pour faciliter la relecture des procédures, vous pouvez leur adjoindre des commentaires. En effet, tout ce que vous tapez à la suite du signe **;** est, à l'exécution, ignoré par la machine, mais restitué quand vous passez en mode éditeur ou quand vous faites imprimer la procédure. (Bien entendu, un commentaire doit *toujours* être écrit en fin de ligne, soit dans le titre, soit au cours de la procédure).

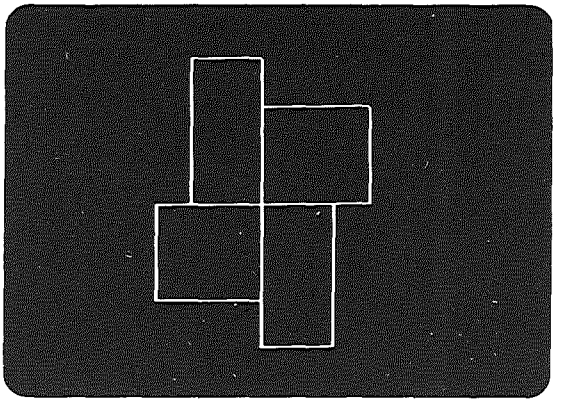
```
; POUR FLEUR ;TAILLE FIXE, 12 PETALES
FENETRE ;CETTE PROCEDURE UTILISE CARRE
REPETE 2 [ DROITE 30 FENETRE ]
FIN
```

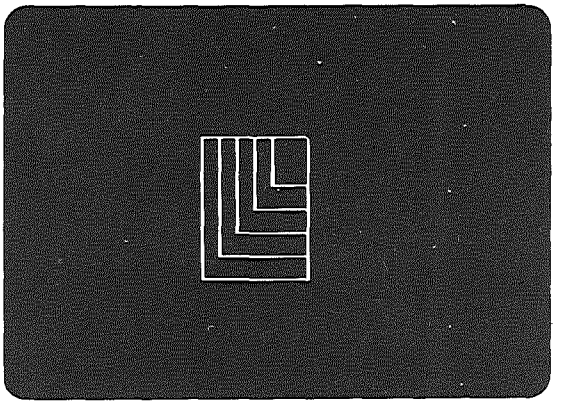
- ***Pour aller plus loin...***

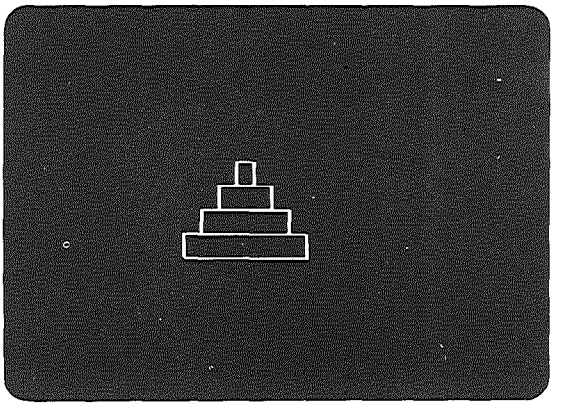
Voici quelques dessins faciles à obtenir à partir de l'une ou l'autre des procédures de ce chapitre.





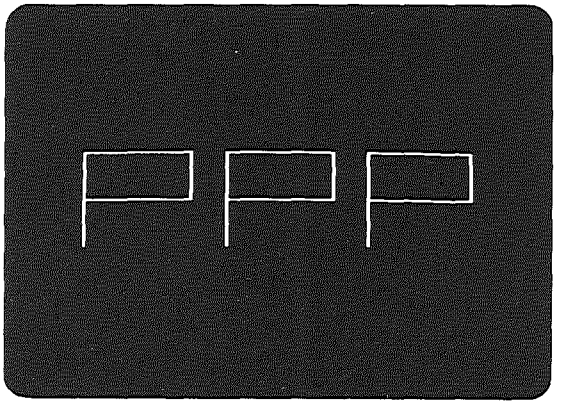


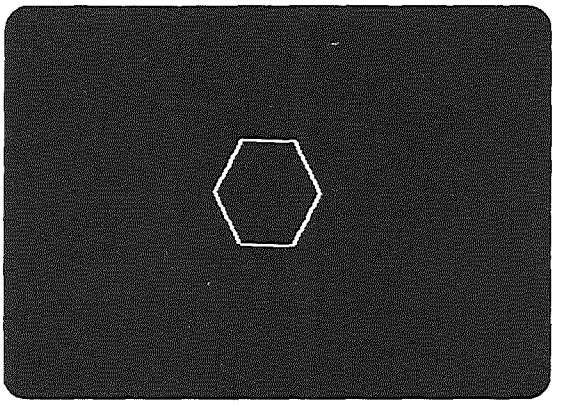


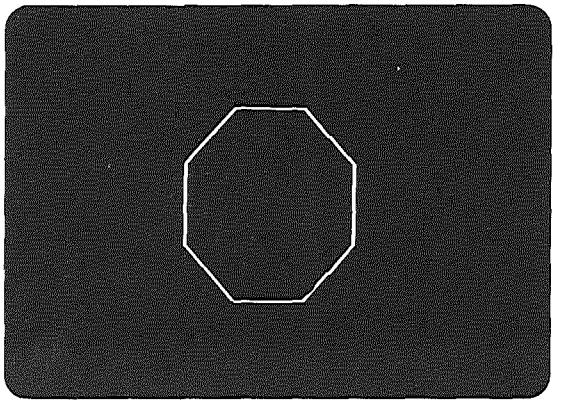


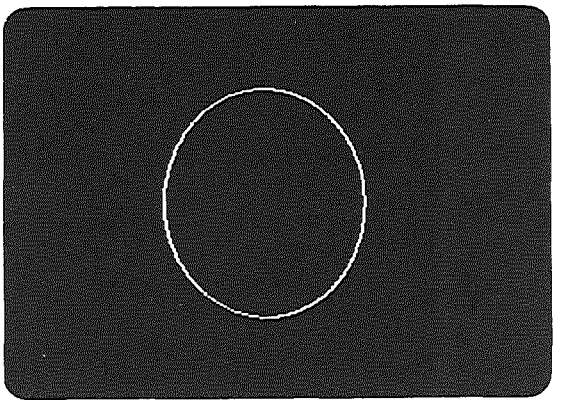
CHAPITRE III

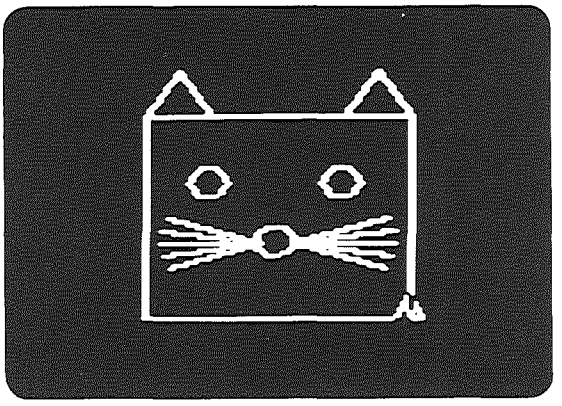
ENSEIGNER A LOGO DES MOTS NOUVEAUX











Ils vous donneront sûrement bien d'autres idées...

CHAPITRE IV

MIEUX QU'UNE CALCULETTE

Dans ce chapitre, nous allons explorer quelques-unes des possibilités de LOGO sur les nombres. LOGO connait les nombres entiers, comme 18 par exemple, jusqu'à 2 147 483 647 mais il faut toujours écrire un nombre d'un seul tenant : ainsi ce plus grand nombre entier compris par LOGO doit être écrit 2147483647 sans aucun espace. LOGO connait également les nombres décimaux (*), mais ceux-ci doivent être écrits avec un point décimal, comme sur les calculatrices de poche, et non avec une virgule : 17.8 au lieu de 17,8 . Ces nombres, entiers ou décimaux, peuvent être positifs (écrits sans signe ou précédés du signe +) ou négatifs (précédés du signe −).

entiers

décimaux

Les nombres décimaux ne doivent pas comporter trop de chiffres. LOGO n'en affiche que 6. Ainsi 1.123456789 est affiché 1.12346 (6 et non 5 comme dernier chiffre, à cause de l'arrondi) ; 123.12345 est affiché 123.123 . Cependant les calculs sont effectués avec une précision un peu supérieure. Ainsi, la somme de 1.1231264 et 2.3451264 est 3.46825 et non 3.46826 que l'on obtiendrait si l'arrondi était pris avant le calcul de la somme.

notation exponentielle

Par le biais de la notation exponentielle *on peut faire des calculs sur des nombres décimaux qui ne rentrent pas dans le modèle précédent. Un nombre écrit en notation exponentielle se compose de trois parties* d'un seul tenant (non *séparées par un espace) :*

mantisse

- *la* mantisse, *qui est soit un nombre entier, soit un décimal écrit sous la forme standard avec un point décimal*
- *une lettre, qui est soit E soit N*

exposant

- *un* exposant, *qui est* obligatoirement un entier postitif.

Les écritures suivantes sont donc incorrectes :

2.3E1.4	*exposant décimal*
3 E5	*espace entre la mantisse et la lettre*
E18	*pas de mantisse*
17.4N 3	*espace entre lettre et exposant*
3,8E5	*décimal écrit avec une virgule au lieu d'un point*
78E-4	*exposant négatif*

Indiquons sur quelques exemples le fonctionnement de la notation exponentielle :

5E3 est le nombre 5ØØØ c'est-à-dire 5 × 1Ø³
1Ø.73E2 est le nombre 1Ø73 c'est-à-dire 1Ø,73 × 1Ø²
1E1 est le nombre 1Ø c'est-à-dire 1 × 1Ø
43N2 est le nombre Ø,43 c'est-à-dire 43 × 10⁻²
3.98N4 est le nombre Ø,ØØØ398 c'est-à-dire 3,98 × 10⁻⁴

(*) Le jargon informatique emploie le terme "nombres réels". Cette terminologie qui ne s'accorde ni avec le sens courant ni avec l'usage mathématique ne doit pas être retenue.

• *Opérations arithmétiques*

LOGO sait faire les quatre opérations avec les nombres entiers.
L'addition utilise le signe **+** , que l'on obtient grâce à la touche [SHIFT] et la touche [+ ;] (juste à gauche des touches marquées des flèches [←] et [→]). La soustraction utilise le signe **–**, situé à droite de la rangée des chiffres. La multiplication utilise le signe * (et non là lettre x) : celui-ci s'obtient avec la touche [SHIFT] et la touche [* :] à gauche de la touche [=] . La division utilise le signe **/** , en bas à droite du clavier. *(Apple IIe voir la documentation)*

Essayons quelques opérations :

3 + 8

QUE DOIS-JE FAIRE AVEC 11

Tiens ! L'opération a été effectuée, le 11 en témoigne, mais pourtant un message d'erreur est renvoyé. En effet, si pour l'utilisateur il est pratiquement évident qu'il veut voir le résultat, cela ne va pas de soi pour la machine, et il faut lui préciser qu'elle doit afficher ce résultat.

AFFICHE 8 + 3

AFFICHE

11

AFFICHE 203 * 87

17661

AFFICHE 987654321 – 123456789

864197532

AFFICHE 7 / 3

2.33333

La division fournit le quotient décimal. Si on veut le quotient entier, on utilisera le primitif QUOTIENT

QUOTIENT

```
QUOTIENT 7 3
```

```
QUE DOIS-JE FAIRE AVEC 2
```

...sans oublier la commande d'affichage !!!

```
AFFICHE QUOTIENT 2468 7
```

```
352
```

- ***Priorité entre les opérations***

Essayons plusieurs opérations à la suite :

```
AFFICHE 18 - 4 * 3
```

```
6
```

Vous aviez peut-être prévu 42, en pensant 18 - 4 cela fait 14, multiplié par 3 donne 42. Un certain nombre de petites calculatrices de poche vous donneraient raison, mais pas toutes.

En effet, quand il y a plusieurs opérations à effectuer dans un calcul, il faut préciser *comment* on doit faire le calcul. Si vous avez trouvé 42, c'est parce que vous avez utilisé, peut-être sans y réfléchir, la *règle de lecture,* qui consiste à effectuer les calculs de gauche à droite, dans l'ordre où ils sont écrits.

Tant qu'on ne travaille qu'avec des nombres écrits à l'aide de chiffres, cela ne présente pas d'inconvénient. Mais quand on fait le calcul littéral (par exemple $2x^2 + 3x - 4 + 5x - 9x^3 + 8x + 7$), ce n'est plus ainsi que l'on procède : on définit alors des règles de priorité entre les opérations. Les calculettes un peu sophistiquées et les ordinateurs adoptent ce système, que l'on appelle parfois système opératoire algébrique.

priorité

La multiplication et la division ont priorité sur l'addition et la soustraction. Si vous voulez que

le calcul s'effectue autrement, il faut l'indiquer à l'aide de parenthèses (que l'on obtient avec les touches SHIFT **8** et SHIFT **9** sur Apple II : attention à ne pas confondre avec les crochets).

```
2 + 3 * 5 - 9 * 5 / 7
    15      45
 17          6.42857
     1Ø.5714
```

Quand les opérations ont même niveau de priorité, les calculs s'effectuent de gauche à droite :

```
354 / 8 * 3
 44.25
     132.75
```

parenthèses

Les parenthèses indiquent les calculs qui doivent être effectués en premier :

```
( 2 + 3 ) * 5 - 9 * ( 5 / 7 )
   5                  Ø.714285
     25       6.42857
        18.5714
```

autre exemple

```
354 / ( 8 * 3 )
     24
  14.75
```

Par ailleurs, si vous utilisez l'opération QUOTIENT, elle sera effectuée après les opérations usuelles :

3 + 8 * QUOTIENT 57 15 - 9

a pour résultat 75, car le premier calcul est 15 – 9

(soit 6), puis QUOTIENT 57 6 (qui est 9) ; ensuite 8 × 9 font 72, et + 3 qui donne 75.
Au contraire,

```
3 + 8 * ( QUOTIENT 57 15 ) - 9
```

a pour résultat 18 : le quotient entier de 57 par 15 est 3, le produit de 8 par 3 est 24, auquel on ajoute 3 et retranche 9.

• *Opérations sur les décimaux*

Alors que les opérations sur des nombres entiers fournissent toujours des résultats exacts (dans la mesure où le résultat ne dépasse pas 2 147 483 647 en valeur absolue), les opérations sur les décimaux se font toujours avec seulement sept chiffres significatifs.
L'addition, la soustraction, la multiplication de nombres entiers donnent un nombre entier si le résultat reste dans les limites acceptables pour les entiers,

```
1234567890 - 123456789
```

```
QUE DOIS-JE FAIRE AVEC 1111111101
```

```
AFFICHE 52341 * 4012
```

```
209992092
```

et un décimal écrit sous forme exponentielle dans le cas contraire :

```
1234567890 + 123456789
```

```
QUE DOIS-JE FAIRE AVEC 2.46913E9
```

```
AFFICHE 54321 * 54321
```

```
2.95077E9
```

Dans certains cas LOGO peut transformer un décimal en notation exponentionelle par exemple 0.01 en 9.99999N3.
La division donne toujours un nombre décimal (c'est un décimal entier, le cas échéant, remarquable par son point décimal) :

```
AFFICHE 45 / 5
```

```
9.
```

L'addition, la soustraction ou la multiplication d'un nombre (entier ou décimal) par un décimal donne toujours un nombre décimal :

1234567890 – 123456789.1

QUE DOIS-JE FAIRE AVEC 1.11111E9

AFFICHE 1234567890 – 45 / 9

1.23457E9

Il peut alors y avoir une perte d'information : en particulier, dans l'exemple précédent, si l'on remplace 45 / 9 par QUOTIENT 45 9 , on a comme résultat 1234567885, dont les cinq derniers chiffres étaient perdus dans le calcul avec des décimaux.

On peut, bien que cela n'offre en général aucun intérêt, transformer facilement un nombre entier en décimal : il suffit de le multiplier par 1. ; le résultat décimal fourni par l'ordinateur est toujours affiché sous forme normalisée, où la mantisse est un décimal qui n'a qu'un chiffre avant le point décimal.

123785241 * 1.

QUE DOIS-JE FAIRE AVEC 1.23785E8

Inversement, il existe des primitifs pour obtenir des entiers à partir de décimaux.

ENTIER donne la partie entière, c'est-à-dire le nombre entier "qui précède le point décimal" :

AFFICHE ENTIER 3.2

3

AFFICHE ENTIER 6.8 ENTIER

6

AFFICHE ENTIER –4.28

–4

AFFICHE ENTIER –6.97

–6

ARRONDI donne l'entier le plus proche :

AFFICHE ARRONDI 3.2

3

AFFICHE ARRONDI 6.8 ARRONDI

7

```
AFFICHE ARRONDI -4.28
-4
AFFICHE ARRONDI -6.97
-7
```

Dans le cas où le résultat serait un nombre trop grand pour être écrit sous forme d'entier, LOGO signale une erreur.

- ***Comparer des nombres : les signes = , < , >***

Dans les exercices de calcul que l'on fait à l'école, on trouve souvent des énoncés sous la forme
4 + 7 =
que tout le monde interprète comme "calcule 4 + 7 et écris le résultat que tu as trouvé". C'est peut-être ainsi que vous auriez pensé à modifier votre demande

```
4 + 7
```

en réponse au message

```
QUE DOIS-JE FAIRE AVEC 11
```

Essayez :

```
4 + 7 =
```

```
IL MANQUE QUELQUECHOSE APRES =
```

Si vous n'aviez pas déjà constaté que LOGO sait calculer, cela aurait pu vous paraître décevant, de se voir demander un résultat quand on s'attend à ce que la machine le fournisse.

```
4 + 7 = 11
```

```
QUE DOIS-JE FAIRE AVEC VRAI
```

Voilà qui est plus intéressant ; essayez maintenant de lui donner un résultat incorrect et demandez-lui d'afficher son avis :

```
AFFICHE 3 * 8 = 12
FAUX
```

Les deux autres signes < et > fonctionnent de la même façon :

```
AFFICHE 152 > 521
FAUX
AFFICHE 12 + 2 < 12 * 2
VRAI
```

Les signes = , < et > sont des *prédicats*, c'est-à-dire des ordres auxquels LOGO répond par VRAI ou par FAUX. Nous verrons au chapitre 6 des exemples d'utilisation de ces prédicats.

• *Sortir un résultat pour le réutiliser*

Essayons de rédiger une procédure qui calcule le carré d'un nombre, c'est-à-dire qui, étant donné un nombre, multiplie ce nombre par lui-même. Ce nombre n'étant pas connu à l'avance, mais à déterminer au moment de l'utilisation, nous avons à écrire une procédure paramétrée.

```
POUR CARRE :NOMBRE
:NOMBRE * :NOMBRE
FIN
```

Si vous avez déjà une procédure CARRE, effacez-la ou changez de titre

Et puis, tant que nous y sommes, une super-procédure pour le cube :

```
POUR CUBE :NOMBRE
CARRE :NOMBRE * :NOMBRE
FIN
```

Essayons sur quelques exemples :

```
CARRE 25
QUE FAUT-IL FAIRE AVEC 625, LIGNE
  :NOMBRE * :NOMBRE
AU NIVEAU 1 DE CARRE
```

Voilà qui vous rappelle quelque chose ? Ah oui, il faut faire afficher. Corrigeons donc les deux procédures avant de reprendre les essais :

```
POUR CARRE :NOMBRE
AFFICHE :NOMBRE * :NOMBRE
FIN
```

et

```
POUR CUBE :NOMBRE
AFFICHE CARRE :NOMBRE * :NOMBRE
FIN
```

Allons-y

```
CARRE 5
25
CARRE 100
10000
```

Ça a l'air de marcher parfaitement ; passons au cube :

```
CUBE 5
625
CARRE N'A RIEN SORTI, LIGNE
```

```
AFFICHE CARRE :NOMBRE * :NOMBRE
AU NIVEAU 1 DE CUBE
```

Là, ça ne va pas du tout. D'où provient le 625, et que signifie le message ? Voyons ... 625 est le carré de 25, qui est lui-même 5 fois 5. Nous retrouvons donc le même pépin qu'avec QUOTIENT : les calculs signalés par un signe d'opération sont effectués en premier. Nous ne voulions pas

```
CARRE ( :NOMBRE * :NOMBRE)
```

mais

```
( CARRE :NOMBRE ) * :NOMBRE
```

Modifions donc la procédure CUBE, en insérant les parenthèses, puis essayons à nouveau :

```
CUBE 5
25
( N'A RIEN SORTI, LIGNE
  AFFICHE (CARRE :NOMBRE ) * :NOMBRE
AU NIVEAU 1 DE CUBE
```

Le message ressemble au précédent, et n'est pas plus compréhensible. Essayons de modifier CUBE en supprimant AFFICHE en début de ligne

```
POUR CUBE :NOMBRE
(CARRE :NOMBRE.) * :NOMBRE
FIN
```

pour voir si elle va faire le calcul du cube, et seulement nous demander ce qu'il faut faire du résultat :

```
CUBE 5
25
```

```
IL MANQUE QUELQUECHOSE AVANT * , LIGNE
   ( CARRE :NOMBRE ) * :NOMBRE
AU NIVEAU 1 DE CUBE
```

Mais oui, bien sûr ... Le carré de 5 a été affiché ; la parenthèse ne produit donc aucun nombre qui puisse être multiplié par 5 à nouveau. Donc l'affichage ne doit pas se faire à l'intérieur de la procédure CARRE. On pourrait peut-être supprimer le mot AFFICHE

```
POUR CARRE :NOMBRE
:NOMBRE * :NOMBRE
FIN
```

et le mettre à l'extérieur

```
AFFICHE CARRE 5
QUE FAUT-IL FAIRE AVEC 25, LIGNE
   :NOMBRE * :NOMBRE
AU NIVEAU 1 DE CARRE
```

Il faut donc non seulement que la procédure effectue le calcul, mais en plus qu'elle le transmette, (qu'elle le ''sorte'', comme le réclamaient les messages précédents) pour une utilisation ultérieure, affichage ou suite du calcul. Le primitif qui réalise cela est SORS. Modifions donc encore une fois notre CARRE :

```
POUR CARRE :NOMBRE
SORS :NOMBRE * :NOMBRE
FIN
```

Essayons à nouveau

```
CARRE 5
```

```
QUE DOIS-JE FAIRE AVEC 25
```

Remarquez que cette fois-ci, le message est légèrement différent et ne mentionne plus de problème avec la ligne de texte dans la procédure.

```
AFFICHE CARRE 5
25
AFFICHE CARRE 30
900
```

On progresse. Et le cube, alors ?

```
AFFICHE CUBE 5
QUE FAUT-IL FAIRE AVEC 125, LIGNE
  (CARRE :NOMBRE ) * :NOMBRE
AU NIVEAU 1 DE CUBE
```

Cette fois-ci, la procédure semble bien avoir effectué le calcul que nous attendions, et le seul problème à régler est probablement encore de mettre la commande SORS :

```
POUR CUBE :NOMBRE
SORS ( CARRE :NOMBRE ) * :NOMBRE
FIN
```

Voyons un peu :

```
AFFICHE CUBE 5
125
AFFICHE CUBE 10
1000
```

Nous aurons l'occasion de rencontrer à nouveau

des procédures où il est indispensable de *sortir* le résultat, pour pouvoir l'afficher ou le réutiliser.

• *Pour aller plus loin*

1. Construire une procédure OPPOSE dépendant d'un paramètre, qui fournit l'opposé du nombre donné.

2. Construire une procédure INVERSE dépendant d'un paramètre, qui fournit l'inverse du nombre donné.

3. Utiliser le signe = avec divers nombres et diverses opérations (un décimal entier est-il égal à l'entier correspondant ? cas des deux divisions ...).

4. Construire une procédure qui calcule le carré de l'hypoténuse, connaissant chacun des deux côtés de l'angle droit.

5. Construire une procédure qui fournit le reste décimal quand on divise un décimal par un entier ou par un autre décimal, le quotient étant arrêté ''à la virgule''.

6. Comment déterminer si la partie décimale d'un nombre est inférieure à Ø,5 ?

7. Comment déterminer si un nombre donné est un nombre entier (un décimal entier comme 3. doit être considéré comme un entier).

CHAPITRE V

MANIPULER DES MOTS ET DES LISTES

LOGO est plus riche que la plupart des langages de programmation, en ce qui concerne le traitement des chaînes de caractères. En effet, il peut travailler sur deux types d'objets : les *mots* et les *listes.*

• *Les mots*

Jusqu'à présent nous avons utilisé un certain nombre de *termes* que LOGO comprend :

- des *primitifs* : AVANCE, DESSINE, AFFICHE, LEVEPLUME, QUOTIENT...
- des *procédures* : CARREAU, FENETRE, CUBE...

Mais si vous cherchez à inscrire un texte sur l'écran, vous allez vous heurter à un refus systématique :

```
BONJOUR
BONJOUR N'A PAS ETE DEFINI
AFFICHE BONJOUR
BONJOUR N'A PAS ETE DEFINI
IMPRIME BONJOUR
IL N'Y A PAS DE PROCEDURE BONJOUR
```

et essayer de définir la procédure BONJOUR par

```
POUR BONJOUR
AFFICHE BONJOUR
FIN
```

n'améliore pas la situation :

```
BONJOUR
PLUS D'ESPACE DISPONIBLE !, LIGNE
    AFFICHE BONJOUR
AU NIVEAU 88 DE BONJOUR
```

(des variantes de cette procédure avec BONJOUR seul comme texte, ou IMPRIME BONJOUR, ou SORS BONJOUR, amènent des réactions variables –il vous faudra peut-être même utiliser CTRL G pour reprendre la situation en main ! –, mais ne répondent toujours pas à notre question).

Il nous faut trouver un moyen d'indiquer à LOGO que BONJOUR doit être considéré comme l'objet à afficher, c'est-à-dire comme un mot. Nous avons déjà rencontré une situation semblable quand nous avons utilisé des paramètres : il fallait distinguer un paramètre d'un nom de procédure, et c'était le signe **:** qui était utilisé. Ici, c'est le signe de *guillemets* (que l'on obtient par SHIFT **2** sur Apple II) qui fournit cette indication. Les guillemets doivent immédiatement précéder le mot (pas d'espace entre les guillemets et le mot) :

mot

```
AFFICHE "BONJOUR
BONJOUR
```

Modifions maintenant la procédure BONJOUR, de la manière suivante :

```
POUR BONJOUR
SORS "BØNJØUR
FIN
```

(c'est volontairement que nous écrivons des zéros au lieu de lettres O).

```
AFFICHE "BONJOUR
BONJOUR
AFFICHE BONJOUR
BØNJØUR
```

Ces deux derniers essais sont très instructifs :

- un mot peut être considéré comme *mot* même s'il désigne également le titre d'une procédure

- AFFICHE "BONJOUR inscrit sur l'écran le mot indiqué par les guillemets. AFFICHE BONJOUR inscrit sur l'écran ce que transmet la procédure BONJOUR.

On aura aussi des résultats différents avec le primitif IMPRIME :

```
IMPRIME BONJOUR
POUR BONJOUR
SORS "BØNJØUR
FIN
```

alors que

```
IMPRIME "BONJOUR
IMPRIME N'AIME PAS RECEVOIR "BONJOUR
```

Le primitif IMPRIME doit impérativement être suivi d'un nom de procédure.

Un *mot* peut être un mot au sens usuel du terme (MARMITE, SACHA, DIPLODOCUS sont des mots), mais aussi des assemblages plus inhabituels de caractères : R2D2 , H2SO4, B?+(4 seront considérés par LOGO comme des mots pourvu que vous les fassiez précéder du caractère *guillemets.* Attention : ne succombez pas à la tentation de *fermer les guillemets,* car LOGO considèrerait ce caractère comme faisant partie du mot !

```
AFFICHE "BONJOUR"
BONJOUR"
```

Un mot ne peut pas comporter d'espaces : en effet, le premier espace rencontré indique la fin du mot.

CHAPITRE V

MANIPULER DES MOTS ET DES LISTES

Les nombres sont des mots. En toute rigueur, on devrait écrire

```
AFFICHE "18
```

ou

```
AFFICHE "25 * "32
```

Mais, si LOGO ne voit aucun inconvénient à cette écriture de puriste, il peut également se débrouiller sans les guillemets au début des nombres. En effet, il n'y a aucun risque de confusion, car un nombre ne peut pas être utilisé comme nom de procédure :

```
POUR 123
POUR N'AIME PAS RECEVOIR 123
```

Si on fait précéder un primitif du caractère guillemets, *ce nom est considéré comme un mot, et la commande correspondante n'est pas exécutée :*

```
AFFICHE "QUOTIENT 54 9
QUOTIENT
QUE DOIS-JE FAIRE AVEC 54
```

mot vide Il existe un *mot vide* : il se note [''] , et son affichage produit une ligne vide :

```
AFFICHE " AFFICHE "BONJOUR

BONJOUR
```

- ***Que peut-on faire avec des mots ?***

Nous venons de voir que l'on peut afficher un mot. Si l'on veut afficher plusieurs mots, il faudra le repréciser pour chacun, ou bien en faire une liste ; sinon, on retrouvera la question bien connue

```
AFFICHE "IL "FAIT "JOUR
IL
QUE DOIS-JE FAIRE AVEC "FAIT
```

L'absence de guillemets devant le terme FAIT aurait provoqué le message :

```
FAIT N'A PAS ETE DEFINI
```

Mais

```
AFFICHE "IL AFFICHE "FAIT AFFICHE "JOUR
IL
FAIT
JOUR
```

ne réalise pas tout à fait la mise en page souhaitable. Nous y reviendrons...

Quatre primitifs permettent d'extraire des parties d'un mot :

PREMIER extrait la première lettre de ce mot
DERNIER extrait la dernière lettre de ce mot
SAUFPREMIER fabrique un mot comportant tous les caractères du mot donné, sauf le premier
SAUFDERNIER fabrique un mot comportant tous les caractères du mot donné, à l'exception du dernier.

```
PREMIER
DERNIER
SAUFPREMIER
SAUFDERNIER
```

Exemples

```
AFFICHE PREMIER "JOUR
J
AFFICHE DERNIER "NUIT
T
AFFICHE SAUFPREMIER "TAXE
AXE
AFFICHE SAUFDERNIER "PIN
PI
```

Bien sûr, on peut combiner ces primitifs entre eux :

```
AFFICHE SAUFPREMIER SAUFDERNIER "LAPIN
```

```
API
```

et aller chercher la troisième lettre d'un mot

Les trois points signifient : à taper sur une seule ligne.

```
AFFICHE PREMIER SAUFPREMIER ...
... SAUFPREMIER "TAXI
```

```
X
```

Si vous essayez d'extraire quelque chose d'un mot vide, vous recevrez un message d'erreur : attention à l'ordre des primitifs

```
AFFICHE PREMIER SAUFPREMIER ...
... PREMIER "TAXI
PREMIER N'AIME PAS RECEVOIR
```

De même qu'on a pu extraire des parties d'un mot, on peut fabriquer un mot à partir de deux morceaux : le primitif est MOT

```
MOT AFFICHE MOT "BOULE "VERSER
BOULEVERSER
```

Puisque les nombres sont des mots en LOGO, on pourra effectuer sur les nombres les mêmes manipulations :

```
AFFICHE SAUFDERNIER 1732
173
AFFICHE (MOT 18 24 ) + MOT ( 12 16)
3040
```

Attention :

```
AFFICHE MOT 18 24 + MOT 12 16
181240
```

infixé **préfixé**

En effet, + étant une opération infixée (écrite entre les nombres sur lesquels elle porte) est effectuée avant les opérations préfixées (celles qui sont écrites avant les nombres sur lesquels elles portent) : le premier calcul est donc MOT 12 16, puis 24 + MOT 12 16, et enfin MOT 18 (24 + MOT 12 16).

• *Les listes*

liste

Une liste est une suite de mots : c'est un peu comme une phrase en Français. En Français, on reconnait une phrase parce qu'elle commence par une majuscule et qu'elle se termine par un point. En LOGO, on reconnait une liste parce qu'elle commence par un crochet ouvrant **[** (que l'on obtient avec SHIFT **N** sur Apple II) et qu'elle se termine par un crochet fermant **]** (que l'on obtient avec SHIFT **M** sur Apple II).

Nous avons déjà utilisé des listes à deux reprises sans le dire : lors de la prise de contact, quand nous avons écrit :

```
AFFICHE [LOGO EST UN LANGAGE]
```

et quand nous avons utilisé le primitif REPETE.

Remarquons que, de même que le caractère *guillemets* ne s'inscrit pas lors de l'affichage d'un mot, les caractères **[** et **]** (qu'il ne faut surtout pas confondre avec les parenthèses) ne sont pas non plus reproduits à l'écran lors de l'affichage d'une liste. De même, si vous mettez plusieurs espaces pour séparer les mots, un seul sera pris en compte :

```
AFFICHE [PLUSIEURS      ESPACES]
PLUSIEURS ESPACES
```

Enfin, à l'intérieur d'une liste les mots *ne doivent pas* être précédés de guillemets, sauf si vous en avez expressément besoin :

```
AFFICHE [LE NOMBRE "PI"]
LE NOMBRE "PI"
```

Bien qu'à l'affichage on ne puisse pas distinguer un mot d'une liste comportant ce seul mot,

```
AFFICHE "UN
UN
AFFICHE [ UN ]
UN
```

il ne s'agit pas du même objet ; demandez plutôt à LOGO son opinion sur ce sujet :

```
AFFICHE "UN = [ UN ]
FAUX
```

liste vide La liste vide existe : on peut la noter **[ ]** , et son affichage produit une ligne vide :

```
AFFICHE [ ] AFFICHE [ SUITE ]

SUITE
```

- ***Que peut-on faire avec des listes ?***

Les primitifs que nous avons vus pour manipuler les mots sont aussi utilisables avec des listes.

PREMIER PREMIER extrait le premier *mot* de la liste donnée

DERNIER DERNIER extrait le dernier *mot* de la liste donnée

SAUFPREMIER SAUFPREMIER fabrique une *liste* comportant tous les mots de la liste donnée à l'exception du premier

SAUFDERNIER SAUFDERNIER fabrique une *liste* comportant tous les mots de la liste donnée à l'exception du dernier.

Voici quelques exemples :

```
AFFICHE PREMIER [ UN DEUX TROIS ]
UN
AFFICHE DERNIER [ NOUS IRONS AU BOIS ]
BOIS
```

```
AFFICHE SAUFDERNIER [ QUATRE CINQ SIX ]
QUATRE CINQ
AFFICHE SAUFPREMIER ...
                ... [ CUEILLIR DES CERISES ]
DES CERISES
```

Là encore on peut combiner les primitifs, mais il ne faut pas perdre de vue que le PREMIER et le DERNIER d'une *liste* sont des *mots* :

```
AFFICHE DERNIER SAUFDERNIER PREMIER ...
                ... [ DERRIERE CHEZ MOI ]
R
```

Décomposons les opérations :

PREMIER [DERRIERE CHEZ MOI]	donne DERRIERE
SAUFDERNIER "DERRIERE	donne DERRIER
DERNIER "DERRIER	donne R

Comme avec les mots, LOGO signale par un message si on essaie d'extraire quelque chose du vide.

Nous avons vu que, pour LOGO, un mot et une liste ne comportant que ce seul mot ne sont pas la même chose ; de la même manière, le mot vide et la liste vide sont deux objets distincts :

```
AFFICHE "  = [ ]
FAUX
```

L'analogue de MOT est PHRASE, qui s'abrège en PH:

```
PHRASE
AFFICHE PHRASE ...
... [ VOICI COMMENT FONCTIONNE ] ...
                ... [ LE PRIMITIF PHRASE ]
VOICI COMMENT FONCTIONNE ...
                ... LE PRIMITIF PHRASE
```

Ce primitif a besoin de deux données, qui peuvent être soit des mots, soit des listes, et il produit une liste :

```
AFFICHE PHRASE ...
        ... "AUJOURD'HUI [IL FAIT BEAU]
AUJOURD'HUI IL FAIT BEAU
AFFICHE PHRASE "MERCI "INFINIMENT
MERCI INFINIMENT
```

- ***Affecter un nom à quelque chose***

paramètre

Revenons sur une idée du chapitre 3 : utiliser un mot pour désigner quelque chose que l'on précisera plus tard.

Le paramètre peut être un nombre, comme dans

```
POUR CARREAU :COTE1 :COTE2
AVANCE :COTE1 DROITE 90
AVANCE :COTE2 DROITE 90
AVANCE :COTE1 DROITE 90 AVANCE :COTE2
FIN
```

ou dans

```
POUR CUBE :NOMBRE
SORS :NOMBRE * :NOMBRE * :NOMBRE
FIN
```

ou un mot :

```
POUR MANIPULE :MOT
AFFICHE PHRASE ...
        ... [LE PREMIER CARACTERE DE:] :MOT
```

```
AFFICHE PHRASE "EST PREMIER :MOT
AFFICHE PHRASE ...
    ... [ LE DERNIER CARACTERE DE: ] :MOT
AFFICHE PHRASE "EST DERNIER :MOT
FIN
```

ou une liste :

```
POUR TOURNE :LISTE
AFFICHE PHRASE ...
    ... SAUFPREMIER :LISTE PREMIER :LISTE
FIN
```

Essayez :

```
AFFICHE CUBE 30
27000
```

puis

```
MANIPULE "ALPHABET
LE PREMIER CARACTERE DE: ALPHABET
EST A
LE DERNIER CARACTERE DE: ALPHABET
EST T
```

et enfin

```
TOURNE [ BELLE MARQUISE, VOS BEAUX ...
    ... YEUX ME FONT MOURIR D'AMOUR ]
MARQUISE, VOS BEAUX YEUX ...
    ... ME FONT MOURIR D'AMOUR BELLE
```

Pourtant, si maintenant vous demandez

```
AFFICHE :NOMBRE
```

```
:NOMBRE N'A PAS ETE CREE
```

Même chose pour les deux autres paramètres.

paramètre local

Quand on utilise une procédure paramétrée, le paramètre n'a de valeur qu'à l'intérieur de cette procédure : on dit qu'il est *local* à cette procédure.

CREE

On peut également affecter un nom à un objet en dehors de toute procédure. C'est particulièrement utile quand l'objet doit être utilisé plusieurs fois, surtout s'il est un peu long à écrire. Le primitif est CREE : il est suivi du *nom* que l'on veut attribuer, puis de *l'objet* auquel on l'attribue.

```
CREE "NUMERO 20
CREE "TERME "CLAFOUTIS
CREE "PUB [ MANGEZ DES POMMES, ...
        ... VOUS AUREZ DE BELLES DENTS ]
```

Il faut maintenant faire attention à distinguer l'étiquette (''NUMERO, ''TERME, ''PUB) , de l'objet qui est dans la boite ainsi étiquetée.

```
AFFICHE "NUMERO
NUMERO
AFFICHE :NUMERO
20
AFFICHE "TERME
TERME
AFFICHE :TERME
CLAFOUTIS
AFFICHE "PUB
PUB
```

```
AFFICHE :PUB
MANGEZ DES POMMES,...
...VOUS AUREZ DE BELLES DENTS
```

C'est le caractère **:** (qui doit *immédiatement précéder* le nom, surtout pas d'espace) qui indique à LOGO d'utiliser le *contenu* de la boite et non l'*étiquette.*

Il existe également un primitif qui fait la même chose que le caractère **:** *C'est le primitif CHOSE. Il est indispensable quand plusieurs étiquettes sont attribuées en chaine :*

```
CHOSE
```

```
CREE "CLASSIFICATION "TERME
AFFICHE :CLASSIFICATION
TERME
AFFICHE CHOSE "CLASSIFICATION
TERME
AFFICHE CHOSE CHOSE "CLASSIFICATION
CLAFOUTIS
```

L'objet qui est dans une ''boite'' peut être utilisé comme paramètre dans une procédure :

```
AFFICHE CUBE :NUMERO
8000
```

De la même manière

```
MANIPULE :TERME
LE PREMIER CARACTERE DE: CLAFOUTIS
EST C
LE DERNIER CARACTERE DE: CLAFOUTIS
EST S
```

Si la procédure ne précise pas la nature de l'objet

utilisé comme paramètre, on peut utiliser un autre type d'objet :

```
MANIPULE :NUMERO
LE PREMIER CARACTERE DE: 2Ø
EST 2
LE DERNIER CARACTERE DE: 2Ø
EST Ø
```

quitte à obtenir parfois des résultats un peu bizarres :

```
MANIPULE :PUB
LE PREMIER CARACTERE DE: MANGEZ DES ...
... POMMES, VOUS AUREZ DE BELLES DENTS
EST MANGEZ
LE DERNIER CARACTERE DE: MANGEZ DES ...
... POMMES, VOUS AUREZ DE BELLES DENTS
EST DENTS
```

- ***Pour aller plus loin***

1. Y a-t-il des mots tels que le PREMIER du mot soit égal au mot ?
2. Y a-t-il des listes telles que le PREMIER de la liste soit égal à la liste ?
3. Y a-t-il des mots pour lesquels le PREMIER du mot soit égal au SAUFPREMIER ?
4. Y a-t-il des mots pour lesquels le PREMIER du mot soit égal au SAUFDERNIER ?
5. Ecrire une procédure de conjugaisons (présent de l'indicatif par exemple).
6. Ecrire une procédure de table de multiplication.
7. Ecrire une procédure de redoublement : par exemple qui, pour le paramètre ZO sort ZOZO.

CHAPITRE VI
PROCÉDURES PLUS SOPHISTIQUÉES

Ce chapitre est principalement consacré à la récursion, outil très puissant que ne possèdent pas des langages comme FORTRAN et la plupart des versions de BASIC. C'est un caractère naturel pour un langage procédural. En effet, résoudre un problème à l'aide de procédures conduit d'une manière évidente à un fractionnement de ce problème en sous-problèmes plus élémentaires, qui seront pris en charge par des procédures spécifiques. De la même manière qu'une procédure en appelle une autre pour réaliser la suite de la tâche, il est indispensable qu'elle puisse éventuellement s'appeler elle-même. Les premiers exemples peuvent laisser penser que la récursion n'est qu'une variante de l'itération et ne diffère guère d'une boucle. L'analyse de la circulation de l'information et les exemples de la fin du chapitre essayent de montrer la puissance de cette technique, ainsi que la brièveté et l'élégance de procédures utilisant l'appel récursif.

Attention : dans ce chapitre, nous remplaçons le primitif DESSINE par la procédure DESSIN :

```
POUR DESSIN
DESSINE
PAGE
FIN
```

Le primitif PAGE a été expliqué dans le chapitre 2.

• *Tourner en rond*

Retrouvons notre amie la tortue, et essayons de lui faire tracer un rond. La définition scolaire du cercle, comme ensemble des points situés à une distance donnée --le rayon-- d'un point donné --le centre-- ne nous est visiblement d'aucune utilité. En effet, si nous essayons de nous imaginer à la place de la tortue, il va de soi que ce point lointain que les géomètres appellent centre ne nous sert pas à grand chose, et que seul a de l'importance pour la tortue ce qui se passe ''sous ses pattes''.

Se prendre pour la tortue, quand on ne sait pas quels ordres lui donner, est une idée puissante (pour reprendre la terminologie de Seymour PAPERT, qui fait référence au schéma corporel).

Si je suis la tortue et que je dois parcourir un cercle, je vais presque mettre en pratique la chanson bien connue "mettre un pied devant l'autre, et recommencer". En effet, je vais avancer un petit peu, tourner un petit peu, et refaire pareil.
Allons-y :

```
POUR ROND :UN.PETIT.PEU
AVANCE :UN.PETIT.PEU DROITE :UN.PETIT.PEU
```

et maintenant, qu'est-ce que cela veut dire : "refaire pareil" ? Ce que je viens de faire s'appelle ROND, donc je dois refaire ROND. Autrement dit, nous arrivons à la procédure

```
POUR ROND :UN.PETIT.PEU
AVANCE :UN.PETIT.PEU DROITE UN.PETIT.PEU
ROND :UN.PETIT.PEU
FIN
```

Voyons un peu :

```
DESSIN ROND 1
```

Ça y est, la tortue trace un cercle, et même le

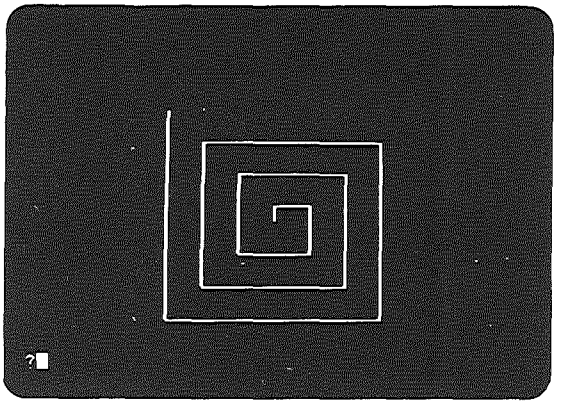
parcourt indéfiniment. Pour l'arrêter, il faudra taper CTRL **G.** On obtient alors le message :

```
ARRET ! LIGNE
AVANCE :UN.PETIT.PEU DROITE :UN.PETIT.PEU
AU NIVEAU 584 DE ROND
```

(Vous aurez probablement un autre nombre que 584).

- ***Dessiner une spirale carrée***

Changeons un peu de forme, et essayons de dessiner une spirale carrée.



On va commencer par avancer d'une certaine distance, puis tourner à droite de 9Ø degrés. Il faut alors avancer d'une distance un peu plus grande, et tourner à droite de 9Ø degrés, autrement dit, refaire exactement ce qu'on venait de faire, en augmentant, légèrement la distance, ... et ainsi de suite. Ceci décrit complètement la procédure :

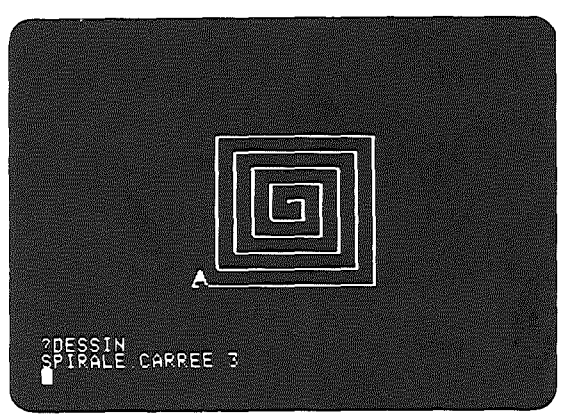
```
POUR SPIRALE.CARREE :DISTANCE
AVANCE :DISTANCE DROITE 90
SPIRALE.CARREE :DISTANCE + 5
FIN
```

Voyons un peu ce que cela donne :

```
DESSIN SPIRALE.CARREE 3
```

Ça commence très bien, mais ça s'arrête sur le message :

```
TORTUE HORS ECRAN, LIGNE
    AVANCE :DISTANCE DROITE 90
AU NIVEAU 48 DE SPIRALE.CARREE
```



En effet, la tortue arrive alors sur le bord de l'écran, et avertit qu'elle n'a plus assez de place pour continuer son trajet.

• ***Essayer de ne pas se cogner contre les murs***

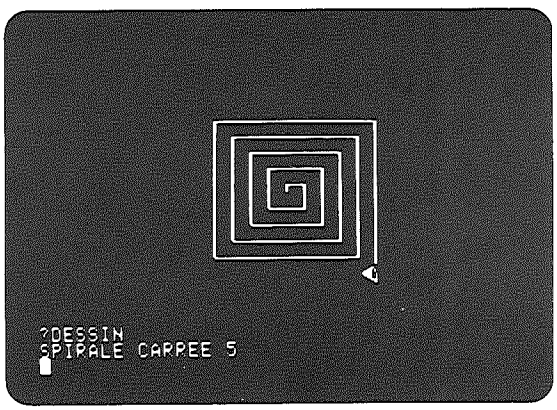
On doit bien pouvoir éviter ce message, en prévenant notre tortue qu'elle doit s'arrêter quand le côté de la spirale devient un peu grand. Mettons qu'on lui demande de s'arrêter quand le côté devient 100. Modifions notre procédure (à l'aide de EDITE, comme d'habitude), de la manière suivante :

```
POUR SPIRALE.CARREE :DISTANCE
SI :DISTANCE = 100 ALORS STOP
AVANCE :DISTANCE DROITE 90
SPIRALE.CARREE :DISTANCE + 5
FIN
```

SI
ALORS
STOP

Essayons :

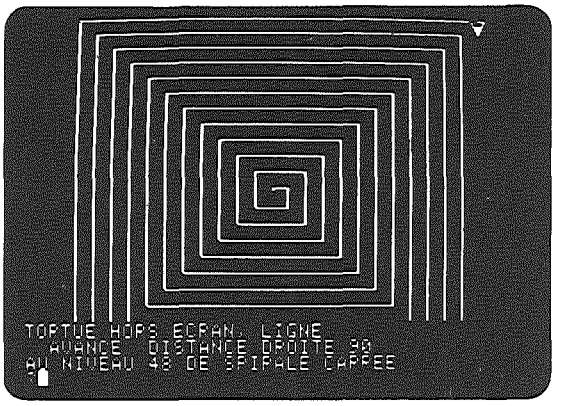
```
DESSIN
SPIRALE.CARREE 5
```



C'est parfait, on a bien une spirale carrée, et la tortue s'arrête toute seule.

Autre essai :

```
DESSIN
SPIRALE.CARREE 3
```



Mais là, ça ne va plus du tout, car elle continue, comme tout-à-l'heure, et nous envoie à nouveau le message

```
TORTUE HORS ECRAN, LIGNE
    AVANCE :DISTANCE DROITE 90
AU NIVEAU 48 DE SPIRALE.CARREE
```

Qu'a-t-il bien pu se passer ? C'est d'autant plus surprenant que l'exemple précédent fonctionnait à la perfection. Regardons de plus près ce que fait la tortue. On exécute SPIRALE.CARREE 3 :

DISTANCE est 3, donc la tortue avance de 3 pas, et tourne de 9Ø degrés, puis on passe à SPIRALE. CARREE 8 (à 3 on a ajouté 5) :

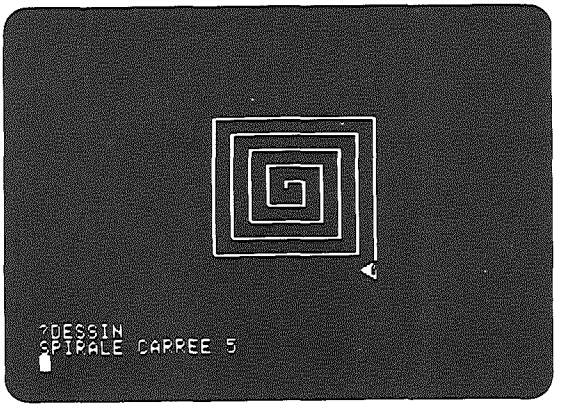
DISTANCE est 8, donc la tortue avance de 8 pas, et tourne de 9Ø degrés. Au passage suivant, DISTANCE est 13 ... La tortue va donc avancer

successivement de 13, 18, 23, 28 ... pas, en tournant à chaque fois de 9Ø degrés. On arrivera un peu plus tard à 43, 48, 53, 58 pas, toujours en tournant de 9Ø degrés à chaque coup. Plus tard encore, on a 83, 88, 93, 98, 1Ø3, 1Ø8... pas, voilà le *hic,* on ne retombe pas sur la valeur 1ØØ, et c'est pour cela qu'elle ne s'arrête pas toute seule, mais seulement quand elle arrive au bord de l'écran.

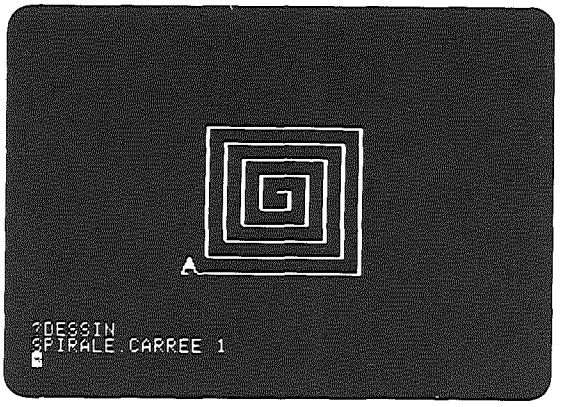
Modifions donc à nouveau la procédure :

```
POUR SPIRALE.CARREE :DISTANCE
SI :DISTANCE > 1ØØ ALORS STOP
AVANCE :DISTANCE DROITE 9Ø
SPIRALE.CARREE :DISTANCE + 5
FIN
```

Maintenant, ça fonctionne comme nous le désirions quelle que soit la valeur de DISTANCE :

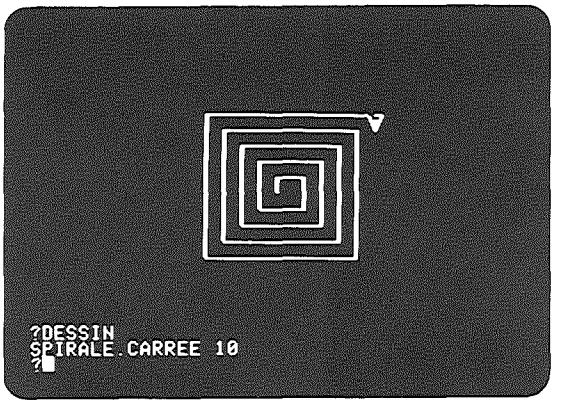



```
DESSIN
SPIRALE.CARREE 5
```



DESSIN

SPIRALE.CARREE 1



DESSIN

SPIRALE.CARREE 10

• *Un compte à rebours*

Il n'y a pas qu'avec la tortue qu'une procédure puisse s'appeler elle-même (c'est-à-dire utiliser la *récursion*). Voyons avec des nombres : qu'est-ce que compter à rebours ? Partir d'un nombre entier, mettons 1Ø, puis écrire l'entier immédiatement inférieur, et ainsi de suite ... Quand vous décrivez un projet en utilisant ''et ainsi de suite'', cela doit vous faire penser à la récursion.

récursion

```
POUR REBOURS :NOMBRE
AFFICHE :NOMBRE
REBOURS :NOMBRE - 1
FIN
```

Essayons :

```
REBOURS 1Ø
1Ø
9
8
7
6
5
4
3
2
1
Ø
-1
-2
```

Au secours ! (CTRL **G**)
Nous avons encore oublié le test d'arrêt : nous ne voulons pas des nombres inférieeurs à zéro, mais si nous ne le précisons pas à l'ordinateur, comment peut-il le deviner ?

```
POUR REBOURS :NOMBRE
SI :NOMBRE < Ø ALORS STOP
AFFICHE :NOMBRE
REBOURS :NOMBRE – 1
FIN
```

Ça devrait aller maintenant :

```
REBOURS 1Ø
```

```
1Ø
9
8
7
6
5
4
3
2
1
Ø
```

Essayez d'autres valeurs pour NOMBRE.

• *Récursion et mots*

Les procédures que nous avons vues dans ce chapitre utilisent toutes des paramètres. Cela signifie-t-il que la récursion ne peut se faire que sur des procédures paramétrées ? Non, bien sûr :

```
POUR ECHO
AFFICHE "COUCOU
ECHO
FIN
```

Cette procédure fonctionne très bien :

```
ECHO
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
COUCOU
```

tellement bien, même, qu'il faut taper CTRL **G** pour l'arrêter.

Si on n'a pas de paramètre, il n'y aura pas de moyen élémentaire d'arrêter la procédure.

Cela ne veut pas dire que la présence d'un paramètre dans la procédure permette toujours un test d'arrêt. Modifions la procédure précédente :

```
POUR ECHO :CRI
AFFICHE :CRI
ECHO :CRI
FIN
```

permet d'avoir des variations sur l'écho :

```
ECHO "EHO
EHO
EHO
EHO
EHO
EHO
EHO
EHO
EHO
EHO
EHO
```

sans qu'il y ait pour autant un arrêt automatique : là encore on aura besoin de recourir à CTRL **G**.

Vous avez déjà essayé de réaliser une procédure qui double les mots, par exemple :

```
POUR DOUBLE :DONNEE
AFFICHE MOT :DONNEE :DONNEE
FIN
```

qui fonctionne ainsi :

```
DOUBLE "ZO
ZOZO
```

```
DOUBLE "MINOU
MINOUMINOU
DOUBLE "BLA
BLABLA
```

mais on peut écrire une procédure qui répète sans fin :

```
POUR REPETE.SANS.FIN :DONNEE
AFFICHE :DONNEE
REPETE.SANS.FIN MOT :DONNEE :DONNEE
FIN
```

Voyons un peu :

```
REPETE.SANS.FIN "HA
HA
HAHA
HAHAHAHA
HAHAHAHAHAHAHAHA
HAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHA
```

Là encore, il faut taper CTRL **G** pour arrêter !

- ***Récursion et listes***

Il est de toute évidence très facile de transposer pour les listes ce que nous venons de faire avec les mots : voici une bonne collection d'exercices que vous allez retrouver explicitement dans les suggestions pour aller plus loin, en fin de chapitre. Ensemble, nous allons explorer quelques nouveaux primitifs, ce qui vous donnera d'autres idées de choses à faire avec les nombres ou les mots.

Fabriquons ensemble une procédure qui compte les éléments d'une liste. Quels sont les cas où l'on sait tout de suite répondre ? Bien sûr, quand la liste est vide : la réponse est alors Ø . Pour une liste qui comporte un seul élément, ce n'est pas beaucoup plus difficile : non, non, ce n'est pas une lapalissade mais un véritable raisonnement ! Qu'est-ce, au fond, qu'une liste à un élément ? Une liste dont le SAUFPREMIER est vide. Et la liste vide, nous venons de le voir, est un problème résolu. Et pour une liste avec un plus grand nombre d'éléments, on va s'y prendre de la même manière : extraire le SAUFPREMIER et rajouter 1 pour le résultat obtenu pour ce SAUFPREMIER.

La description en phrases n'est pas très claire, mais si on écrit la procédure et qu'on l'essaie sur quelques exemples, on comprend beaucoup mieux.

```
POUR COMPTE.LISTE :DONNEE
TESTE   TESTE :DONNEE = []
SIVRAI  SIVRAI SORS Ø STOP
SIFAUX  SIFAUX SORS 1 + COMPTE.LISTE ...
                ... SAUFPREMIER :DONNEE
FIN
```

Au lieu de TESTE ..., SIVRAI ..., SIFAUX ..., nous aurions pu utiliser SI ... ALORS...SINON...

Regardons de près ce qui se passe pour COMPTE.LISTE [LE PETIT CHAPERON ROUGE]
COMPTE.LISTE commence
DONNEE est [LE PETIT CHAPERON ROUGE]
Le test a pour résultat FAUX
La ligne SIVRAI est donc ignorée
Il faut alors calculer
1 + COMPTE.LISTE SAUFPREMIER
[LE PETIT CHAPERON ROUGE]

A nouveau COMPTE.LISTE commence
DONNEE est maintenant
[PETIT CHAPERON ROUGE]
Le test a pour résultat FAUX
La ligne SIVRAI est donc ignorée
Il faut alors calculer
1 + COMPTE.LISTE SAUFPREMIER
[PETIT CHAPERON ROUGE]

Une nouvelle fois COMPTE.LISTE commence
DONNEE est [CHAPERON ROUGE]
Le test a pour résultat FAUX
La ligne SIVRAI est donc ignorée
Il faut alors calculer
1 + COMPTE.LISTE SAUFPREMIER
[CHAPERON ROUGE]

De nouveau COMPTE.LISTE commence
DONNEE est [ROUGE]
Le test a pour résultat FAUX
La ligne SIVRAI est donc ignorée
Il faut alors calculer
1 + COMPTE.LISTE SAUFPREMIER [ROUGE]

Encore une fois COMPTE.LISTE commence
DONNEE est [ ]
Le test a pour résultat VRAI
La ligne SIVRAI s'exécute
La procédure retourne Ø
La procédure retourne 1 + Ø (c'est-à-dire 1)
La procédure retourne 1 + 1 (c'est-à-dire 2)
La procédure retourne 1 + 2 (c'est-à-dire 3)
La procédure retourne 1 + 3 (c'est-à-dire 4), et si
on ne lui a pas donné d'instructions particulières,
envoie le message

- ***Une spirale qui affiche sa taille***

L'analyse de l'exemple précédent montre un peu comment fonctionne la récursion.
Regardons ensemble un nouvel exemple.

Que pensez-vous qu'il se passe avec

```
POUR SPIC :DISTANCE
SI :DISTANCE > 5Ø ALORS STOP
AFFICHE :DISTANCE
AVANCE :DISTANCE DROITE 9Ø
SPIC :DISTANCE + 1Ø
FIN
```

Bien deviné ; pour SPIC 10 on obtient une spirale carrée, analogue à celle du début de ce chapitre, et en dessous du dessin, on voit s'afficher la mesure des côtés :

```
1Ø
2Ø
3Ø
4Ø
5Ø
```

Maintenant

```
POUR SPIC? :DISTANCE
SI :DISTANCE > 5Ø ALORS STOP
AVANCE :DISTANCE DROITE 9Ø
SPIC? :DISTANCE + 1Ø
AFFICHE :DISTANCE
FIN
```

L'affichage se produit comme avec la procédure précédente ?
Il ne s'affiche rien du tout ?
Perdu ! En tapant DESSINE SPIC? 1Ø, on a bien le même tracé de spirale, mais l'affichage est maintenant

```
5Ø
4Ø
3Ø
2Ø
1Ø
```

Bizarre ?
Regardons de plus près :
SPIC? commence
DISTANCE est 1Ø
Le test donne FAUX, donc le STOP est ignoré.
La tortue avance et tourne à droite
Une nouvel appel à la procédure SPIC? se produit, mais dans l'appel en cours on n'est pas encore arrivé au mot FIN.

SPIC? commence
DISTANCE est 2Ø
Le test donne FAUX, donc le STOP est ignoré
La tortue avance et tourne à droite
Un nouvel appel à la procédure SPIC? se produit, mais dans l'appel en cours, on n'est pas encore arrivé au mot FIN.

SPIC? commence
DISTANCE est 3Ø
Le test donne FAUX, donc le STOP est ignoré
La tortue avance et tourne à droite
Un nouvel appel à la procédure SPIC? se produit, mais dans l'appel en cours, on n'est pas encore arrivé au mot FIN.

SPIC? commence
DISTANCE est 4Ø
Le test donne FAUX, donc le STOP est ignoré
La tortue avance et tourne à droite
Un nouvel appel à la procédure SPIC? se produit, mais dans l'appel en cours, on n'est pas encore arrivé au mot FIN.

SPIC? commence
DISTANCE est 5Ø
Le test donne FAUX, donc le STOP est ignoré
La tortue avance et tourne à droite
Un nouvel appel à la procédure SPIC? se produit, mais dans l'appel en cours on n'est pas encore arrivé au mot FIN.

SPIC? commence
DISTANCE est 6Ø
Le test donne VRAI, la procédure stoppe, et retourne l'exécution à l'appel précédent.

Il reste encore une ligne à exécuter : à l'écran s'affiche la valeur de DISTANCE, c'est-à-dire 5Ø
La procédure se termine (FIN) , et retourne l'exécution à l'appel précédent.

Il reste encore une ligne à exécuter : à l'écran s'affiche la valeur de DISTANCE, c'est-à-dire 4Ø.
La procédure se termine (FIN) , et retourne l'exécution à l'appel précédent.

Il reste encore une ligne à exécuter : à l'écran s'affiche la valeur de DISTANCE, c'est-à-dire 3Ø.
La procédure se termine (FIN) , et retourne l'exécution à l'appel précédent.

Il reste encore une ligne à exécuter : à l'écran s'affiche la valeur de DISTANCE, c'est-à-dire 2Ø
La procédure se termine (FIN) , et retourne l'exécution à l'appel précédent.

Il reste encore une ligne à exécuter : à l'écran s'affiche la valeur de DISTANCE, c'est-à-dire 1Ø. La procédure se termine (FIN) , et il n'y a maintenant plus aucun appel en attente.

- ***Un lampion de mots***

La règle, en fin de compte, est assez simple : quand une procédure appelle une autre procédure, la première procédure (*procédure appelante*) attend jusqu'à ce que la deuxième procédure (*procédure appelée*) soit terminée, puis continue à exécuter les lignes d'instructions qui suivent l'appel.

Ceci dit, il faut une longue manipulation des procédures récursives pour se sentir un peu plus familier avec cet outil. Mais c'est une bonne gymnastique, et surtout un moyen extrêmement efficace pour résoudre bien des problèmes de répétition.

Voyons ensemble un dernier exemple.

```
POUR LAMPION :TERME
SI :TERME = " ALORS STOP
AFFICHE :TERME
LAMPION SAUFDERNIER :TERME
AFFICHE :TERME
FIN
```

Essayons :

```
LAMPION "MISERE
MISERE
MISER
MISE
MIS
```

CHAPITRE VI

PROCÉDURES PLUS SOPHISTIQUÉES

MI
M
M
MI
MIS
MISE
MISER
MISERE

- ***Pour aller plus loin***

1. Essayer différentes valeurs pour UN.PETIT.PEU dans la procédure ROND.
2. Ecrire une procédure ROND avec deux paramètres, un pour AVANCE, un pour DROITE, et essayer différentes valeurs.
3. Ecrire une procédure SPIRALE avec deux paramètres, un pour AVANCE; un pour DROITE, et essayer différentes valeurs.
4. Ecrire une procédure SPIRALE en faisant varier l'angle au lieu du côté dans l'appel récursif. Essayer différentes valeurs.
5. Ecrire une procédure SPIRALE où la quantité dont on augmente le côté est un paramètre ; essayer différentes valeurs.
6. Ecrire une procédure qui fait attendre, pendant un temps variable au gré de l'utilisateur, entre l'exécution de deux instructions successives.
7. Ecrire une procédure pour doubler une liste, ou pour la répéter sans fin.
8. Ecrire une procédure COMPTE.MOT.

CHAPITRE VII

BOITE A MUSIQUE ET TAMBOURIN

Les commandes de musique ne sont pas des primitifs de la version LOGO pour votre Apple II, mais sont des procédures disponibles dans le fichier MUSIQUE qui se trouve sur la disquette utilitaire. Pour utiliser la musique avec LOGO, il nous faut donc tout d'abord recopier ces procédures dans la mémoire centrale du micro-ordinateur.

Après avoir chargé le langage LOGO, retirez la disquette langage du lecteur, et rangez-la, puis insérez la disquette utilitaire, tapez

RAMENE "MUSIQUE RETURN

La lampe témoin rouge du lecteur-enregistreur de disquette s'allume, et au bout d'un certain temps s'éteint. Une longue liste de titres de procédures s'affiche alors à l'écran, puis après un nouvel accès au disque on retrouve l'habituel caractère *point d'interrogation* qui indique que LOGO attend vos instructions.

Retirez la disquette utilitaire, et rangez-la.

• *Blocs mélodiques*

Les blocs mélodiques fournissent une activité qui a été conçue et mise au point par Jeanne BAMBERGER au Massachusetts Institue of Technology.

En voici un exemple ;

Tapez

FRERE RETURN

Un message à l'écran vous propose d'essayer

quatre procédures **FA**, **FE**, **FI** et **FO**. Pensez à appuyer sur RETURN, après chacun de ces mots. Quand vous aurez écouté ces quatre *blocs mélodiques,* vous aurez probablement identifié l'air connu dont ils proviennent, et vous serez déjà en train de réarranger ces blocs pour reconstituer le chant en question.

Si vous n'êtes pas certain d'avoir trouvé la bonne réponse, tapez **FZ** et vous entendrez l'air complet, tout en voyant s'afficher ce que vous auriez dû taper.

Vous trouverez dans les suggestions pour aller plus loin, en fin de chapitre, des indications pour créer d'autres exemples de ce type.

Avant de passer au paragraphe suivant, tapez CTRL **G.**

• *Encore des blocs mélodiques*

Tapez maintenant :

CHAMP RETURN

C'est un peu la même règle du jeu qui s'applique à nouveau : vous pouvez essayer cinq procédures, **CHA**, **CHE**, **CHI**, **CHO**, **CHU**. N'oubliez pas d'appuyer sur RETURN après chacun de ces mots.

Cette fois-ci, ça semble se compliquer un peu. Le problème n'est pas vraiment qu'il y ait cinq blocs, au lieu de quatre précédemment, mais plutôt que, probablement, vous n'avez aucune idée de l'air dont ces blocs ont bien pu être tirés. Et les astucieux qui essaieraient d'avoir de l'aide en tapant **CHZ** en seront pour leurs frais :

CHZ N'A PAS ETE DEFINI

Autrement dit, il ne s'agit pas de *trouver la bonne réponse*, mais à partir de ces blocs mélodiques de composer vous-même une chanson.

Quand vous serez satisfait de votre résultat, notez-le soigneusement, pour pouvoir en discuter avec des amis à qui vous proposerez ce jeu.

Là encore, les suggestions de fin de chapitre vous proposeront des indications pour créer d'autres exemples de ce type.

Si vous êtes vraiment curieux, tapez **O** (la lettre **O**, pas le chiffre **Ø**), et vous entendrez la mélodie originale. Vous aurez probablement quelque surprise.

Avant de passer au paragraphe suivant, tapez CTRL **G**.

- ***Hauteurs et durées***

Pour écrire de la musique avec LOGO, il faut spécifier les hauteurs des notes, par des indications, entrées en une liste, et donner également les durées de ces notes, là aussi au moyen d'une liste. Ces deux listes sont indispensables pour que l'ordre JOUE puisse être exécuté. Sinon, vous recevrez un message :

```
IL MANQUE QUELQUECHOSE APRES JOUE
```

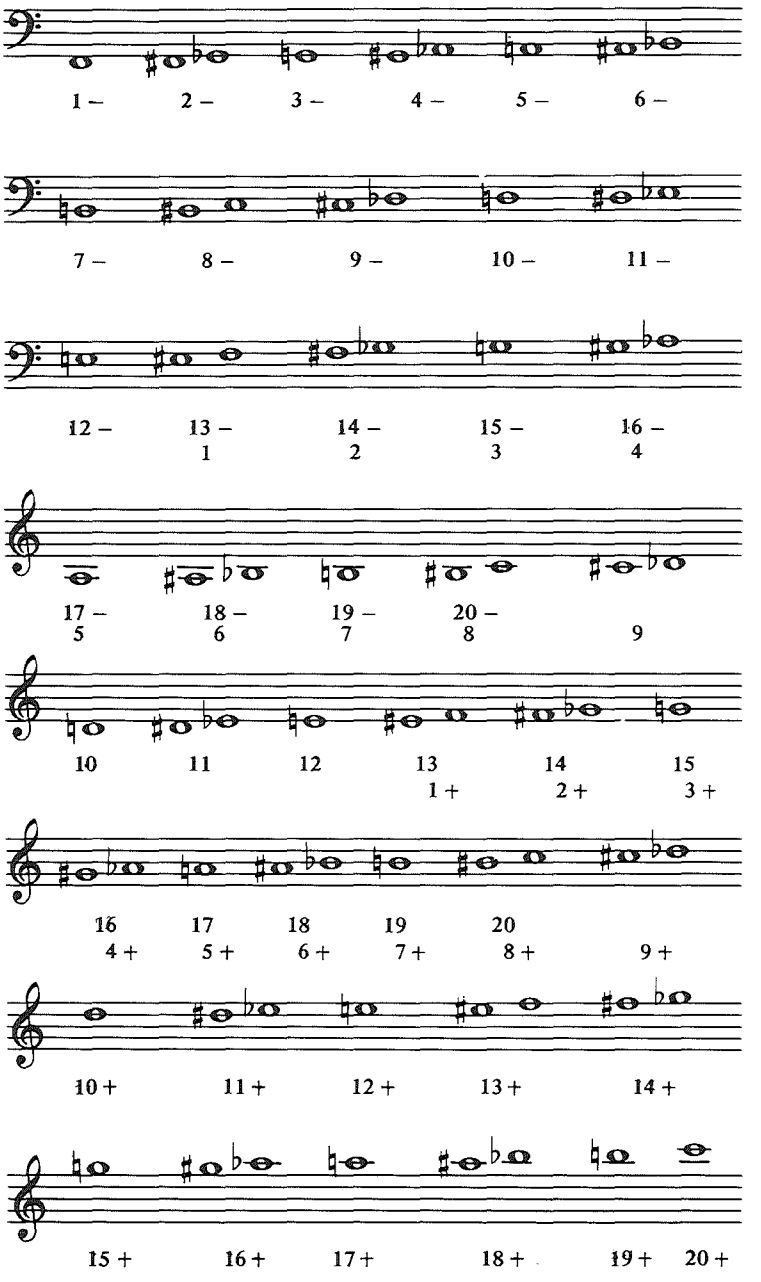
Le tableau de concordance des codes de hauteurs et des notations habituelles se trouve page suivante.

Les durées indiquent combien de temps une note est jouée. Une durée de 4Ø est deux fois plus longue qu'une durée de 2Ø, et deux fois moins longue qu'une durée de 8Ø. Essayez, pour trouver quel rythme vous voulez donner à votre morceau.

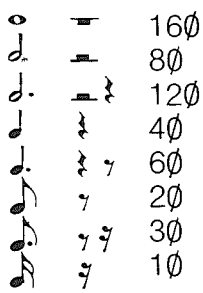
On peut avoir besoin de silence : on tapera alors **S** au lieu d'un code de note, et on donnera sa durée dans la liste des durées.

CHAPITRE VII

BOITE A MUSIQUE ET TAMBOURIN



Exemple :



Voici un exemple :

JOUE [8 8 1Ø 8 13 12] [2Ø 2Ø 4Ø 4Ø 4Ø 8Ø]
JOUE [8 8 1Ø 8 15 13 S] [2Ø 2Ø 4Ø 4Ø 4Ø 6Ø 2Ø]
JOUE [8 8 2Ø 17 13 12 1Ø S] [2Ø 2Ø 4Ø 4Ø 4Ø 4Ø 6Ø 2Ø]
JOUE [18 18 17 13 15 13] [2Ø 2Ø 4Ø 4Ø 4Ø 8Ø]

Au lieu de dialoguer avec la boite à musique, faites donc des procédures pour chacune de ces phrases musicales : vous pourrez ainsi les réutiliser plusieurs fois, sans avoir à tout retaper.

```
POUR AN1
JOUE [ 8 8 1Ø 8 13 12 ] [ 2Ø 2Ø 4Ø 4Ø 4Ø 8Ø ]
FIN
POUR AN2
JOUE ...
  ... [ 8 8 1Ø 8 15 13 S ] [ 2Ø 2Ø 4Ø 4Ø 4Ø 6Ø 2Ø ]
FIN
POUR AN3
JOUE ...
... [ 8 8 2Ø 17 13 12 1Ø S ] [ 2Ø 2Ø 4Ø 4Ø 4Ø 4Ø 6Ø 2Ø ]
FIN
```

```
POUR AN4
JOUE [ 18 18 17 13 15 13 ] [ 2Ø 2Ø 4Ø 4Ø 4Ø 8Ø ]
FIN
```

- ***Manipuler la musique***

Vous pouvez créer vos propres compositions, en utilisant JOUE et les deux listes de nombres, comme nous venons de le voir, mais également imposer quelques modifications à des airs connus.

Par exemple, transposer un morceau : c'est très simple, il suffit d'ajouter ou de retrancher un même nombre à tous les codes de hauteur.

Ou modifier le rythme en changeant les valeurs dans la liste des durées.

Ou passer du majeur au mineur :

```
POUR AN1MIN
JOUE [ 8 8 9 8 13 11 ] [ 2Ø 2Ø 4Ø 4Ø 4Ø 8Ø ]
FIN
```

Toutes les fantaisies sont permises, à vous de jouer.

Bien entendu, les primitifs que nous avons vus dans le chapitre "mots et listes" sont tous utilisables ici. Et pourquoi ne pas inventer des procédures récursives en musique ?

- ***Pour aller plus loin***

1. Imprimer ou éditer la procédure FRERE, et s'en inspirer pour une procédure MAMAN, avec les blocs :

MA : JOUE ...
... [15 15 13 13 12 12 11] [2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 4Ø]
ME : JOUE ...
... [13 13 12 12 1Ø 1Ø 8] [2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 4Ø]
MI : JOUE ...
... [8 8 15 15 17 17 15] [2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 4Ø]

2. Même exercice avec CLAIR et
CLA : JOUE [8 12 1Ø 1Ø 8] [2Ø 2Ø 2Ø 2Ø 4Ø]
CLE : JOUE [1Ø 8 7 5 3] [2Ø 2Ø 2Ø 2Ø 4Ø]
CLI : JOUE [8 8 8 1Ø 12 1Ø] [2Ø 2Ø 2Ø 2Ø 4Ø 4Ø]
CLO : JOUE [1Ø 1Ø 1Ø 1Ø 5 5] [2Ø 2Ø 2Ø 2Ø 4Ø 4Ø]

3. Même exercice avec TABAC et
TA : JOUE [1Ø 12 13 13 12 12] [1Ø 1Ø 2Ø 2Ø 2Ø 2Ø]
TE : JOUE [15 15 13 12] [2Ø 1Ø 1Ø 2Ø]
TI : JOUE [8 1Ø 12 8 1Ø] [1Ø 1Ø 1Ø 1Ø 2Ø]
TO : TJOUE [1Ø 12 13 15 8] [1Ø 1Ø 2Ø 2Ø 4Ø]
TU : JOUE [1Ø 12 13 15 1Ø] [1Ø 1Ø 2Ø 2Ø 4Ø]

4. Imprimer ou éditer la procédure CHAMP et s'en inspirer pour une procédure JEAN avec les blocs :
JA : JOUE [9+ 9+ 9+ 9+ 5+ 9+ 12+ 9+ 5+] ...
... [1Ø 1Ø 1Ø 1Ø 1Ø 1Ø 2Ø 1Ø 3Ø]
JE : JOUE [5+ 5+ 5+ 5+ 12 5+ 9+ 5+ 12] ...
... [1Ø 1Ø 1Ø 1Ø 1Ø 1Ø 2Ø 1Ø 3Ø]
JI : JOUE [12 5+ 9+ 12 5+] [1Ø 1Ø 1Ø 6Ø 6Ø]
JO : JOUE [12 12+ 12 12+ 5+] [2Ø 1Ø 2Ø 1Ø 3Ø]
JU : JOUE [12 5+ 9+ 12+ 9+] [1Ø 1Ø 1Ø 6Ø 3Ø]

5. Même exercice avec SAMA et
SA : JOUE [5+ 3+ 5+ 7+ 3+ 3+ 3+] ...
... [1Ø 1Ø 1Ø 1Ø 2Ø 2Ø 4Ø]
SE : JOUE [5+ 3+ 5+ 7+ 3+] [1Ø 1Ø 1Ø 1Ø 4Ø]
SI : JOUE [3+ 1Ø 3+ 5+ 7+ 7+ 7+] ...
... [1Ø 1Ø 1Ø 1Ø 2Ø 2Ø 4Ø]
SO : JOUE [8+ 1Ø+ 8+ 7+ 5+ 5+ 5+] ...
... [1Ø 1Ø 1Ø 1Ø 2Ø 2Ø 4Ø]

CHAPITRE VIII

OU L'ON RETROUVE LES NOMBRES

Nous nous proposons ici de reprendre quelques exemples de procédures utilisant le primitif SORS avec les nombres, et de décrire les principaux autres primitifs spécifiques de ce domaine.

• *Des procédures qui calculent pour vous*

Trouver le maximum de deux nombres, c'est comparer le premier au deuxième ; si le premier est plus grand que le second, on sort le premier, sinon on sort le deuxième.

```
POUR MAX :N1 :N2
SINON  SI :N1 > :N2 ALORS SORS ...
                ... :N1 SINON SORS :N2
FIN
```

Exemple :

```
AFFICHE MAX 12 18
18
AFFICHE MAX 3 -4
IL MANQUE QUELQUECHOSE APRES MAX
```

parenthèses

En effet, −4 n'a pas été considéré comme un deuxième nombre, car l'opération 3 − 4 a été effectuée : pour éviter cette méprise, il suffit de mettre des parenthèses autour de −4.

```
AFFICHE MAX 3 (-4)
3
AFFICHE MAX -12 8
8
```

Trouver le maximum d'une suite de nombres peut se faire de la manière suivante :

- si la liste est vide, le problème ne se pose pas
- si la liste n'a qu'un élément, c'est ce nombre qui est le maximum
- si la liste a plus d'un élément, on compare les deux premiers, on élimine le plus petit des deux, et on recommence avec la liste privée de l'élément que l'on vient d'éliminer.

Comment reconnait-on qu'une liste a un seul élément ? Par exemple, parce que son SAUFPREMIER est vide.

```
POUR MAXI :SUITE
SI :SUITE = [ ] ALORS STOP
SI SAUFPREMIER :SUITE = [ ] ...
                ... ALORS SORS PREMIER :SUITE
TESTE PREMIER :SUITE > PREMIER ...
                ... SAUFPREMIER :SUITE
SIVRAI SORS MAXI PHRASE PREMIER ...
... :SUITE SAUFPREMIER SAUFPREMIER :SUITE
SIFAUX SORS MAXI SAUFPREMIER :SUITE
FIN
```

Exemples :

```
AFFICHE MAXI [ 1 2 8 5 3 1 ]
8
AFFICHE MAXI [ -4 -9 -7 ]
-4
```

Attention : à l'intérieur d'une liste, il ne faut pas entourer les nombres négatifs de parenthèses, ni séparer le signe du nombre par un espace.

Trouver le reste dans la division euclidienne, c'est soustraire du dividende le produit du diviseur par le quotient entier.

```
POUR RESTE :A :B
```

```
SORS :A - :B * QUOTIENT :A :B
FIN
```

Exemples :

```
AFFICHE RESTE 18 5
3
AFFICHE RESTE 24 2
Ø
```

Il y a un hic, me direz vous, c'est que vous avez reçu un message d'erreur en essayant

```
POUR RESTE
RESTE EST UN PRIMITIF LOGO
```

Alors de deux choses l'une :

- ou bien vous avez essayé ce nouveau primitif, et constaté qu'il répond justement à notre question
- ou bien vous avez modifié le titre en RISTE ou RESTER ou autre chose encore.

• *Tirages au sort*

LOGO possède une fonction aléatoire : c'est le primitif HASARD.

HASARD doit être suivi d'un entier naturel *n*, et sort un entier au hasard entre 0 et *n* − *1*.

Exemple :

```
HASARD
REPETE 10 [ AFFICHE HASARD 50 ]
40
15
```

```
13
35
5
14
34
10
26
41
REPETE 10 [ AFFICHE HASARD 50 ]
12
29
31
1
27
43
49
46
6
7
```

A chaque fois que l'on met en marche LOGO, la liste des nombres aléatoires est la même : c'est pour cela que vous avez sur votre écran exactement la même liste que celle qui figure dans le manuel. Cela peut être un inconvénient. Aussi, on dispose d'un deuxième primitif, AUHASARD, qui a un double rôle. En tapant AUHASARD, les appels ultérieurs à HASARD 5Ø ne donneront pas la même série de nombres au hasard.

Exemple :

Eteindre et recharger LOGO

```
REPETE 10 [ AFFICHE HASARD 50 ]
40
15
13
35
5
14
34
10
26
41
REPETE 10 [ AFFICHE HASARD 50 ]
12
29
31
1
27
43
49
46
6
7
```

Eteindre et recharger LOGO

```
AUHASARD                                    AUHASARD
REPETE 1Ø [ AFFICHE HASARD 5Ø ]
28
17
4
44
26
29
17
8
6
26
REPETE 1Ø [ AFFICHE HASARD 5Ø ]
41
4
1
2Ø
4
46
49
23
44
49
```

Par contre, on peut avoir besoin de retrouver

régulièrement les mêmes séries de nombres au hasard. Pour cela, utiliser AUHASARD suivi d'un nombre :

Exemple :

```
(AUHASARD 2Ø)
REPETE 1Ø [ AFFICHE HASARD 5Ø ]
14
11
38
6
29
40
34
28
7
37
```

Le primitif doit alors être écrit entre parenthèses, sinon on obtient un message d'erreur : QUE DOIS-JE FAIRE AVEC ...

Le nombre qui suit alors AUHASARD n'a rien à voir avec le nombre qui suit ultérieurement HASARD : c'est seulement une indication de l'endroit à partir duquel la liste des nombres au hasard va être déroulée.

```
(AUHASARD 2Ø)
REPETE 1Ø [ AFFICHE HASARD 5Ø ]
14
11
```

```
38
6
29
40
34
28
7
37
```

- ***Un peu de trigonométrie***

LOGO connaît les fonctions *sinus* et *cosinus* : ce sont les primitifs SIN et COS, qui nécessitent la donnée d'un nombre mesurant en degrés un angle. Vous pouvez donc facilement construire la tangente et la cotangente.

```
POUR TANGENTE :ANGLE                                    TANGENTE
SI :ANGLE = 90 ALORS AFFICHE ...
... [ PAS DE VALEUR POUR TANGENTE 90 DEGRES ] ...
                                        ... STOP
SORS (SIN :ANGLE) / COS :ANGLE                          SIN
FIN                                                     COS
```

Exemple :

```
AFFICHE TANGENTE 45
1.
AFFICHE TANGENTE 60
1.73205
```

Il faut penser à mettre entre parenthèses SIN :ANGLE, sinon la division a pour dividende :ANGLE et pour diviseur COS :ANGLE, puis on prend le Sinus du quotient qui en résulte.

Une autre fonction bien utile est ATG. Ce primitif a besoin de deux données, *x* et *y* , et sort un angle (mesuré en degrés) dont l'arc tangente est *x/y*. Le résultat est un nombre compris entre Ø et 36Ø , le quadrant étant déterminé par les signes de *x* et y, avec une précision de l'ordre du centième.

Exemple :

```
ATG
AFFICHE ATG 2 2
45.ØØ6
AFFICHE ATG -2 (-2)
225.ØØ1
```

Voici deux procédures pour obtenir l'*arc sinus* et l'*arc cosinus*.

```
ARCSIN
POUR ARCSIN :X
SORS ATG :X (RCAR 1 - :X * :X)
FIN
```

RCAR — Le primitif RCAR permet le calcul de la racine carrée.

```
ARCCOS
POUR ARCCOS :X
SORS ATG (RCAR 1 - :X * :X) :X
FIN
```

- ***Pour aller plus loin***

1. Ecrire une procédure qui trouve le minimum de deux nombres

2. Ecrire une procédure qui trouve la valeur absolue d'un nombre.
3. Ecrire une procédure qui trouve le minimum d'une suite de nombres.
4. Ecrire une procédure qui sort la suite des multiples d'un entier donné.
5. Ecrire une procédure qui sort l'ensemble des diviseurs d'un nombre entier donné.
6. Ecrire une procédure qui simule un tirage à pile ou face.
7. Ecrire une procédure qui simule le jet d'un dé.

CHAPITRE IX

OU L'ON RETROUVE DU TEXTE

Quatre parties bien différentes dans ce chapitre : tout d'abord un complément sur les manipulations de listes, puis quelques indications pour la mise en page sur l'écran. Ensuite, une nouvelle manière de construire des procédures, sans passer par l'éditeur. Enfin des primitifs permettant le dialogue entre une procédure et un utilisateur, avec un exemple d'application pour une procédure de ''code secret''.

• *Listes de listes*

Au chapitre 5, nous avons vu comment extraire des éléments d'une liste, et comment recoller des mots ou des listes pour fabriquer de nouvelles listes. Une liste peut avoir pour éléments des listes : c'est d'ailleurs ce qui fait tout l'intérêt de la notion de liste, bien qu'au début cela paraisse un peu embrouillant.

```
AFFICHE [ [ DERRIERE CHEZ MOI ] ...
                ... IL Y A [ UN ETANG ] ]
[ DERRIERE CHEZ MOI ] IL Y A [ UN ETANG ]
```

Remarquez que les crochets extérieurs ne s'affichent pas. Pour mieux comprendre ce qui se passe, essayons des variantes.

```
CREE "A [ [ DERRIERE CHEZ MOI ] ...
                ... IL Y A [ UN ETANG ] ]
CREE "B [ DERRIERE CHEZ MOI ] ...
                ... [ IL Y A ] UN ETANG ]
AFFICHE PREMIER :A
DERRIERE CHEZ MOI
AFFICHE PREMIER :B
```

```
DERRIERE
AFFICHE DERNIER :A
UN ETANG
AFFICHE DERNIER :B
ETANG
```

Nous avons vu que l'on peut recoller un mot à une liste, ou deux listes ensemble pour former une liste plus grande.

```
CREE "C PHRASE :A "CHARMANT
AFFICHE :C
[ DERRIERE CHEZ MOI ]...
    ...IL Y A [ UN ETANG ] [ CHARMANT ]
CREE "D PHRASE :A [ DEPUIS LONGTEMPS ]
AFFICHE :D
[ DERRIERE CHEZ MOI ] IL Y A...
    ...[ UN ETANG ] DEPUIS LONGTEMPS
```

On peut aussi insérer un objet, mot ou liste dans une liste, soit au début, avec le primitif INSEREP (contraction de *INSERE EN PREMIER*), soit à la fin avec le primitif INSERED (mis pour *INSERE EN DERNIER*).

```
CREE "E INSERED :A "CHARMANT
INSERED N'AIME PAS RECEVOIR "CHARMANT
```

INSERED

On ne peut insérer que dans une liste et pas dans un mot !

```
CREE"E INSERED "CHARMANT :A
AFFICHE :E
```

CHAPITRE IX

OU L'ON RETROUVE DU TEXTE

```
[ DERRIERE CHEZ MOI ]...
... IL Y A [ UN ETANG ] CHARMANT
```

```
INSEREP  CREE "F INSEREP [ DEPUIS LONGTEMPS ] :A
         AFFICHE :F
```

```
[ DEPUIS LONGTEMPS ] [ DERRIERE CHEZ MOI ]...
... IL Y A [ UN ETANG ]
```

Regardons le nombre d'éléments de la liste obtenue :
- avec le primitif PHRASE appliqué à une liste de n *éléments, et à un mot, on obtient une liste de* n + 1 *éléments. Ainsi, :A est une liste à cinq éléments, (le premier et le dernier sont des listes, et il y a trois mots) ; :C'est une liste à six éléments.*

- avec le primitif PHRASE appliqué a une liste de n *éléments, et à une liste de* p *éléments, on obtient une liste de* n + p *éléments. Ainsi, :D est une liste à sept éléments.*

- avec le primitif INSEREP (ou INSERED) appliqué à une liste de n *éléments et à une liste de* p *éléments, on obtient une liste de* n + 1 *éléments, car la deuxième donnée est insérée sous forme d'*une *sous-liste. Ainsi, :E et :F sont des listes à six éléments.*

Le primitif LISTE permet de créer une liste à deux éléments à partir d'une liste à *n* éléments, et d'une liste à *p* éléments.

Exemple :

```
LISTE  CREE "H LISTE ...
          ... [ IL FAIT BEAU ] [ C'EST SURPRENANT ]
       AFFICHE :H
       [ IL FAIT BEAU ] [ C'EST SURPRENANT ]
```

LISTE accepte aussi les mots

```
AFFICHE LISTE "ICI [ IL FAIT BEAU ]
ICI [ IL FAIT BEAU ]
```

- ***Mise en page***

Vous avez déjà remarqué que le *tampon d'édition* est plus grand qu'une ligne d'affichage à l'écran, aussi bien en mode direct qu'en mode édition. Peut-être avez vous été intrigué, en mode édition, par l'affichage d'un point d'exclamation quand le contenu du tampon dépasse la ligne d'écran:c'est justement pour vous indiquer que vous êtes toujours sur la même ''ligne''.

Cela peut être important : par exemple, SI ... ALORS ... SINON ... doit être écrit sur une seule ''ligne''. Au contraire, TESTE ... SIVRAI ... SIFAUX ... doit être écrit sur trois lignes différentes.

Quand vous prévoyez un affichage, il faut tenir compte de la longueur d'une ligne d'écran, c'est-à-dire quarante caractères. C'est particulièrement important lorsque vous voulez faire afficher un texte en n'ayant pas de coupure intempestive des mots.

Les deux commandes d'affichage sont :
- AFFICHE, qui inscrit votre texte et positionne le curseur au début de la ligne suivante,
- AFR (contraction de AFFICHE ET RESTE), qui inscrit votre texte et laisse le curseur à la position qui suit le dernier caractère de votre texte, au lieu de le faire passer au début de la ligne suivante.

AFR

Voyons cela :

```
AFFICHE "BONJOUR
BONJOUR
■
```

(Le curseur attend votre ordre suivant sous le caractère B)

```
REPETE 3 [ AFFICHE "BONJOUR ]
```

CHAPITRE IX

OU L'ON RETROUVE DU TEXTE

```
BONJOUR
BONJOUR
BONJOUR
```

Au contraire

```
AFR "BONJOUR
BONJOUR
```

(le curseur reste à côté du R)

```
REPETE 3 [ AFR "BONJOUR ]
BONJOURBONJOURBONJOUR
```

Les mots ne sont pas séparés.

Pour donner un aspect plus agréable à votre affichage, vous aurez intérêt à le présenter sous des pages d'écriture successives. On obtient une nouvelle page grâce au primitif VIDETEXTE, qui efface tout l'affichage existant sur l'écran, et positionne le curseur au début de la première ligne de texte disponible :

. en mode ECRANMIXTE (après une commande DESSINE), c'est la ligne supérieure des quatre lignes de texte au bas de l'écran

. en mode ECRANTEXTE (après une commande ECRIS), c'est la ligne du haut de l'écran.

ECRIS

Si l'on veut positionner le curseur à un endroit particulier de l'écran, on utilisera le primitif CURSEUR, suivi de deux entiers naturels, indiquant, pour le premier, la colonne où doit se situer le curseur (de Ø à 39, en partant de la gauche), pour le deuxième, la ligne où doit apparaître le curseur (de Ø à 23, en partant du haut). Autrement dit, CURSEUR Ø Ø met le curseur en haut à gauche de

l'écran, CURSEUR Ø 23 en bas à gauche, CURSEUR 38 23 en bas à droite.

• *Définition de procédures sans passer par l'éditeur*

Le primitif EXECUTE, qui doit être suivi d'une liste, exécute cette ligne comme s'il s'agissait d'une ligne d'instruction que vous entrez au clavier.

exemple :

```
CREE "TRIANGLE [ AVANCE 3Ø DROITE 6Ø ] ...
... AVANCE 3Ø DROITE 6Ø AVANCE 3Ø DROITE 6Ø ]
EXECUTE :TRIANGLE
```

trace ... un demi-hexagone !

Attention : la liste d'ordres à exécuter ne doit pas excéder la longueur du tampon d'édition, c'est-à-dire 255 caractères.

La liste peut très bien être terminée par une commande incomplète, si vous prenez garde à fournir au moment de l'exécution les données complémentaires. Par exemple :

```
CREE "Z [ QUOTIENT 1Ø ]
EXECUTE PHRASE :Z 5
QUE DOIS-JE FAIRE AVEC 2
```

EXECUTE

Vous verrez dans le programme PICOLO, qui figure sur la disquette utilitaire, une utilisation de EXECUTE qui permet d'enlever une ou plusieurs commandes précédemment entrées, de la même manière que la touche CE sur une calculatrice efface la dernière entrée (*)

* *Le fonctionnement de* **E**, *mis pour ENLEVE, est exactement le même que le* **CE** *de Big Trak : en appuyant une fois sur cette touche, on enlève la dernière entrée, en appuyant* **n** *fois sur cette touche, on enlève les* **n** *dernières entrées.*

CHAPITRE IX

OU L'ON RETROUVE DU TEXTE

Alors que EXECUTE ne crée pas de procédures à proprement parler (mais permet d'affecter un nom à une action), DEFINIS est un moyen de créer une procédure sans passer par l'éditeur. Cela peut être précieux soit parce qu'on ne veut pas détruire une image tracée par la tortue sur l'écran, soit pour définir une procédure quand on est déjà à l'intérieur d'une procédure.

Ce primitif ne permet cependant que la définition de procédures courtes, (à cause de la limitation à 255 caractères du tampon d'édition).

DEFINIS doit être suivi d'un nom, qui sera le titre de la procédure, et d'une liste de listes : la première est la liste des paramètres --ce sera la liste vide s'il n'y a pas de paramètres--, les listes suivantes constituent les lignes de la procédure. On ne mentionne pas FIN.

Exemple :

```
DEFINIS "ROND [ [ ] ...
   ... [ REPETE 18Ø [ AVANCE 1 DROITE 2 ] ] ]
IMPRIME ROND
POUR ROND
REPETE 18Ø [ AVANCE 1 DROITE 2 ]
FIN
```

Dans le cas d'un polygone de taille variable :

```
DEFINIS "POLY [ [ :COTE :ANGLE ] ...
... [ REPETE 36Ø / :ANGLE ] ...
      ... [ AVANCE :COTE DROITE :ANGLE ] ] ]
IMPRIME POLY
POUR POLY :COTE :ANGLE
```

```
REPETE 360 / :ANGLE ...
        ... [ AVANCE :COTE DROITE :ANGLE ]
FIN
```

A l'inverse, disposant d'une procédure, on peut avoir besoin de la considérer comme une liste de listes (ce qui permet alors de la manipuler en utilisant les primitifs que nous connaissons sur les listes). Le primitif TEXTE assure cette transformation.

Par exemple :

```
POUR AMPUTE :BIDULE
DEFINIS MOT "NOUV. :BIDULE ...
        ... SAUFDERNIER TEXTE :BIDULE
FIN
```

Ayant un polygone fermé avec :

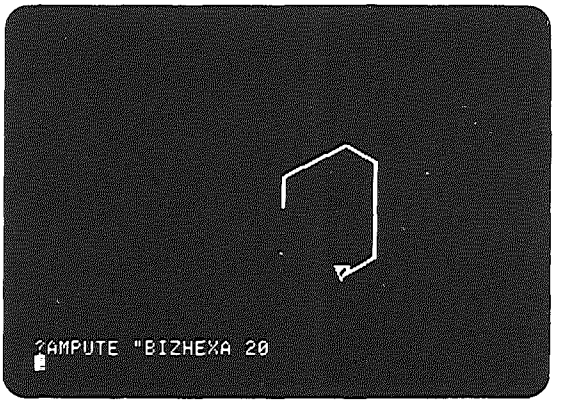
```
POUR BIZHEXA
AV 2Ø DR 6Ø AV 4Ø DR 6Ø
AV 2Ø DR 6Ø AV 6Ø DR 6Ø
AV 2Ø DR 6Ø
AV 4Ø DR 6Ø AV 2Ø
FIN
```

on obtient le dessin ci-après en tapant

```
AMPUTE "BIZHEXA
NOUV. BIZHEXA
```

CHAPITRE IX

OU L'ON RETROUVE DU TEXTE



• *Dialogue entre une procédure et un utilisateur : un "code secret"*

Nous allons voir ensemble comment rédiger une procédure qui code des messages : l'utilisateur fournit une phrase "en langage clair" et la procédure restitue la phrase codée.

Le codage est très simple :

Caractère donné	Ø	1	2	3	4	5	6	7	8	9	A	B
Caractère codé	A	B	C	D	E	F	G	H	I	J	K	L
Caractère donné	C	D	E	F	G	H	I	J	K	L	M	N
Caractère codé	M	N	O	P	Q	R	S	T	U	V	W	X
Caractère donné	O	P	Q	R	S	T	U	V	W	X	Y	Z
Caractère codé	Y	Z	Ø	1	2	3	4	5	6	7	8	9

Cette procédure, que nous allons baptiser CODAGE, se décompose en plusieurs sous-procédures :

Tout d'abord une procédure de PRESENTATION, qui comporte presque exclusivement de l'affichage. On pourra améliorer la présentation en utilisant le primitif CURSEUR, dont nous venons de parler. On aurait pu choisir de remplir complètement les lignes, ce qui obligerait à s'arrêter à **PER**, et mettre un trait d'union. Nous avons préféré avoir des lignes de longueur plus variable, mais aucun mot coupé : la lisibilité est un peu supérieure.

A la fin de cette procédure, on enregistre la phrase proposée par l'utilisateur : le primitif LISLIGNE arrête l'exécution du programme et attend la réponse de l'utilisateur.

De là, on passe à une procédure TRAITE, à laquelle on transfère en paramètre la réponse qui a été enregistrée.

```
POUR PRESENTATION
AFFICHE ...
  ... [ LE PROGRAMME QUE VOUS UTILISEZ ]
AFFICHE ...
... [ VOUS PERMET DE CODER UNE PHRASE. ]
AFFICHE [ ]
AFFICHE [ DONNEZ-MOI VOTRE PHRASE ]
AFFICHE ...
... [ JE VOUS INDIQUERAI LA MEME PHRASE ]
AFFICHE ...
... [ APRES TRANSFORMATION PAR CODAGE. ]
AFFICHE [ ] AFFICHE [ ] AFFICHE [ ]
AFFICHE [ TAPEZ MAINTENANT VOTRE PHRASE ]
```

```
AFFICHE [ ET APPUYEZ ENSUITE] ...
                    ... [SUR LA TOUCHE 'RETURN' ]
[CREE "REP LISLIGNE]
[FIN]
```

Dans la procédure TRAITE

```
[POUR TRAITE :DONNEE :NOMBRE]
[SI :DONNEE = [ ] ALORS BIZARRE] ...
... [:NOMBRE SINON AFFICHE ENCODE :DONNEE [ ]]
[FIN]
```

la première chose à vérifier c'est qu'on a effectivement quelque chose à faire, autrement dit que la phrase de l'utilisateur n'est pas vide. D'où un aiguillage sur une procédure BIZARRE, s'il n'y a rien à traiter, et sur une procédure ENCODE (toujours avec passage du paramètre) si la réponse n'est pas vide.

La procédure BIZARRE dépend d'un paramètre qui joue le rôle d'un compteur. Au premier passage, (en provenance de TRAITE), le compteur vaut 0 : l'utilisateur peut avoir mal compris le jeu, ou appuyé sur 'RETURN' par inadvertance, on lui laisse donc une deuxième chance.

```
[POUR BIZARRE :COMPTEUR]
[TESTE :COMPTEUR = Ø]
[SIVRAI AFFICHE [ N'APPUYEZ SUR 'RETURN'] ...
... [QU'APRES AVOIR TAPE ] AFFICHE] ...
... [[ LA PHRASE A CODER ] AFFICHE [ A VOUS: ]] ...
          ... [CREE "REP LISLIGNE TRAITE :REP 1]
[SIFAUX AFFICHE [ VOTRE MESSAGE EST VIDE ]] ...
... [AFFICHE [ UNE FOIS CODE, IL EST] ...
```

```
... TOUJOURS VIDE ] STOP
REPETE 3 [ AFFICHE [ ] ]
AFFICHE ...
... [ SI VOUS VOULEZ RECOMMENCER, TAPEZ ]
AFFICHE "CODAGE
FIN
```

. La procédure ENCODE fait subir le codage à la phrase proposée par l'utilisateur. On va utiliser le code ASCII des caractères pour réaliser le codage.

En informatique, chaque caractère a un code numérique appelé code ASCII (ce sigle signifie American Standard Code for Information Inter change). Le code des caractères les plus employés se trouve page suivante.

Il existe un primitif qui permet d'avoir le code ASCII d'un caractère. C'est justement ASCII. Le caractère choisi doit être précédé des guillemets.

Par exemple :

```
AFFICHE ASCII "A
65
AFFICHE ASCII "!
33
AFFICHE ASCII "4
52
```

Inversement, on peut faire apparaître un caractère dont on connait le code ASCII. Le primitif CAR réalise cela.

```
AFFICHE CAR 83
S
```

Codes de caractères ASCII

DEC = code décimal ASCII n/a = non accessible directement à partir du clavier APPLE II.

DEC	Ce qu'il faut taper	DEC	Ce qu'il faut taper	DEC	Ce qu'il faut taper
0	ctrl @	32	barre d'espacement	64	@
1	ctrl A	33	!	65	A
2	ctrl B	34	"	66	B
3	ctrl C	35	#	67	C
4	ctrl D	36	$	68	D
5	ctrl E	37	%	69	E
6	ctrl F	38	&	70	F
7	ctrl G	39	'	71	G
8	ctrl H ou ←	40	(	72	H
9	ctrl I	41	)	73	I
10	ctrl J	42	*	74	J
11	ctrl K	43	+	75	K
12	ctrl L	44	,	76	L
13	ctrl M ou RETURN	45	-	77	M
14	ctrl N	46	.	78	N
15	ctrl O	47	/	79	O
16	ctrl P	48	0	80	P
17	ctrl Q	49	1	81	Q
18	ctrl R	50	2	82	R
19	ctrl S	51	3	83	S
20	ctrl T	52	4	84	T
21	ctrl U ou →	53	5	85	U
22	ctrl V	54	6	86	V
23	ctrl W	55	7	87	W
24	ctrl X	56	8	88	X
25	ctrl Y	57	9	89	Y
26	ctrl Z	58	:	90	Z
27	ESC	59	;	91	n/a
28	n/a	60	<	92	n/a
29	ctrl shift-M	61	=	93	] (shift-M)
30	ctrl ^	62	>	94	^
31	n/a	63	?	95	n/a

Le codage que nous envisageons est simple :
si le caractère donné a un code ASCII :X compris entre 48 et 57, le caractère codé a pour code ASCII :X + 17,
si le caractère donné a un code ASCII :X compris entre 65 et 8Ø alors le caractère code a pour code ASCII :X + 1Ø
si le caractère donné a un code ASCII :X compris entre 81 et 9Ø, le caractère codé a pour code ASCII :X − 33

La procédure élémentaire est le codage d'une lettre :

```
POUR CODE.LETTRE :LETTRE
SI ALAFOIS ASCII :LETTRE > 47 ...
... ASCII :LETTRE < 58 ALORS SORS ...
                ... CAR (ASCII :LETTRE) + 17
SI ALAFOIS ASCII :LETTRE > 64 ...
... ASCII :LETTRE < 81 ALORS SORS ...
                ... CAR (ASCII :LETTRE) + 1Ø
SI ALAFOIS ASCII :LETTRE > 8Ø ...
... ASCII :LETTRE < 91 ALORS SORS ...
                ... CAR (ASCII :LETTRE) − 33
SORS :LETTRE
FIN
```

Le codage d'une lettre est utilisé de manière récursive pour coder un mot :

```
POUR CODE.MOT :TERME :CODE
TESTE :TERME = ""
SIVRAI SORS :CODE
SIFAUX CREE "CODE MOT :CODE ...
            ... CODE.LETTRE PREMIER :TERME
```

```
SORS CODE.MOT SAUFPREMIER :TERME :CODE
FIN
```

Enfin, le codage d'un mot est utilisé de manière récursive pour coder la phrase proposée par l'utilisateur :

```
POUR ENCODE :DONNEE :RESULTAT
TESTE :DONNEE = []
SIVRAI SORS :RESULTAT
SIFAUX CREE "RESULTAT PHRASE ...
... :RESULTAT CODE.MOT PREMIER :DONNEE "
SORS ENCODE SAUFPREMIER ...
                    ... :DONNEE :RESULTAT
FIN
```

La procédure de codage s'écrit donc finalement

```
POUR CODAGE
PRESENTATION
TRAITE :REP ∅
```

Attention ne commencez pas vos mots à coder par un ∅, celui-ci serait ignoré.
On peut ensuite prévoir une deuxième partie dans le programme, où un autre joueur aurait la possibilité d'essayer de décoder le message, en demandant la réponse pour quelques caractères. Au lieu d'utiliser LISLIGNE, qui oblige l'utilisateur à appuyer sur RETURN pour terminer son message (et où le message s'affiche à l'écran), on utilisera le primitif LISCAR, qui arrête l'exécution du programme jusqu'à ce qu'une touche du clavier soit pressée.

Quelles sont les différences entre LISCAR et LISLIGNE ?

LISCAR	LISLIGNE
L'utilisateur n'a pas à appuyer sur RETURN pour envoyer sa réponse.	L'utilisateur doit appuyer sur RETURN pour valider sa réponse.
Un seul caractère est pris en compte. La réponse est du type *caractère.*	Une phrase est prise en compte. La réponse est du type *liste.*
Il n'y a pas d'affichage de la réponse à l'écran.	La phrase de réponse s'affiche à l'écran.

On peut parfois avoir seulement besoin de savoir si une touche a été pressée, sans attacher aucune importance à *quelle* touche a été pressée. On dispose alors du primitif LC? qui sort VRAI si une touche a été pressée, FAUX sinon.

- ***Pour aller plus loin***

1. Rédiger une procédure qui inverse tout mot (c'est-à-dire qui, pour PORT sort TROP).
2. Rédiger une procédure qui inverse toute phrase sans inverser les mots (c'est-à-dire qui, pour LA PORTE LE CACHE, sort CACHE LE PORTE LA).
3. Rédiger une procédure qui inverse toute phrase en inversant les mots (c'est-à-dire qui, pour ERIC TE LAVA, sort AVAL ET CIRE).
4. Ecrire une procédure qui sort un mot au hasard d'une liste fournie en paramètre
5. Ecrire une procédure de dialogue avec un autre joueur sur le pluriel des mots *terminés par le son ''ou''* : vous pouvez soit choisir de donner le pluriel, l'autre joueur donnant le singulier, ou au contraire opter pour une procédure qui affiche au hasard un mot terminé par le son ''ou'' et vérifie que la réponse est correcte
6. Améliorer la présentation du programme de codage, en décalant les débuts de phrase, etc...

CHAPITRE X

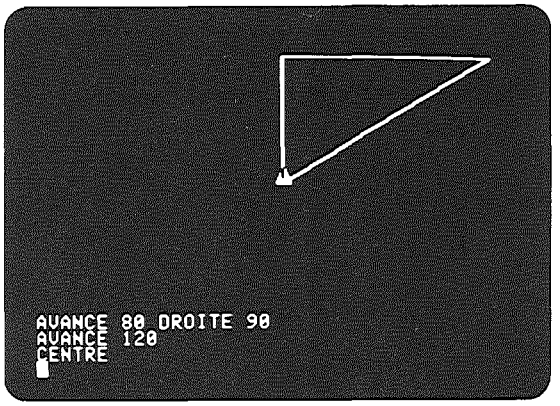
ENCORE LA TORTUE

Jusqu'à présent, nous avons presque toujours considéré les déplacements de la tortue par rapport à elle-même (AVANCE, RECULE, DROITE, GAUCHE) et non par rapport à l'écran : c'est ainsi que la tortue ne ''comprend'' pas les ordres HAUT, BAS, et que, quand elle redescend, sa droite est la gauche de l'écran. Nous allons maintenant explorer ses aptitudes à utiliser un repère absolu, par rapport à deux axes de coordonnées ayant pour origine le centre de l'écran.

• *Savoir où est la tortue*

Nous avons déjà vu que la tortue sait revenir au point de l'écran où elle apparaît quand on tape DESSINE. C'est le primitif CENTRE qui réalise ce retour à son point de départ :

```
DESSINE
AVANCE 80 DROITE 90
AVANCE 120
CENTRE
```



Ce point de l'écran joue certes un rôle particulièrement important, mais nous allons voir qu'on peut envoyer la tortue en n'importe quel point de l'écran.

Reprenons une page propre :

```
DESSINE
```

Puisqu'elle regarde vers le haut de l'écran, faisons avancer un peu la tortue :

```
AVANCE 3Ø
AFFICHE XCOR
Ø.
AFFICHE YCOR
3Ø.
```

Voilà qui est intéressant. Continuons :

```
AVANCE 8Ø
AFFICHE XCOR
Ø.
AFFICHE YCOR
11Ø.
```

11Ø ? Ah oui, 3Ø + 8Ø. Comme vous l'avez déjà deviné, le deuxième nombre indique le nombre de pas parcourus vers le haut de l'écran.

Allons maintenant vers la droite :

```
DROITE 9Ø AVANCE 5Ø
AFFICHE XCOR
5Ø.
AFFICHE YCOR
11Ø.
```

CHAPITRE X

ENCORE LA TORTUE

Cette fois-ci, la réponse à XCOR a changé. Le premier nombre a l'air d'être lié au nombre de pas parcourus vers la droite ou vers la gauche de l'écran, et on peut aussi savoir de combien de degrés la tortue a tourné globalement par rapport à sa position initiale (après DESSINE), avec AFFICHE CAP.

Reprenons un peu tout ce que nous venons d'essayer :

```
DESSINE
AVANCE 7Ø DROITE 9Ø AVANCE 35
AFFICHE XCOR
35.
AFFICHE YCOR
7Ø.
AFFICHE CAP
9Ø.
```

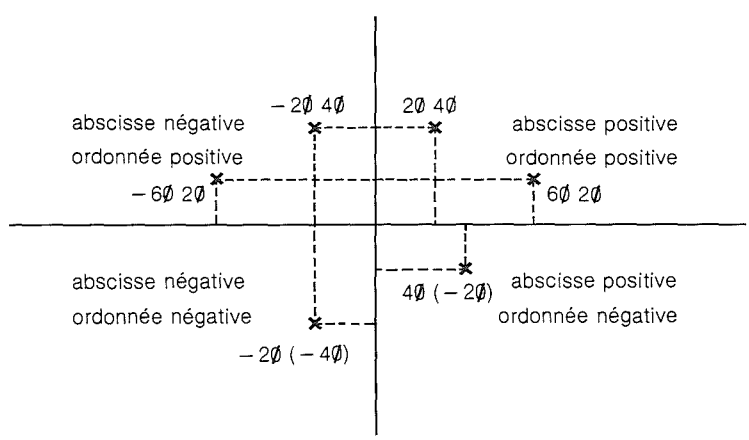
Ça a bien l'air d'être ça.

```
RECULE 12Ø
AFFICHE XCOR
-85.
AFFICHE YCOR
7Ø.
AFFICHE CAP
9Ø.
DROITE 9Ø AVANCE 12Ø
AFFICHE XCOR
-85.
AFFICHE YCOR
```

```
-50.
AFFICHE CAP
180.
```

Quand la tortue passe à gauche du centre, ou en dessous, les nombres sont précédés du signe —.

Dit d'une autre manière, l'écran peut être considéré comme une grande grille de mots croisés, mais au lieu de repérer une case par une lettre et un nombre, on repère les points par la donnée de deux nombres :
- le premier nombre (en mathématiques on dit l'*abscisse*) indique de combien de pas de tortue on s'écarte vers la droite ou vers la gauche à partir du centre ; si on va vers la droite, on utilisera un nombre positif, et si on va vers la gauche on utilisera un nombre négatif (un nombre précédé du signe —)
- le deuxième nombre (en mathématiques on dit l'*ordonnée*), indique de combien de pas de tortue on s'éloigne du centre vers le haut ou vers le bas ; si on va vers le haut, on utilisera un nombre positif, si on va vers le bas, on utilisera un nombre négatif.



Attention à l'ordre dans lequel on indique ces deux nombres : un point d'abscisse :A et d'ordonnée :B est rarement situé à la même place qu'un point d'abscisse :B et d'ordonnée :A.

• *Positionner la tortue, en utilisant l'abscisse, l'ordonnée et le cap*

Si nous voulons aller placer la tortue à l'endroit d'abscisse 8Ø, d'ordonnée 4Ø, et en lui donnant le cap 6Ø, nous pouvons utiliser la séquence suivante :

```
DESSINE
LEVEPLUME
DROITE 9Ø AVANCE 8Ø
GAUCHE 9Ø AVANCE 4Ø
DROITE 6Ø
POSEPLUME
```

Mais on peut utiliser, beaucoup plus simplement, les primitifs FIXEXY et FIXECAP.

```
DESSINE
LEVEPLUME
FIXEXY 8Ø 4Ø
FIXECAP 6Ø
POSEPLUME
```

Bien entendu, on peut aussi utiliser, pour changer l'abscisse (en gardant la même ordonnée), le primitif FIXEX, suivi de la nouvelle abscisse désirée ; et pour changer d'ordonnée (sans modifier une abscisse à laquelle on ne s'intéresse pas), le primitif FIXEY, suivi de la nouvelle ordonnée désirée.

Si l'on ne relève pas la plume, le primitif FIXEXY permet de tracer un segment de droite du point où l'on se trouve à un point d'abscisse :A et d'ordonnée :B n'importe où sur l'écran.

Il est possible d'orienter la tortue vers un point dont on connait l'abscisse :A et l'ordonnée :B. Il suffira d'écrire par exemple,

```
FIXECAP VERS 12Ø 6Ø
```

En effet, quand la tortue est à un point d'abscisse :I et d'ordonnée :J, VERS :A :B est l'équivalent de ATG :A – :I :B – :J.

• *Pour aller plus loin*

1. Ecrire une procédure qui trace les axes de coordonnées et inscrit une graduation sur chacun de ces deux axes (le pas de la graduation devrait être variable). Il n'est pas interdit de numéroter !
2. Ecrire une procédure qui trace une sinusoïde.
3. Ecrire une procédure qui trace une parabole.
4. Ecrire une procédure qui trace une tangente.
5. Ecrire une procédure qui trace un diagramme à barres. Par exemple, pour représenter une distribution de notes :

note	Ø	1	2	3	4	5	6	7	8	9	1Ø
nombre d'élèves	3	–	1	–	–	2	5	4	6	2	1

6. Réaliser une procédure qui trace la représentation en secteurs des pourcentages suivants :
A 30% B 45% C 8% D 12% E 5%.

CHAPITRE XI

COMPLEMENTS

Ce chapitre couvre des domaines variés : tout d'abord l'utilisation de disquettes pour conserver vos procédures ; puis quelques indications pour chercher les ''pépins'' dans une procédure qui ne fait pas ce qui était prévu ; enfin la possibilité d'utiliser la couleur.

• *Sauvegarder des procédures*

Dès que nous avons vu ensemble comment rédiger des procédures, nous avons commencé à parler de garder trace du travail réalisé. La façon la plus simple de conserver des informations, c'est le journal de bord : sur un cahier ou un classeur, inscrire les essais, les résultats, ainsi que le libellé des procédures. Mais, malheureusement, quand on éteint l'appareil, les procédures que l'on a créées disparaissent, et --surtout si elles sont un peu longues-- on trouve très vite fastidieux d'avoir à les retaper à chaque nouvelle séance. Il est beaucoup plus pratique de conserver ces procédures dans des fichiers sur une disquette.

Contrairement aux cassettes que vous mettez dans un magnétophone pour enregistrer des sons, et qui sont immédiatement prêtes à l'emploi, les disquettes pour un micro-ordinateur doivent être préparées (on dit *initialisées*), avant que vous puissiez vous en servir.

Si vous ne disposez pas de disquettes déjà initialisées, et que vous ayez des procédures à sauvegarder, effectuez une sauvegarde temporaire sur la disquette utilitaire, puis initialisez quelques disquettes vierges (en vous conformant aux indications données à la fin du chapitre 1), recopiez alors ce fichier temporaire sur une nouvelle disquette puis détruisez la copie provisoire sur la disquette utilitaire.

Donc, pour sauvegarder vos procédures il vous faut une disquette initialisée, en place dans le lecteur. Choisissez un nom pour le fichier que vous allez créer sur cette disquette, et qui contiendra toutes les procédures actuellement disponibles dans votre espace de travail. Mettons que nous appellions ce fichier ESSAI. Vous tapez

GARDE "ESSAI RETURN

(il faut un caractère *guillemets* avant le nom de votre fichier, mais pas de guillemets après, sauf bien sûr si votre nom de fichier comporte ce caractère).

Les disquettes où vous conservez votre travail peuvent être considérées comme un espace de rangement. Chaque fichier serait en quelque sorte un classeur, et les procédures, des feuilles contenant certains renseignements. Le nom du fichier est comme l'étiquette sur le classeur. Et de la même façon que vous avez peut-être un classeur ECOLE avec à l'intérieur des chemises ECOLE (contenant par exemple le numéro de téléphone, le règlement intérieur), BULLETINS (où vous rangez les bulletins trimestriels), SPORT (avec les renseignements sur l'Association Sportive) etc..., vous pouvez de même avoir un fichier POLYGONE avec une procédure CARRE, une procédure POLYGONE dépendant de plusieurs paramètres, une procédure RECTANGLE... etc. Mais si vous avez l'habitude de choisir des étiquettes spécialisées, au lieu de ECOLE pour le classeur vous aurez peut-être mis SCOLARITE ; de la même façon vous pouvez choisir un nom de fichier qui ne soit le nom d'aucune procédure de ce fichier (mais essayant de donner une description générale des procédures qui y sont contenues), par exemple ici POLY.

- ***Rappeler un fichier de procédures***

On peut obtenir la liste des fichiers qui existent sur une disquette en tapant

CATALOGUE

Les titres des fichiers sont alors affichés à l'écran. Chacun d'eux est suivi d'une mention précisant la nature du fichier. Par exemple, notre fichier ESSAI apparaît ainsi :

ESSAI.LOGO

Pour disposer à nouveau des procédures contenues dans le fichier ESSAI lors d'une nouvelle séance, il suffit de taper

RAMENE "ESSAI RETURN

Attention :
a. il faut des guillemets avant le nom de fichier, pas après
b. on ne mentionne pas .LOGO, qui suit le titre lors de l'affichage du catalogue.

Reprenons la comparaison avec les classeurs et les documents. Il y a une différence entre ce qui se passe quand on rappelle un fichier, et ce que l'on fait en prenant un classeur sur un rayonnage. En effet, prendre un classeur sur un rayonnage, c'est physiquement l'enlever, et laisser un espace vide à sa place pendant tout le temps où on l'utilise. Au contraire, rappeler un fichier d'une disquette ne supprime pas ce fichier de la disquette : c'est un peu comme si on en avait mis une photocopie dans la mémoire de l'ordinateur, tout en laissant l'original sur la disquette.

De la même manière, quand on garde son espace de travail en le mettant dans un fichier sur une

disquette, on ne détruit pas les procédures qui sont disponibles dans la mémoire. *Attention : si l'on utilise un nom de fichier déjà existant, le précédent fichier portant ce nom est détruit.*

On peut rappeler dans la mémoire le contenu de plusieurs fichiers. Cela peut permettre de fondre plusieurs fichiers en un seul, en tapant GARDE et un nom de fichier après avoir rappelé plusieurs fichiers.

• *Sauvegarde et rappel de dessins*

Les procédures qui utilisent la tortue pour réaliser des tracés peuvent être sauvegardées sur disquettes, et leur rappel permet de reconstituer les tracés en utilisant les procédures.

Mais on a parfois de très intéressantes figures obtenues en mode pilotage : alors, est-il impossible d'en garder trace autrement qu'en les dessinant sur son journal de bord ou en en prenant une photographie ?

Heureusement, un primitif permet aussi de copier un dessin sur une disquette. C'est GARDEDESSIN. Vous avez besoin d'un nom de fichier, par exemple PAYSAGE, et vous tapez

GARDEDESSIN "PAYSAGE RETURN

Lorsque vous faites afficher le catalogue, vous trouvez

PAYSAGE.DESSIN

qui vous indique que ce fichier est un fichier de dessin. Pour le rappeler, si dans quelques jours ou dans quelques mois vous voulez le montrer à des amis, il faut taper

RAMENEDESSIN "PAYSAGE RETURN

L'image est aussitôt reconstituée sur votre écran.

- ***Se débarrasser de fichiers devenus inutiles***

Si vous voulez détruire un fichier de procédures, nommé PAT (repérable grâce à .LOGO qui suit le nom du fichier quand on fait afficher le catalogue à l'écran), tapez :

DETRUISFICHIER "PAT

Si vous voulez détruire un fichier de figure, nommé BOB (repérable grâce à .DESSIN qui suit le nom du fichier quand on fait afficher le catalogue à l'écran), tapez

DETRUISDESSIN "BOB

- ***Aurevoir***

Si plusieurs utilisateurs travaillent successivement, chacun, après avoir sauvegardé sur disquette les procédures qu'il souhaite conserver, laissera au suivant un espace de travail tout propre en tapant

AUREVOIR

- ***La chasse aux pépins***

Il arrive souvent qu'une procédure ne réalise pas ce qu'on avait prévu. C'est très souvent ce résultat inattendu qui donne de nouvelles idées, ou fait découvrir des propriétés auxquelles on n'avait pas pensé.
Mais en général, on souhaite quand même modifier la procédure pour arriver au fonctionnement escompté. Il suffit souvent de suivre le déroulement de la procédure pour retrouver à partir de quel endroit elle ne fait plus ce que vous aviez prévu. Mais si vous restez perplexe pour trouver le pépin qui vous ennuie, plusieurs primitifs peuvent vous aider.

TRACE

Le primitif TRACE est un des plus utiles pour repérer les pépins qui font qu'une procédure ne fonctionne pas comme vous vous y attendiez.
Quand vous tapez

TRACE

LOGO adopte un nouveau mode de fonctionnement pour l'exécution des procédures : pour chaque procédure qui s'exécute (soit que vous tapiez son titre au clavier, soit qu'elle soit appelée au cours de l'exécution d'une autre procédure), vous êtes averti de cette exécution et de la valeur des paramètres. De plus, chaque ligne d'instructions de la procédure est affichée à l'écran, et l'exécution est suspendue jusqu'à ce que vous appuyiez sur l'une des touches du clavier (n'importe laquelle). Cette ligne est alors exécutée, puis la suivante s'affiche et attend que vous pressiez une touche, et ainsi de suite.
Voyons cela par exemple avec la procédure :

```
POUR CARROUSEL :LONG
AVANCE :LONG
DRAPEAU :LONG / 2
SI CAP = Ø ALORS STOP
CARROUSEL :LONG
FIN
```

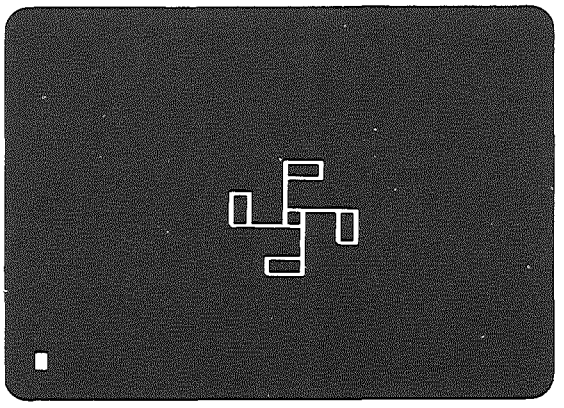
avec

```
POUR DRAPEAU :PAS
AVANCE :PAS
RECTANGLE :PAS :PAS * 2
FIN
```

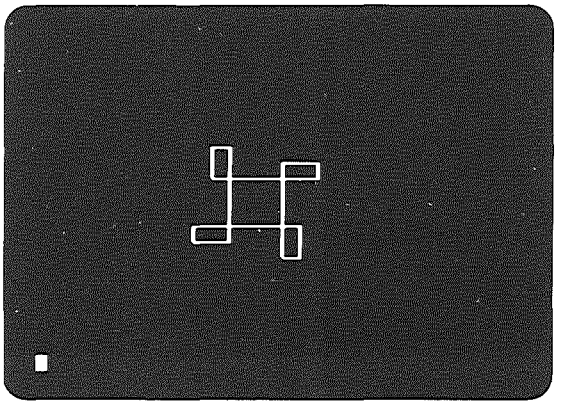
et

```
POUR RECTANGLE :COTE1 :COTE2
AVANCE :COTE1 DROITE 90
AVANCE :COTE2 DROITE 90
AVANCE :COTE1 DROITE 90
AVANCE :COTE2
FIN
```

Alors qu'on s'attend à voir



on obtient



qui n'est pas vilain, mais ne réalise pas la figure escomptée.

```
TRACE
CARROUSEL 3Ø
ICI COMMENCE CARROUSEL 30
AVANCE :LONG
(tapez sur une touche)
DRAPEAU :LONG / 2
(tapez sur une touche)
ICI COMMENCE DRAPEAU 15
AVANCE :PAS
(tapez sur une touche)
(etc.)
```

On verra facilement, (et même d'autant plus facilement qu'on aura pris l'habitude d'écrire des lignes d'instructions relativement courtes –en règle générale pas plus d'une ligne d'écran–), où le tracé diverge du tracé attendu, et comment y remédier.

Le primitif DETRACE remet en mode d'exécution normal. Soulignons que les primitifs TRACE et DETRACE sont des primitifs analogues à tous les autres primitifs, et qu'ils peuvent donc être utilisés à l'intérieur d'une procédure. Lorsqu'on a remarqué, que toute une partie de la réalisation est satisfaisante, on peut éditer la procédure et insérer les primitifs TRACE et DETRACE autour de la portion qui semble poser problème.

PAUSE et CONTINUE.

Vous pouvez insérer le primitif PAUSE dans une procédure : cette commande a pour effet d'arrêter l'exécution, qui ne sera reprise que lorsque vous taperez CONTINUE (**CO** en abrégé). Introduisons par exemple PAUSE en première ligne de RECTANGLE, comme suit

```
POUR RECTANGLE :COTE1 :COTE2
PAUSE
AVANCE :COTE1 DROITE 90
AVANCE :COTE2 DROITE 90
AVANCE :COTE1 DROITE 90
AVANCE :COTE2
FIN
```

Quand l'exécution de CARROUSEL aura appelé RECTANGLE, vous obtiendrez le message :

```
PAUSE, LIGNE
    PAUSE
AU NIVEAU 3 DE RECTANGLE
N3?
```

L'indication NIVEAU 3 vous précise qu'il y a actuellement trois procédures en cours d'exécution, autrement dit que RECTANGLE a été appelé par une procédure qui était elle-même appelée par la procédure CARROUSEL dont vous avez demandé l'exécution au clavier.

Tous les messages d'erreur émis lors de l'exécution d'une procédure comportent également le niveau où l'erreur s'est produite.
Le message d'erreur indique :
- la nature du problème rencontré
- la ligne où ce problème se situe
- le niveau où l'erreur s'est produite
- le nom de la procédure où se trouve l'erreur
Pour une procédure qui n'est appelée par aucune autre, mais dont vous demandez directement l'exécution en tapant son titre au clavier, le niveau est 1.
Si la procédure "fautive" est appelée par une procédure de niveau 1, cette procédure erronée est de niveau 2, et ainsi de suite.

L'indication N3? vous rappelle que vous êtes au niveau 3 avec suspension d'exécution, et que vous pouvez taper absolument n'importe quel primitif LOGO, exactement comme si vous étiez en mode pilotage : vous pouvez ici faire pivoter la tortue, ou l'avancer..., mais vous pouvez aussi faire afficher les valeurs des paramètres de la procédure où vous êtes en PAUSE.

Pour poursuivre l'exécution, taper **CONTINUE** (en abrégé **CO**).

Si vous voulez sortir de la procédure, tapez CTRL **G**.

Vous pouvez provoquer une pause à tout moment au cours de l'exécution d'une procédure, en tapant CTRL **Z**. De la même manière, vous reprendrez l'exécution avec CONTINUE.

- ***Utiliser la couleur***

Si vous avez une carte couleur vous permettant de brancher votre Apple II sur votre poste de télévision couleur (par exemple en France une carte du type Chat Mauve) vous pouvez tracer en couleur avec la tortue, et également colorier la page sur laquelle la tortue se déplace.

Sur fond noir :

COLOREPLUME 0 trace en noir sur fond noir (c'est ce qui permet d'effacer un trait)

COLOREPLUME 1 trace en blanc

COLOREPLUME 2 trace en vert

COLOREPLUME 3 trace en mauve

COLOREPLUME 4 trace en orange
COLOREPLUME 5 trace en bleu
COLOREPLUME 6 donne un effet d'inverseur, c'est-à-dire efface en repassant sur un trait précédemment tracé, et trace quand il n'y a pas de dessin préexistant.

(L'inversion ne fonctionne que partiellement sur fond 0 ; il vaut mieux utiliser le fond 6).

Trait noir ou blanc sur fond de couleur :

COLOREFOND 0	fond noir
COLOREFOND 1	fond blanc
COLOREFOND 2	fond vert
COLOREFOND 3	fond mauve
COLOREFOND 4	fond orange
COLOREFOND 5	fond bleu
COLOREFOND 6	fond noir

Traits de couleur sur fond de couleur :

*Là, il faut vous attendre à pas mal de surprises, comme d'obtenir un trait vert en utilisant COLOREPLUME 4 (*a priori *orange) sur un fond mauve obtenu par COLOREFOND 3. De plus, si vous changez la couleur du fond après avoir déjà tracé en couleur, il peut se produire des mouchetages curieux.*

• *Pour aller plus loin*

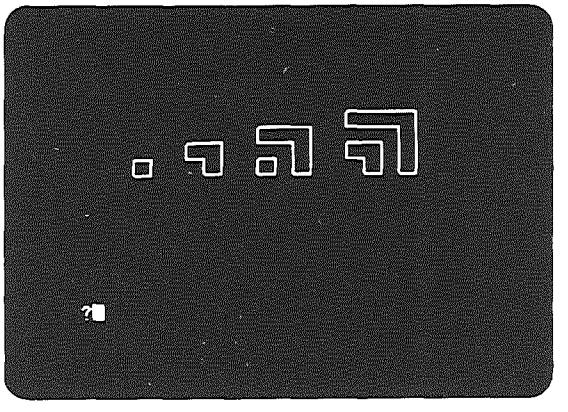
1. Insérez plusieurs PAUSE dans les procédures constituant CARROUSEL et faites afficher les valeurs des variables

2. Corrigez la procédure CARROUSEL après l'avoir mise en mode TRACE

3. Faites-vous un tableau des effets de couleurs obtenus en traçant en couleur sur fond de couleur ;

Notez en particulier quelles combinaisons assurent les effaçages.

4. Utilisez l'effet d'inverseur pour réaliser un carré à malices



5. Utilisez l'effet d'inverseur pour réaliser un lampion qui s'enroule et se déroule

CHAPITRE XII

LA DISQUETTE UTILITAIRE

Si ce n'est pas déjà fait, vous devriez vous préparer une copie de la disquette utilitaire : pour cela, utilisez le programme COPYA de la disquette SYSTEM MASTER qui vous a été livrée en même temps que votre Apple II.

La disquette utilitaire comporte trois sortes de fichiers :

- des fichiers de démonstration : CIBLE, CIRCUIT, SPIRALES, FUSEE, TET.

- des fichiers d'exploration : MUSIQUE, DYNA-TORTUE, ANIMAL, PICOLO.

- des fichiers utilitaires proprement dits : ASSEMBLEUR, EDIT.FORME, APPRENDS, TORTUE, M&L, FID.

Ces divers fichiers vous fournissent un certain nombre de procédures intéressantes ou utiles : vous pourrez vous en inspirer pour vous en fabriquer d'autres.

Pour utiliser les fichiers de la disquette utilitaire :

1. charger EDI-LOGO.

2. retirer la disquette langage et insérer votre copie de la disquette utilitaire.

3. pour avoir la liste des fichiers disponibles, taper **CATALOGUE**.

4. pour ramener un fichier, taper **RAMENE** ''nom du fichier (sans la mention .LOGO).

• *Les fichiers de démonstration*

CIBLE

Taper

RAMENE "CIBLE

Vous disposez des commandes

AV pour AVANCE
RE pour RECULE
DR pour DROITE
GA pour GAUCHE

et vous devez amener la flèche dans la cible.

CIRCUIT

Taper

RAMENE "CIRCUIT

Trace un circuit comme une piste de circuit automobile, et positionne la tortue au départ.

Les commandes sont : → pour faire pivoter la tortue dans le sens des aiguilles d'une montre, ← pour la faire pivoter dans le sens inverse, la barre d'espacement pour donner une impulsion à la tortue.

Il s'agit bien sûr de faire tourner la tortue le plus longtemps possible sur la piste, sans heurter les bords.

SPIRALES

Taper

RAMENEDESSIN "SPIRALES

Reproduit sur l'écran un dessin à base de spirales.

FUSEE

Taper

RAMENE "FUSEE

puis

DEBUT

Démonstration de l'utilisation d'une tortue ayant une forme différente.

Utilise les fichiers FUSEE.AUX et FUSEE.FORMES qui sont une illustration d'une utilisation typique de formes définies par l'utilisateur.

TET

Taper

RAMENE "TET

puis

TET 1ØØ 1

Essayez les niveaux suivants de récursivité avec TET 1ØØ 2 , TET 1ØØ 3 etc...

Vous verrez le dessin se compliquer petit à petit.

- ***Les fichiers d'exploration***

MUSIQUE

Procédures utilisées et décrites dans le chapitre 7.

Taper

RAMENE "MUSIQUE

Il y a trois fichiers : MUSIQUE.LOGO qui contient les procédures, MUSIC.SRC.LOGO, et MUSIC.BIN qui contiennent des programmes en langage assembleur pour jouer les notes.

DYNATORTUE

Taper

RAMENE "DYNATORTUE

Vous vous trouvez alors devant une tortue dynamique (c'est-à-dire un objet newtonien) similaire à celle utilisée dans CIRCUIT. Celle-ci peut, en plus laisser une trace (P comme POSEPLUME), ou non (L comme LEVEPLUME), recevoir une impulsion plus forte (touche +) , ou moins forte (touche −).

Quelques suggestions : faites-lui tracer un angle droit, un carré, un cercle,...

ANIMAL

Taper

RAMENE "ANIMAL

puis

ANIMAL

Ce programme est capable d'augmenter ses connaissances sur les animaux, en fonction des indications que vous lui fournissez.

Si vous voulez conserver un vocabulaire enrichi pour une prochaine utilisation, tapez, quand vous aurez fini d'utiliser ce programme :

GARDE "ANIMAL

Si au cours d'une séance vous voulez retourner à l'état de départ, tapez

CTRL **G**

puis

DEBUT

PICOLO

Tapez

RAMENE "PICOLO.PARLE

puis

DEBUT

et suivez les instructions.

Vous pouvez réaliser des dessins avec une tortue qui répond à des commandes simplifiées.

Ne tapez sur une touche que lorsque le curseur clignote.

Si vous avez une carte de synthèse de la parole PORTE-PAROLE d'EDICIEL, les explications vous seront données de vive voix par la machine. Ce programme convient particulièrement bien à l'initiation de très jeunes enfants sachant à peine lire.

Si vous voulez éviter la présentation orale,

tapez

RAMENE "PICOLO.MUET

au lieu de

RAMENE "PICOLO.PARLE

- ***Les fichiers utilitaires proprement dits***

ASSEMBLEUR, AMODES et OPCODES, ainsi que ADRESSES, sont utilisés en assembleur. Voir le manuel de référence.

EDIT.FORME sert à modifier la forme de la tortue. Voir le manuel de référence.

APPRENDS

Taper

```
RAMENE "APPRENDS
```

La procédure APPRENDS permet de définir une procédure sans passer par l'éditeur.

Exemple d'utilisation

```
NOM DE LA PROCEDURE : CARRE
PARAMETRES : :COTE
>SI CAP = Ø STOP
>AVANCE :COTE DROITE 9Ø
>CARRE :COTE
>FIN
CARRE EST DEFINI
```

TORTUE

Taper

```
RAMENE "TORTUE
```

Contient des procédures pour manipuler plus agréablement la tortue.

VISIBLE? sort "VRAI ou "FAUX selon que la tortue est montrée ou cachée.

LP? sort "VRAI ou "FAUX selon que la plume est levée ou posée.

POS sort sous forme de liste [XCOR YCOR CAP].

EFFACEUR permet d'effacer un trait en repassant dessus (valable également pour des tracés en couleur).

INVERSEUR permet d'effacer un trait en repassant dessus, et de tracer si l'emplacement est vierge.

ENCRE remet en mode de dessin normal après l'utilisation d'EFFACEUR ou d'INVERSEUR.

POINT trace un point à l'emplacement de coordonnées X et Y spécifiées.

DELTAXY déplace la tortue d'une valeur spécifiée selon l'axe des X et d'une valeur spécifiée selon l'axe des Y.

FPOS positionne la tortue à l'abscisse donnée, l'ordonnée donnée, avec le cap donné.

M&L

Taper

RAMENE "M&L

Contient des procédures pour manipuler plus agréablement les mots et les listes, telles que :

VIDE? teste si un objet (mot ou liste) est vide, et sort l'information.

COMPTE compte le nombre de caractères dans un mot, ou le nombre d'éléments dans une liste.

NIEME sort le n-ième caractère d'un mot, ou le n-ième élément d'une liste.

FID

Taper **RAMENE "FID**

Permet de détruire, verrouiller, renommer des fichiers.

CHAPITRE XIII

PISTES POUR LES PROJETS

Les propositions de solutions que vous trouverez dans ce chapitre ne sont là que pour vous donner des idées si vous êtes vraiment en panne. Elles n'ont pas l'ambition d'être la meilleure réponse possible.

• *Piloter une tortue*

1. Du centre de l'écran au bord supérieur, il y a 120 pas, et 119 pas du centre au bord inférieur en PLEINECRAN ou 79 pas en ECRANMIXTE. Du centre au bord gauche ou au bord droit, il y a 140 pas et 139 pas respectivement.

La dimension d'un pas "horizontal" est immuable, mais la dimension d'un pas "vertical" peut être modifiée en utilisant le primitif .ASPECT (cf. chapitre 3 page 45)
Les valeurs qui viennent d'être indiquées correspondent à .ASPECT .8, qui est la valeur par défaut.
Si on choisit .ASPECT 1, il n'y a plus que 96 pas pour arriver en haut de l'écran, et 95 pour arriver en bas, quand on part du centre.

2. En ECRANMIXTE, la diagonale représente 343 pas En PLEINECRAN, la diagonale représente 366 pas de tortue.

Ces valeurs correspondent à .ASPECT .8 .

3. Le carré peut se faire ordre par ordre. On prend, pour fixer les idées, une longueur de cinquante pas pour le côté.

```
AVANCE 5Ø DROITE 9Ø AVANCE 5Ø DROITE 9Ø
```

```
AVANCE 5Ø DROITE 9Ø AVANCE 5Ø
```

(La tortue est alors revenue à sa position de départ)

```
DROITE 9Ø
```

(La tortue est alors revenue à sa position de départ, avec son orientation initiale).

Ou en utilisant le primitif REPETE :

REPETE 4 [AVANCE 5Ø DROITE 9Ø]

Pour le rectangle, choisissons 6Ø et 8Ø pour la largeur et la longueur :

AVANCE 6Ø DROITE 9Ø AVANCE 8Ø DROITE 9Ø

AVANCE 6Ø DROITE 9Ø AVANCE 8Ø

remet la tortue dans sa position initiale après avoir tracé un rectangle ''couché sur sa longueur''

DROITE 9Ø

remet la tortue dans sa position initiale avec l'orientation de départ.

AVANCE 8Ø DROITE 9Ø AVANCE 6Ø DROITE 9Ø

AVANCE 8Ø DROITE 9Ø AVANCE 6Ø

réalise un rectangle ''debout sur sa largeur''.
Si l'on veut utiliser le primitif REPETE, on aura par exemple

REPETE 2 [AVANCE 8Ø DROITE 9Ø] ...
... AVANCE 6Ø DROITE 9Ø]

4. La réalisation d'un triangle n'est pas très simple : en effet, seul le triangle équilatéral (trois côtés de même longueur, trois angles de 6Ø degrés), peut être tracé sans connaissances particulières. Toutefois

REPETE 3 [AVANCE 5Ø DROITE 6Ø]

ne réalise pas un triangle équilatéral, mais un demi-hexagone. En effet, à chaque sommet du triangle, la tortue tourne, non de l'angle intérieur de la figure mais de son supplément (ici 12Ø degrés). C'est donc

REPETE 3 [AVANCE 5Ø DROITE 12Ø]

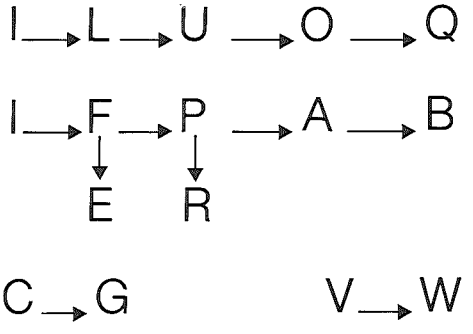
qui donnera le tracé voulu.

Pour le triangle rectangle isocèle, il faut déjà penser à utiliser la symétrie, ou la relation sur la somme des trois angles d'un triangle. De plus, la mesure de l'hypoténuse nécessite un petit calcul,... que l'on peut éviter en utilisant une technique d'approximations successives, pour refermer le triangle. par exemple :

```
AVANCE 5Ø DROITE 9Ø AVANCE 5Ø
DROITE 135 AVANCE 71
```

5. Pour pouvoir utiliser CTRL **P** facilement, il est bon de prendre l'habitude de taper un espace après chaque instruction, même en bout de ligne. Le caractère ''espace'' est alors mémorisé dans le tampon, et le rappel par CTRL **P** peut être employé à plusieurs reprises.

6. On commencera par des formes faciles, comme les lettres et chiffres en cristaux liquides qui apparaissent à l'affichage de certaines calculatrices. Remarquer que certaines lettres forment des familles amène à l'idée de procédure, qui va être développée au chapitre suivant.



- ***Enseigner des mots nouveaux***

1. Un drapeau
Voici une solution. D'autres donnent des résultats aussi satisfaisants.

```
POUR DRAPEAU
AVANCE 2Ø RECTENGLE
FIN
```

2. Un "Quatredrapeaux". On va, bien sur, utiliser la procédure précédente, après l'avoir modifiée, de manière à ramener la tortue à son point de départ, avec son orientation initiale.

```
POUR DRAPEAUX
DRAPEAU
DROITE 9Ø RECULE 2Ø
FIN
```

Il est alors très facile d'obtenir

```
POUR QUATRE.DRAPEAUX
REPETE 4 [ DRAPEAUX DROITE 9Ø ]
FIN
```

L'idée de ramener la tortue à son point de départ avec son orientation initiale, est une technique générale, extrêmement fructueuse parce qu'elle permet d'y voir toujours clair.
3. Une rosace.
Élle utilise la procédure RECTANGLE définie dans ce chapitre.

```
POUR ROSACE
REPETE 4 [ RECTANGLE ]
FIN
```

4. Un tunnel de carrés.
On a visiblement besoin d'une procédure paramétrée. Nous allons donc utiliser CARREAU, avec deux paramètres ayant la même valeur.

```
POUR TUNNEL
CARREAU 2Ø 2Ø
CARREAU 3Ø 3Ø
CARREAU 4Ø 4Ø
CARREAU 5Ø 5Ø
CARREAU 6Ø 6Ø
FIN
```

5. Un gâteau d'anniversaire (ou une pyramide aztèque)

On va commencer par une procédure ETAGE, qui reprend CARREAU.

```
POUR ETAGE :BASE :HAUTEUR
CARREAU :HAUTEUR :BASE
AVANCE :HAUTEUR DROITE 9Ø
AVANCE :HAUTEUR GAUCHE 9Ø
FIN
```

Ceci positionne la tortue pour le début de l'étage suivant. Il s'agit en somme d'une variante de la règle du retour à la position et à l'orientation initiales. On a alors

```
POUR GATEAU
ETAGE 7Ø 1Ø
ETAGE 5Ø 1Ø
ETAGE 3Ø 1Ø
```

```
ETAGE 1Ø 1Ø
CACHETORTUE
FIN
```

6. Une rangée de drapeaux.
On va réutiliser la procédure DRAPEAUX du deuxième exemple.

```
POUR RANGEE
LEVEPLUME
GAUCHE 9Ø AVANCE 5Ø DROITE 9Ø
POSEPLUME
DRAPEAUX
LEVEPLUME
DROITE 9Ø AVANCE 5Ø GAUCHE 9Ø
POSEPLUME
DRAPEAUX
LEVEPLUME
DROITE 9Ø AVANCE 5Ø GAUCHE 9Ø
POSEPLUME
DRAPEAUX
FIN
```

7. Un hexagone
Le "pépin" rencontré pour l'exercice sur le triangle au chapitre précédent nous fournit une indication pour la réalisation de l'hexagone.

```
POUR HEXA
REPETE 6 [ AVANCE 5Ø DROITE 6Ø ]
FIN
```

Il est très important de savoir utiliser des essais qui n'avaient pas donné le résultat attendu et d'en tirer des idées nouvelles.

8. Un Octogone

```
POUR OCTO
REPETE 8 [ AVANCE 3Ø DROITE 45 ]
FIN
```

9. Si vous n'avez pas réussi à faire tracer à la tortue quelque chose qui ressemble à un rond, attendez le chapitre 6, ou reportez-vous y pour y trouver des idées. De même pour le chat.

- ***Mieux qu'une calculette***

1. Une procédure qui sort l'opposé.

```
POUR OPPOSE :NOMBRE
SORS - :NOMBRE
FIN
```

2. Une procédure qui sort l'inverse

```
POUR INVERSE :NOMBRE
SI :NOMBRE = Ø AFFICHE ...
... [ J'AI UN PROBLEME POUR L'INVERSE DE ...
                          ... ZERO ] STOP
SORS 1 / :NOMBRE
FIN
```

Si on omet le test en première ligne, les éventuels étourdis qui essaieraient de calculer l'inverse de 0 se verraient annoncer par la machine :

```
TU ESSAIES DE DIVISER PAR ZERO
```

3. L'essentiel est bien sûr de vérifier que 1. = 1

4. Il y a possibilité de scinder ce problème en deux sous-problèmes : a) calculer un carré, b) calculer une somme ;

Exemple :

```
POUR CARRE :NOMBRE
SORS :NOMBRE * :NOMBRE
FIN
```

et

```
POUR CARHY :COTE1 :COTE2
SORS CARRE :COTE1 + CARRE :COTE2
FIN
```

Bien entendu, on peut compacter ces deux tâches en une seule, mais on perd un peu en lisibilité. Il est important d'essayer d'appliquer la devise : ''une procédure pour chaque tâche, et chaque tâche à une procédure''.

5. Là encore, scinder en sous-problèmes : a) calculer le quotient arrêté à la virgule, b) calculer le reste

```
POUR QUOTENTIER :DIVR :DIVDE
SORS ENTIER :DIVR / :DIVDE
FIN
POUR RESTEDEC :DIVR :DIVDE
SORS :DIVR - :DIVDE * QUOTENTIER ...
... :DIVR :DIVDE
FIN
```

6. Comparer ENTIER et ARRONDI.
7. Comparer ENTIER et le nombre.

- ***Manipuler des mots et des listes***

1. Le mot vide ne convient pas, car

```
PREMIER ''
```

amène le message

```
PREMIER N'AIME PAS RECEVOIR
```

(On ne peut pas prendre le ''PREMIER'' de quelque chose qui n'a pas d'élément). Par contre, un mot d'une seule lettre est égal à son premier.
Puisque PREMIER sort un caractère, c'est la seule solution.
2. Solution similaire.
3. Puisque PREMIER sort un caractère, il faut que SAUFPREMIER soit aussi un caractère. Ce caractère peut être celui sorti par PREMIER (mot d'une seule lettre), ou un deuxième caractère (mot de deux lettres) si en plus c'est le même que le premier.
Exemples : ''BB ''H
4. Cette fois-ci, on aboutit à un mot de deux caractères, mais ces deux caractères peuvent ne pas être les mêmes.
Exemples : ''BA ''ZZ
5. Bien sûr, si on se limite à un seul verbe, LAVER par exemple, il suffit de faire de l'affichage :

```
POUR CONJUGUE.LAVER
AFFICHE [ JE LAVE ]
AFFICHE [ TU LAVES ]
AFFICHE [ IL LAVE ]
AFFICHE [ NOUS LAVONS ]
AFFICHE [ VOUS LAVEZ ]
AFFICHE [ ILS LAVENT ]
FIN
```

Mais bien sûr, passer du lavage au repassage oblige à tout recommencer. Il est donc intéressant que le verbe soit utilisé comme paramètre. On suppose ce verbe donné à l'infinitif :

```
POUR CONJUGUE :VERBE
CREE "RADICAL SAUFDERNIER ...
                    ... SAUFDERNIER :VERBE
AFFICHE PHRASE "JE MOT :RADICAL "E
AFFICHE PHRASE "TU MOT :RADICAL "ES
AFFICHE PHRASE "IL MOT :RADICAL "E
AFFICHE PHRASE "NOUS MOT :RADICAL "ONS
AFFICHE PHRASE "VOUS MOT :RADICAL "EZ
AFFICHE PHRASE "ILS MOT :RADICAL "ENT
FIN
```

Bien entendu, il faut quelques aménagements pour AIMER, MANGER, APPELER, ALLER et ensuite se préoccuper des deuxième et troisième groupes.
6. Même démarche : voir d'abord ce qui se passe sur un exemple :

```
POUR TABLE7
AFFICHE [7 * 1 = 7]
AFFICHE [7 * 2 = 14]
AFFICHE [7 * 3 = 21]
AFFICHE [7 * 4 = 28]
AFFICHE [7 * 5 = 35]
AFFICHE [7 * 6 = 42]
AFFICHE [7 * 7 = 49]
AFFICHE [7 * 8 = 56]
```

```
AFFICHE [7 * 9 = 63]
AFFICHE [7 * 10 = 70]
FIN
```

Puis on pensera qu'il faut avoir un paramètre :

```
POUR TABLE :NOMBRE
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 1 = ] :NOMBRE
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 2 = ] :NOMBRE * 2
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 3 = ] :NOMBRE * 3
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 4 = ] :NOMBRE * 4
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 5 = ] :NOMBRE * 5
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 6 = ] :NOMBRE * 6
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 7 = ] :NOMBRE * 7
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 8 = ] :NOMBRE * 8
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 9 = ] :NOMBRE * 9
AFFICHE PHRASE ...
... :NOMBRE PHRASE [ * 10 = ] :NOMBRE * 10
FIN
```

7. Aucune difficulté

```
POUR MOT.DOUBLE :M
SI NON MOT? :M ALORS AFFICHE ...
                ... [ IL ME FALLAIT UN MOT ] STOP
SORS MOT :M :M
FIN
```

- ***Procédures plus sophistiquées***

1. Contrairement à ce que vous aviez probablement prévu, les ronds que vous avez obtenus sont tous approximativement de même taille.
2. Un ROND ayant des valeurs un peu plus grandes pour DROITE est ''visiblement'' un polygone (octogone pour 45, hexagone pour 60, carré pour 90).
3. On pourra également chercher les figures engendrées par des procédures dont la ''brique'' de base est

```
AVANCE :COTE DROITE :ANGLE
AVANCE :COTE * 2 DROITE :ANGLE
```

4. On pourra également explorer les figures engendrées par des procédures dont la ''brique'' de base est :

```
AVANCE :COTE DROITE :ANGLE
 AVANCE :COTE DROITE :ANGLE * 2
```

5. En particulier, si le paramètre vaut Ø, on obtient un polygone, et si le paramètre vaut 1, on pourra colorer toute une surface.
6. Il faut ''occuper'' une procédure paramétrée qui ne fera rien d'autre que ''perdre du temps''. Par exemple :

```
POUR ATTENDS :N
```

```
SI :N = Ø STOP
CREE "VARIABLE :N
ATTENDS :N - 1
FIN
```

Bien entendu, il faudra étalonner cette procédure pour savoir quelle valeur de :N correspond à une durée donnée.

7. Cette procédure est calquée sur celle de l'exercice 7 du chapitre précédent. Si on veut seulement afficher la phrase avec des répétitions progressives, on utilisera une procédure avec récursion terminale (c'est-à-dire en dernière ligne)

```
POUR REPETE.LISTE :L
SI NON LISTE? :L ALORS AFFICHE ...
        ... [ IL ME FALLAIT UNE LISTE ] STOP
AFFICHE :L
REPETE.LISTE PHRASE :L :L
FIN
```

8. Il est évident que, si le mot est vide, on connait la réponse qui est Ø. Si le mot n'est pas vide, on ajoute 1 au résultat trouvé pour le mot amputé de son premier caractère.

```
POUR COMPTE.MOT :M
SI :M = "  SORS Ø
SORS 1 + COMPTE.MOT SAUFPREMIER :M
FIN
```

On transposera facilement pour COMPTE.LISTE qui compte les éléments d'une liste, et COMPTE. CAR qui totalise le nombre de caractères de chacun des mots d'une liste.

- ***Boite à musique et tambourin***

1. L'air bien connu de Mozart s'obtient avec :

```
MI ME MA MA MI ME
```

2. Au Clair de la Lune est obtenu par

```
CLI CLA CLI CLA CLO CLE CLI CLA
```

3. J'ai du bon tabac se retrouve avec

```
TI TA TI TO TE TU TE TU TI TA TI TO
```

4. Un air moins connu : ''Jean de la Lune''. L'air original est

```
JE JA JE JO JU JI
```

5. Là encore un air un peu connu : ''Sainte Maritaine''

```
SI SI SI SA SO SO SE
```

- ***Où l'on retrouve les nombres***

1. Il s'agit de sortir le plus petit des deux nombres

```
POUR MINIMUM :A :B
TESTE :A < :B
SIVRAI SORS :A
SIFAUX SORS :B
FIN
```

2. Il s'agit de sortir le maximum de la paire { nombre donné, opposé à ce nombre}

```
POUR VALEUR.ABSOLUE :A
TEXTE :A < – :A
SIVRAI SORS – :A
```

```
SIFAUX SORS :A
FIN
```

On peut aussi comparer le nombre donné à Ø :

```
POUR VAL.ABS :A
SI :A > Ø ALORS SORS :A SINON SORS – :A
FIN
```

3. Calqué sur la procédure MAXIMUM traitée dans ce chapitre.
Pour aller plus loin, concevoir une procédure qui sort la suite ordonnée à partir d'une suite "en vrac".
4. Comme la suite des multiples est infinie, on donnera en deuxième paramètre la borne qui sert d'arrêt

```
POUR MULTIPLES :NOMBRE :BORNE
(AFFICHE [ VOICI LA LISTE DES MULTIPLES ] ...
... DE ] :NOMBRE [ COMPRIS ENTRE Ø ET ] ...
                    ... :NOMBRE * :BORNE )
MULT :NOMBRE :BORNE Ø
AFFICHE [ C'EST TOUT ]
FIN
```

avec

```
POUR MULT :NOMBRE :BORNE :DEPART
SI :DEPART > :BORNE STOP
AFR :NOMBRE * :DEPART AFR CAR 32
MULT :NOMBRE :BORNE :DEPART + 1
FIN
```

(Il y a un pépin pour une valeur particulière de :NOMBRE)

5.
```
POUR DIVISEURS :NOMBRE
AFFICHE PHRASE [ VOICI LA LISTE ] ...
... DES DIVISEURS NATURELS DE ] :NOMBRE
SI :NOMBRE = Ø AFFICHE [ TOUS LES ] ...
... ENTIERS SONT DES DIVISEURS DE Ø ] STOP
DIV 1 :NOMBRE
AFFICHE [ C'EST TOUT ]
FIN
```
avec
```
POUR DIV :ESSAI :NOMBRE
SI :ESSAI > :NOMBRE STOP
SI RESTE :NOMBRE :ESSAI = Ø (AFFICHE ...
... :ESSAI [ EST UN DIVISEUR DE ] :NOMBRE )
DIV 1 + :ESSAI :NOMBRE
FIN
```

6. On garde pour simplifier Ø pour ''PILE'' et 1 pour ''FACE''. La procédure dépend d'un paramètre qui est le nombre de tirages

```
POUR PILE.OU.FACE :N
REPETE :N [ AFFICHE HASARD 2 ]
FIN
```

On pourra raffiner le programme :
- en faisant écrire PILE ou FACE au lieu de Ø et 1
- en ménageant des interruptions si :N est supérieur à 2Ø : utiliser LC ou LC?, (voir chapitre 9),
ou utiliser AFR suivi de AFR CAR 32, pour avoir un affichage en ligne (voir chapitre 9)

7. Au lieu de HASARD 2, il faut utiliser 1 + HASARD 6, qui sort un nombre entre 1 et 6.

CHAPITRE XIII

PISTES POUR LES PROJETS

• *Où l'on retrouve du texte*

1. Si le mot a un seul caractère, c'est tout fait, sinon, il faut mettre le premier caractère à la fin, c'est-à-dire former un mot avec l'INVERSE du SAUF-PREMIER suivi du PREMIER caractère.

```
POUR RENVERSE :M
SI NON MOT? :M AFFICHE ...
        ... [ IL ME FALLAIT UN MOT ] STOP
SI :M = " AFFICHE ...
    ... [ L'INVERSE DU VIDE EST VIDE ] STOP
AFFICHE RENV :M
FIN
```

avec

```
POUR RENV :M
SI :M = PREMIER :M SORS :M
SORS MOT RENV SAUFPREMIER :M PREMIER :M
FIN
```

2. Il s'agit d'une adaptation de la procédure précédente.
3. Il faut combiner les procédures des deux exercices précédents
4. On va d'abord voir comment sortir le n-ième élément d'une liste.

Nous ferons appel à la procédure

```
POUR COMPTE.LISTE :L
SI :L = [ ] ALORS SORS Ø SINON SORS ...
    ... 1 + COMPTE.LISTE SAUFPREMIER :L
FIN
```

Voici la procédure de niveau Ø

```
POUR TIRE :N :L
SI :N = Ø AFFICHE ...
                ... [ JE NE SAIS PAS FAIRE ] STOP
SI :L = [ ] AFFICHE [ JE NE PEUX RIEN TIRER ] ...
                ... D'UNE LISTE VIDE ] STOP
CREE "P COMPTE.LISTE :L
TESTE :N > :P
SIVRAI (AFFICHE [ JE NE PEUX PAS ] ...
... SORTIR LE ] :N [ -IEME ELEMENT D'UNE ] ...
... LISTE QUI N'EN COMPTE QUE ] :P ) STOP
SIFAUX AFFICHE FTIRE :N :L
FIN
```

La procédure "qui fait le travail" est

```
POUR FTIRE :N :L
TESTE :N = 1
SIVRAI SORS PREMIER :L
SIFAUX SORS FTIRE :N - 1 SAUFPREMIER :L
FIN
```

Maintenant, avoir une procédure qui sort au hasard n'est pas sorcier : il suffit de remplacer :N par 1 + HASARD COMPTE.LISTE :L

Dans ce cas, il n'y a plus lieu de créer "P, car on est sûr que :N est compris entre 1 et :P ; on ne risque plus non plus d'avoir :N égal à zéro.

- ***Encore la tortue***

1.

```
POUR AXES
LP CENTRE POSEPLUME
AVANCE 24Ø CENTRE
DROITE 9Ø
AVANCE 28Ø CENTRE
FIN
```

Ceci trace seulement les axes

```
POUR GRADX :N
LEVEPLUME CENTRE POSEPLUME
XGRAD ABS :N Ø CENTRE
XGRAG – ABS :N 1
CENTRE
FIN
```

(ABS est une procédure qui sort la valeur absolue d'un nombre)

```
POUR XGRAD :N :P
TESTE :N * :P > 139
SIVRAI STOP
SIFAUX LEVEPLUME FIXEX (:N * :P) ...
        ... POSEPLUME AVANCE 5 RECULE 5
XGRAD :N :P + 1
FIN
```

de même

```
POUR XGRAG :N :P
```

```
TESTE :N * :P < - 139
SIVRAI STOP
SIFAUX LEVEPLUME FIXEX (:N * :P) ...
        ... POSEPLUME AVANCE 5 RECULE 5
XGRAG :N :P + 1
FIN
```

Concevoir alors la graduation sur Y ne devrait pas être trop difficile. Bien sûr, :N est le pas de la graduation exprimé en pas de tortue !

2. Si l'on se donne comme équation :Y = 1ØØ * SIN :X (pour que ce soit un peu visible)

```
POUR SINUSOIDE
LEVEPLUME ...
... FIXEXY - 14Ø 1ØØ * SIN - 14Ø ...
                        ... POSEPLUME
SINUS -139
FIN
```

avec

```
POUR SINUS :A
TESTE :A < 14Ø
SIVRAI FIXEXY :A 1ØØ * SIN :A
SIFAUX STOP
SINUS :A + 1
FIN
```

3. Procédure analogue pour une courbe d'équation :Y = :A * :X * :X + :B * :X + :C
choisir :A, :B, et :C pour avoir un bon cadrage dans l'écran

4. idem avec
:Y = (SIN :X) / COS :X
n'oubliez pas les parenthèses !
5. La brique de base est, en notant :N le nombre d'élèves ayant la note donnée

```
POUR BARRE :N
AVANCE 1Ø * :N
DROITE 9Ø AVANCE 2Ø GAUCHE 9Ø
RECULE 1Ø * :N
FIN
```

1Ø * :N a pour but de rendre le diagramme plus lisible, même chose pour la largeur de 2Ø donnée aux barres. Il conviendra de positionner le départ du diagramme, par exemple avec

```
LEVEPLUME FIXEXY – 1ØØ Ø POSEPLUME
```

6. On traduira ces pourcentages en degrés (100 % correspondant à 360 degrés).

Ce manuel d'apprentissage vous a permis d'essayer et de voir fonctionner sur des exemples la plus grande partie des primitifs d'EDI-LOGO (version M.I.T.). Vous trouverez la liste complète de ces primitifs dans le chapitre 3 du manuel de référence, où ils sont décrits et accompagnés d'un exemple succinct d'utilisation. Le manuel de référence contient également la liste complète des messages d'erreur.
C'est maintenant à vous d'aller à la découverte des possibilités du langage LOGO.

```
BIENVENUE A EDI-LOGO
```

Liste et Index des primitifs EDI-LOGO avec les abréviations

Liste et Index des primitifs EDI-LOGO avec les abréviations

Liste et Index des primitifs EDI-LOGO avec les abréviations

R : consulter le manuel de référence

LE CLAVIER DE L'APPLE IIe

Le clavier de l'Apple IIe répond à deux normes (AZERTY et QWERTY) : on passe de l'une à l'autre grâce à un interrupteur situé sous le clavier, en avant à droite de la console. Vous devez utiliser la position QWERTY, qui offre le gros avantage d'avoir les chiffres sans *utiliser la touche* [SHIFT] .

Vérifiez que la touche [⇩] *, située en bas à gauche du clavier, est enfoncée, sinon les messages que vous enverrez à l'ordinateur lui seront incompréhensibles.*

La rangée supérieure de touches, appelée aussi rangée des chiffres, comporte 14 touches : les chiffres, le signe – et le signe = . La touche la plus à droite, [DEL] *, n'est pas utilisée par LOGO. La touche la plus à gauche* [ESC] *permet d'effacer le caractère situé immédiatement à gauche du curseur.*

La deuxième rangée est la rangée des voyelles. Elle est complétée par une touche [→|] *qui n'est pas utilisée par LOGO, et à droite par les deux touches de crochets [et] . Attention, ne pas appuyer sur la touche "majuscules" en même temps que sur les touches de crochets.*

Les touches sont toutes auto-répétitives : si vous maintenez une touche enfoncée, il se produit une répétition automatique du caractère correspondant.

La troisième rangée comporte les touches ; et ' . Les flèches [←] *et* [→] *sont situées en bas à droite de la barre d'espacement ; elles sont également auto-répétitives. LOGO n'utilise pas les touches* [↑] *et* [↓]

Les touches majuscules (SHIFT) sont symbolisées par [⇧] *sur le clavier de l'Apple IIe. Elles sont situées sur la rangée inférieure du clavier, à gauche et à droite.*

[⇧] [+ =] *affiche le caractère + (rangée des chiffres)*

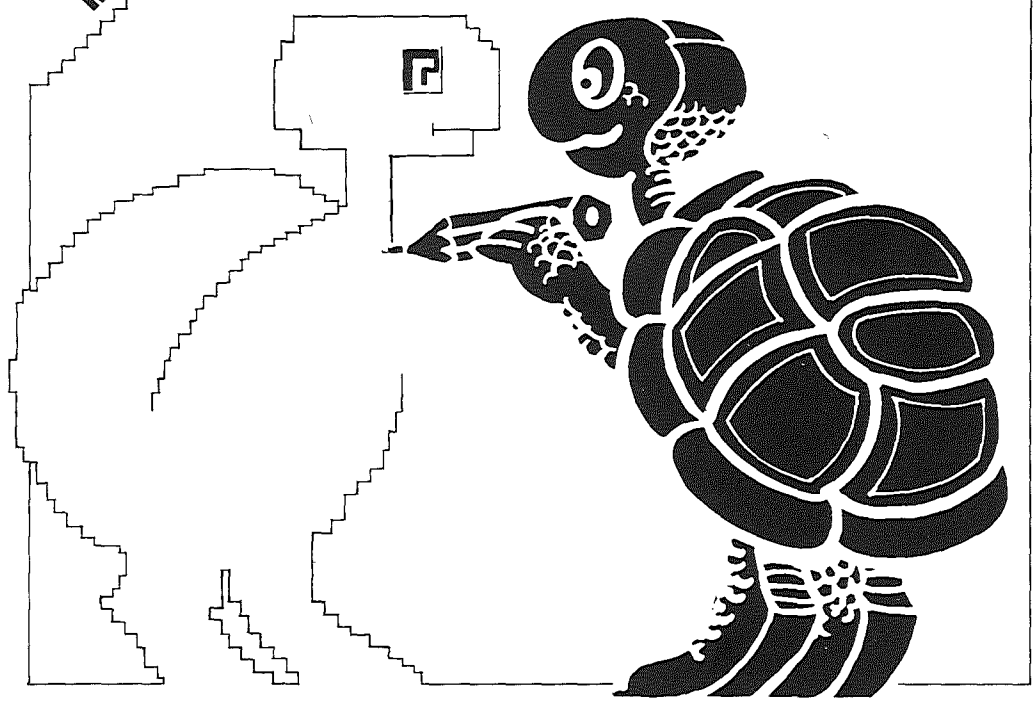
[⇧] [(9] *affiche la parenthèse ouvrante (*

[⇧] [) ø] *affiche la parenthèse fermante)*

La touche de fin de message (RETURN) est une touche de forme bizarre, située à droite du clavier, et portant une flèche [↵]

MANUEL DE REFERENCE EDI-LOGO

EDICIEL
MATRA ET HACHETTE



AVERTISSEMENT

Ce manuel de référence de la version EDI-LOGO pour micro-ordinateur Apple II ou Apple IIe n'est pas un manuel d'apprentissage, et suppose que le lecteur est déjà familier avec le langage et l'environnement LOGO. Les connaissances essentielles pour programmer en LOGO sont présentées dans le manuel ''Premiers pas avec EDI-LOGO'', qui est également fourni avec la disquette du langage EDI-LOGO.

Ce manuel de référence reprend très largement le document ''Reference Manual'' rédigé par le groupe LOGO de M.I.T. pour la version ''M.I.T. LOGO for the Apple II'', sous la direction du Professeur Harold Abelson.

SOMMAIRE

Chapitre VI Divers

CHAPITRE I

AVANT D'UTILISER EDI-LOGO

Ce chapitre présente la configuration requise pour utiliser LOGO, ainsi que quelques utilisations du clavier propres au système LOGO, et indique comment charger LOGO.

• *Configuration requise*

Cette version de LOGO nécessite un micro-ordinateur Apple II ou Apple II "plus", muni d'un lecteur-enregistreur de disquette, de 48 K de mémoire et d'une extension mémoire de 16 K (cette carte doit être enfichée sur la fente 0 dans votre Apple).
La nouvelle configuration Apple IIe qui comporte 64 K de mémoire sans cette extension convient également.
Le logiciel comporte trois disquettes DOS 3.3: celle portant le langage LOGO, vous est fournie en deux exemplaires, vous assurant ainsi une version de secours s'il vous arrivait d'endommager l'une de ces deux disquettes ; l'autre comporte des utilitaires LOGO, qui sont décrits page 152. En plus de ces deux disquettes, langage et utilitaires, vous aurez besoin de disquettes où sauvegarder les programmes que vous aurez conçus. La marche à suivre pour initialiser de telles disquettes-fichiers est expliquée page 205.

• *Le clavier (Apple II ; pour Apple IIe voir page 185)*

LOGO utilise quelques touches du clavier Apple d'une manière spécifique, différente en particulier de leur utilisation avec d'autres langages comme BASIC ou PASCAL.

La touche SHIFT *(majuscules)*

Comme dans la plupart des systèmes Apple, LOGO n'utilise que les lettres majuscules, aussi la touche de majuscules ne sert-elle que pour les touches comportant deux symboles : ainsi 9 affiche un 9 à l'écran, alors que SHIFT 9 (maintenir la touche SHIFT pendant que vous appuyez sur la touche 9) affiche une parenthèse fermante.

Crochets (sur Apple IIe : touches spécifiques)

LOGO utilise les caractères crochet ouvrant [et crochet fermant] qui ne sont pas marqués à l'origine sur le clavier de l'Apple. On les obtient en LOGO en tapant SHIFT **N** et SHIFT **M** respectivement. Vous disposez de petits autocollants à poser sur ces deux touches pour vous aider à les retrouver plus facilement.

La touche de contrôle CTRL

La touche de contrôle, marquée CTRL , est située à gauche du clavier ; elle s'utilise d'une manière similaire à la touche SHIFT : tout en maintenant cette touche CTRL appuyée, vous appuyez sur certaines autres touches.
Par exemple CTRL **G**.

CHAPITRE I

AVANT D'UTILISER EDI-LOGO

Les touches fléchées

Les deux touches fléchées [←] et [→] , situées à droite du clavier, servent à déplacer le curseur, vers la gauche ou vers la droite respectivement, sur la ligne de texte en cours d'affichage.

La touche [ESC]

La touche [ESC] (de l'anglais ESCAPE , s'échapper) est située en haut à gauche du clavier (au dessus de la touche [CTRL] . Elle permet de détruire le caractère situé *immédiatement à gauche* du curseur : dernier caractère frappé, ou caractère situé à gauche du carré clignotant si vous avez déplacé celui-ci à l'aide des touches [←] ou [→] .

La touche [REPT] *(Apple IIe : touches auto-répétitives)*

La touche de répétition [REPT] est située à droite du clavier, au dessus des touches fléchées. Si, tout en maintenant enfoncée la touche [REPT] vous appuyez sur une quelconque touche de caractère, ce caractère se trouve affiché à l'écran et rapidement réaffiché en plusieurs exemplaires, tant que la touche [REPT] est pressée. La répétition fonctionne de la même manière avec la barre d'espacement, avec la touche [ESC] d'effacement, ou avec les touches [←] et [→] de déplacement du curseur.
Cette touche est en particulier utile quand vous êtes en mode éditeur, en combinaison avec les touches fléchées.

La touche [RESET] *(sans risque sur Apple IIe)*

Quand vous utilisez LOGO, surtout *n'appuyez jamais* sur la touche [RESET] . Malheureusement le mécanisme du RESET sur Apple a été conçu d'une manière qui le rend incompatible avec des langages de programmation interactifs complexes comme LOGO, et le fait d'appuyer sur [RESET] provoque la sortie du système LOGO. A force, les utilisateurs de micro-ordinateurs Apple ont imaginé de nombreuses astuces pour s'éviter les ennuis provoqués par la frappe involontaire de la touche [RESET] . Les solutions sont variées : enlever le petit bloc de matière plastique qui constitue la touche, coincer cette touche en position haute, bricoler un couvercle de carton ou de matière plastique pour protéger cette touche.

Quelle que soit la solution que vous retenez, gardez présent à l'esprit que si vous appuyez sur la touche [RESET] pendant que vous utilisez LOGO, cela vous fera perdre tout votre travail et vous obligera à recharger le système en suivant le processus décrit ci-dessous.

- ***Charger LOGO***

Le chargement du langage s'effectue de manières légèrement différentes selon le type d'Apple que vous avez.

Si votre Apple dispose d'une ROM (mémoire morte) "auto-chargeante"–c'est le cas des Apple II "plus" et de l'Apple IIe–, insérez la disquette dans le lecteur alors que votre micro-ordinateur n'est pas encore allumé. Mettez alors sous tension le moniteur puis la console.

Vous verrez apparaître successivement sur l'écran une page de points d'interrogation, puis la marque Apple II, ensuite un caractère] et enfin le message

UN MOMENT...MERCI

Après environ trente secondes, LOGO est prêt et vous en avertit par un message d'accueil

BIENVENUE A EDI-LOGO

A la ligne suivante s'affichent un point d'interrogation et un carré qui clignote (le "curseur") , qui vous indiquent que le système est prêt à recevoir vos ordres.

Retirez alors la disquette langage du lecteur de disques, et rangez-la dans un endroit sûr, à l'abri de la poussière et du magnétisme.

Si votre Apple n'a pas de ROM "autochargeante", insérez la disquette langage dans le lecteur alors que votre micro-ordinateur est éteint, puis allumez le moniteur et la console. Tapez alors sur la touche [RETURN] , puis sur la touche **6** , ensuite sur [CTRL] **P** (maintenez la touche [CTRL] enfoncée pendant que vous appuyez sur la touche **P**) et enfin sur la touche [RETURN] à nouveau.

Comme dans le cas d'un Apple II avec ROM "autochargeante", vous verrez s'afficher sur l'écran APPLE II , et, pour vous faire patienter –et vous indiquer que tout se passe bien – pendant que le langage est en train de se charger

UN MOMENT ... MERCI

Lorsque le chargement du langage est terminé, au bout d'une demi-minute environ, vous en serez prévenu par un message d'accueil

BIENVENUE A EDI-LOGO

et par l'affichage, à la ligne suivante, d'un point d'interrogation suivi d'un carré qui clignote (le "curseur") : le système est prêt à recevoir vos ordres. Par précaution, retirez la disquette du lecteur, et rangez-la dans un endroit sûr.

CHAPITRE I

AVANT D'UTILISER EDI-LOGO

- ***Problèmes avec le système LOGO***

A notre connaissance, LOGO est le programme le plus complexe et le plus important jamais écrit pour l'Apple II ; la version actuelle contient encore quelques *bugs* qui peuvent provoquer une sortie du système. Le symptôme le plus clair de ce genre de problème est que l'ordinateur cesse de travailler en LOGO et vous affiche le message :

TIENS! VOICI UN PEPIN IMPREVU

Dans pratiquement tous les cas, ceci signifie que l'espace mémoire disponible est insuffisant. Il est alors possible de retrouver LOGO en tapant [CTRL] **Y** puis [RETURN] . Ceci étant fait, commencez par essayer de sauver le contenu de votre espace de travail dans un fichier provisoire sur une disquette, puis redonnez-vous un espace de travail vierge en tapant AUREVOIR , et enfin ramenez vos procédures du fichier où vous venez de les stocker.

Dans quelques cas particulièrement complexes, cette méthode à l'aide de [CTRL] **Y** ne fonctionne pas, le contenu de l'espace de travail est alors irrémédiablement perdu, et il est indispensable de charger à nouveau LOGO comme indiqué précédemment.

Rassurez-vous, il est très rare de rencontrer de tels problèmes !

Si, quoi que vous tapiez, LOGO vous répond que ce n'est pas défini, c'est que vous êtes en mode minuscules sur un Apple IIe. Repassez en majuscules.

CHAPITRE II

UTILISATION DU SYSTEME LOGO

Le système LOGO comprend un interpréteur complet du langage LOGO, un éditeur de texte pour l'édition des procédures que vous définissez, et un système de "graphisme tortue" intégré. Ce chapitre fournit des indications sur les interactions entre ces différentes fonctions.

• *Différents modes d'utilisation de l'écran*

Le système LOGO utilise l'écran du moniteur (ou de votre téléviseur) de trois manières –ou modes– différentes.

Mode ECRIS

C'est le mode dans lequel est le système quand le chargement du langage vient de se terminer. LOGO vous informe qu'il est prêt à recevoir vos ordres, en affichant un point d'interrogation, suivi d'un carré qui clignote (et que l'on appelle le "curseur"). Tapez alors vos lignes de commandes, en les terminant par [RETURN] : LOGO exécute les lignes, et, le cas échéant, affiche une réponse.

Chaque fois que le curseur est visible et clignote, cela signifie que LOGO attend que vous tapiez quelque chose. Il ne se produira rien tant que vous n'aurez rien tapé.

Le système comporte un éditeur très souple qui vous permet de corriger les fautes de frappe que vous avez pu faire dans les lignes de commande que vous avez tapées. Les opérations d'édition disponibles en mode "ECRIS" sont :

[ESC] Efface le caractère situé immédiatement à gauche du curseur, et déplace le curseur d'un cran vers la gauche, à l'emplacement du caractère qui vient d'être effacé. Tout texte situé à droite du curseur se trouve alors décalé d'un cran vers la gauche également. Vous trouverez un autocollant **EFF** pour cette touche.

[→] [←] Déplace le curseur d'un cran vers la droite ou vers la gauche respectivement, *sans* effacer aucun caractère.

[CTRL] **A** Déplace le curseur jusqu'au début de la ligne en cours. Le caractère A a été choisi pour ce contrôle car il est situé au début d'une rangée de touches du clavier. De plus, c'est la première lettre de l'alphabet.

[CTRL] **D** *Détruit* le caractère situé sous le curseur, c'est-à-dire le caractère qui apparaît par intermittence à l'emplacement du carré qui clignote. Le reste de la ligne, situé à droite du curseur, se décale d'un cran vers la gauche lorsque le caractère est détruit.

[CTRL] **E** Déplace le curseur jusqu'à la fin de la ligne en cours. Le choix du E provient de l'anglais END (= FIN).

[CTRL] **G** Arrête l'exécution, et affiche un message indiquant cette interruption.

CTRL **K** Détruit tous les caractères de la ligne en cours, situés à la droite du curseur. Le choix de K provient de l'anglais KILL (=TUER).

CTRL **P** Ramène le contenu de la ligne précédente, et le réaffiche à l'écran. Ce contenu peut alors être modifié et/ou réexécuté. Toutefois, dans l'implémentation actuelle, les primitifs DEFINIS, EXECUTE et REPETE ne permettent pas l'utilisation de cette touche de contrôle. Il en est de même pour les primitifs de sauvegarde ou rappel de fichiers sur disquettes.

Attention ! Une "ligne" pour l'ordinateur peut être plus longue qu'une ligne d'affichage à l'écran. Tant que vous n'avez pas appuyé sur RETURN, vous êtes toujours sur la même "ligne". Ceci est en particulier important pour les commandes CTRL **A** , CTRL **E** , CTRL **K** , CTRL **P** .

Mode éditeur.

L'exécution des primitifs POUR et EDITE place LOGO en mode éditeur. Par exemple, si, LOGO étant chargé, vous tapez

POUR POLY :COTE :ANGLE

et appuyez alors sur la touche RETURN , le système entre dans l'éditeur écran, et affiche le texte sur l'écran. Vous êtes informé que vous vous trouvez en mode éditeur par le bandeau en inversion de couleur :

EDITE: CTRL-C POUR DEFINIR, CTRL-G SINON

situé au bas de l'écran.

A partir de maintenant, vous pouvez utiliser toutes les commandes d'édition décrites page 202 pour créer et/ou éditer le texte de la procédure. Si vous tapez CTRL **C**, il y a sortie de l'éditeur avec définition de la procédure, selon le texte que vous avez tapé : on se trouve alors en mode "ECRIS". Si vous tapez CTRL **G**, l'édition est stoppée (un message vous en avertit) : LOGO retourne au mode "ECRIS" sans qu'une nouvelle procédure soit définie.

Autrement dit, lors de la création d'une nouvelle procédure, CTRL **C** permet d'effectivement définir cette procédure, alors que CTRL **G** empêche la définition. Lors de l'édition d'une procédure pré-existante, CTRL **C** permet d'enregistrer une modification apportée à cette procédure (et remplace donc l'ancienne définition de cette procédure par la nouvelle), alors que CTRL **G** permet de changer d'avis, et de ne pas enregistrer les modifications : c'est alors l'ancienne définition de la procédure qui est conservée.

Pour éditer la dernière procédure qui vient d'être définie, il suffit de taper EDITE (en abrégé ED), sans préciser le nom de la procédure en question. Plus exactement, quand on est en mode "DESSINE", la commande EDITE –sans précision d'un nom de procédure– édite la définition actuelle de la procédure la plus récemment éditée ou imprimée. En mode "ECRIS", par contre, la commande EDITE non suivie d'un nom de procédure, remet en mode éditeur avec tout le contenu du tampon d'édition. On pourra ainsi parfois avoir plusieurs procédures affichées dans l'éditeur.

Mode DESSINE

En mode "DESSINE", on utilise la tortue pour dessiner sur l'écran. Si vous utilisez une commande de tortue alors que vous êtes en mode "ECRIS", le système se mettra automatiquement en mode "DESSINE" avant d'exécuter la commande. Le primitif ECRIS (en abrégé EC) fait sortir du mode "DESSINE", et revenir au mode "ECRIS". Egalement, lorsqu'on sort du mode EDITEUR, fût-ce après l'avoir appelé en mode "DESSINE", on se retrouve *toujours* en mode "ECRIS" et non en mode "DESSINE".

En fait, il y a plusieurs types de modes "DESSINE". Le mode ECRANMIXTE est la manière normale d'utiliser le mode "DESSINE". Au bas de l'écran, quatre lignes sont réservées pour le texte, et le reste de l'écran montre l'espace dans lequel se déplace la tortue. En réalité, l'espace dans lequel la tortue peut évoluer s'étend jusqu'en bas de l'écran, et est donc masqué par la région réservée aux quatre lignes de texte.

En mode PLEINECRAN la région réservée au texte disparaît, rendant visible dans sa totalité l'espace dans lequel se déplace la tortue. Toutefois, si le système se trouve amené à émettre un message d'erreur, le mode ECRANMIXTE sera automatiquement restauré, de manière à rendre visible l'affichage de ce message.

Les commandes CTRL **F** et CTRL **S** permettent de passer du mode ECRANMIXTE au mode PLEINECRAN et inversement : ainsi, appuyer sur CTRL **F** quand on est en mode ECRANMIXTE fait passer en mode PLEINECRAN, et taper CTRL **S** fait alors revenir au mode ECRANMIXTE. Il est parfois pratique de pouvoir réaliser un tel passage entre les modes ECRANMIXTE et PLEINECRAN alors qu'on est dans le cours d'une procédure : ceci peut être obtenu grâce aux primitifs ECRANMIXTE et PLEINECRAN.

A l'occasion, il peut se révéler pratique d'être en mesure de voir plus que les quatre lignes de texte, sans pour autant détruire le dessin qui se trouve affiché à l'écran. Ceci s'obtient en tapant CTRL **T** : la figure réalisée par la tortue disparaît alors, et on a la possibilité d'utiliser toute la surface de l'écran pour du texte, exactement comme en mode "ECRIS". Cependant, on est toujours en

mode DESSINE : les commandes de tortue peuvent être exécutées, bien que leur résultat reste invisible. Pour rendre visible à nouveau l'écran graphique, il convient d'utiliser soit CTRL **F** –qui ramène la totalité de l'écran graphique,–soit CTRL **S** qui remet en mode ECRANMIXTE.

En mode DESSINE , les commandes de contrôle valides (outre les commandes d'édition de lignes décrites dans le mode ECRIS) sont les suivantes :

CTRL **F** Rend visible la totalité de la page graphique.

CTRL **G** Arrête l'exécution, et affiche un message indiquant cette interruption.

CTRL **S** Met la page graphique en mode ECRANMIXTE, les quatre lignes du bas de l'écran étant réservées au texte.

CTRL **T** Rend invisible la page graphique –*sans* la détruire– autorisant ainsi l'utilisation de toute la surface de l'écran pour l'affichage du texte.

En mode DESSINE , comme en mode ECRIS , on pourra ralentir l'exécution d'une procédure de la manière suivante :

CTRL **W** Suspend l'exécution. En tapant plusieurs fois CTRL **W** , on obtient une exécution de la procédure ligne par ligne. On pourra par exemple utiliser simultanément CTRL **W** et REPT pour donner l'impression du ralenti. Taper n'importe quel caractère (excepté CTRL **W** et CTRL **G**) , permet de reprendre l'exécution normale de la procédure.

CTRL **Z** Au cours de l'exécution d'une procédure, produit une pause : vous pouvez alors taper n'importe quelle instruction, qui sera exécutée comme s'il s'agissait d'une quelconque ligne de la procédure en cours. En tapant **CONTINUE** (en abrégé **CO**) suivi de la touche RETURN , la suite de la procédure sera exécutée normalement.

- ***L'édition***

Le système LOGO comporte un éditeur pleine page et un éditeur de ligne compatibles. L'éditeur d'écran est utilisé pour définir des procédures LOGO, et l'éditeur de ligne pour taper des commandes LOGO pour exécution immédiate.
Pendant que vous êtes en train de taper une ligne de caractères en LOGO, vous pouvez ne tenir aucun compte de l'éditeur de ligne tant que vous n'avez pas besoin de vous en servir. Si vous avez fait une faute de frappe, vous pouvez effacer le caractère erronné grâce à la touche ESC . Si vous avez oublié de mettre un mot au

début de la ligne, vous déplacez le curseur jusque là grâce à [CTRL] **A** , et vous pouvez alors taper le mot manquant : les caractères nouveaux provoqueront un déplacement vers la droite des caractères déjà tapés, il ne se perdra rien et il n'y a aucun risque de chevauchement. Si vous voulez insérer des caractères quelque part sur la ligne, déplacez tout simplement le curseur jusqu'à l'endroit voulu en utilisant les touches fléchées [→] et [←] puis tapez les caractères manquants. Pour aller à la fin de la ligne, tapez [CTRL] **E**. Pour effacer un caractère situé à l'emplacement du curseur, utilisez [CTRL] **D**. Pour détruire les caractères depuis le curseur jusqu'à la fin de la ligne, tapez [CTRL] **K**. Pour indiquer à l'ordinateur la "fin de message" et valider la ligne pour que LOGO l'exécute, tapez [RETURN]. Il n'est nullement besoin que le curseur soit situé en fin de ligne : tous les caractères figurant sur la ligne seront pris en compte par LOGO.

Les lignes que vous tapez peuvent dépasser les quarante caractères que comporte une ligne d'affichage à l'écran. Arrivé à la droite de l'écran, il y a passage automatique de l'affichage à la ligne suivante. Cependant, tant que vous n'avez pas appuyé sur la touche [RETURN], LOGO considère qu'il s'agit toujours de la même ligne. C'est sur la "ligne ordinateur" et non sur la "ligne d'affichage à l'écran" que portent les commandes [CTRL] **A** , [CTRL] **E** , [CTRL] **K** .

LOGO conserve la ligne qui vient d'être tapée et validée, que ce soit en mode ECRIS ou en mode DESSINE : vous pouvez rappeler cette ligne précédente et l'insérer dans la ligne en cours en tapant [CTRL] **P** . Notez toutefois que dans l'implémentation actuelle, les primitifs DEFINIS , EXECUTE , et REPETE provoquent l'effacement de la ligne précédente, ainsi que les commandes de sauvegarde ou de rappel de fichiers sur disquettes.

Pour la définition de procédures, LOGO dispose d'un éditeur pleine page, où l'on entre en tapant **EDITE** (en abrégé **ED)** ou encore **POUR** suivi du nom de la procédure que l'on désire définir. Une fois en mode éditeur, les caractères que l'on frappe apparaissent sur l'écran. Mais en appuyant sur [RETURN] , le curseur se positionne au début de la ligne d'affichage suivante, entraînant avec lui toute la partie de texte éventuellement située à sa droite, et il n'y a *pas* d'exécution immédiate de la ligne.

De nombreuses commandes d'édition sont disponibles en mode édition : elles sont décrites ci-dessous. Parmi les plus fréquemment utilisées, citons [CTRL] **N** , qui fait passer à la ligne suivante, et [CTRL] **P** qui permet de retourner à la ligne précédente.

Les lignes peuvent avoir n'importe quelle longueur, pourvu qu'elles entrent dans le tampon d'édition. Les lignes qui comportent plus de quarante caractères sont affichées sur plusieurs lignes de l'écran. De plus, un caractère ! à la droite de chaque ligne de l'écran vous prévient qu'il s'agit d'une seule et même "ligne ordinateur" répartie sur plusieurs "lignes d'affichage écran". Ce caractère ne fait bien

sûr pas partie de la procédure en cours de définition, et n'intervient en aucune manière lors de l'exécution. Remarquez que c'est seulement en mode édition que se produit l'affichage du point d'exclamation, et que ceci n'est pas vrai en mode ECRIS ni en mode DESSINE.

Bien qu'en LOGO les procédures aient rarement plus de quelques lignes de long, vous pouvez éditer un texte sans vous restreindre à ne pas dépasser une page. Si le texte que vous êtes en train d'entrer commence à déborder une page d'écran, le système décale automatiquement l'affichage de manière à amener la ligne en cours approximativement au milieu de l'écran. Si vous tapez CTRL **P** ou CTRL **N** et arrivez en haut ou en bas de l'écran, la page précédente ou suivante (respectivement) s'affichera aussitôt. La commande CTRL **F** , en mode éditeur, déplace instantanément à à la page suivante. Pour aller à la page précédente, taper CTRL **B** (ces lettres proviennent de l'anglais FORWARD et BACK, qui signifient respectivement en avant et en arrière ; dans la mesure où tous ces contrôles sont utilisés par d'autres éditeurs que l'éditeur LOGO, il n'a pas paru opportun d'en modifier les caractères). Si l'on se trouve à la première page, CTRL **B** a pour seul effet de positionner le curseur au début de cette page ; d'une manière analogue, si on se trouve à la dernière page, CTRL **F** positionne simplement le curseur à la fin de cette page.

Pour quitter l'éditeur, et faire effectivement définir la procédure que vous venez de taper, utilisez CTRL **C**. Pour quitter l'éditeur sans faire prendre en compte ce que vous venez de taper, utilisez CTRL **G**. Lorsque vous sortez de l'éditeur avec CTRL **G** et changez d'avis, vous pouvez retrouver votre texte intact en tapant seulement **EDITE** (ou **ED** en abrégé) , à condition qu'entre temps vous n'ayez pas utilisé la tortue ou les opérations sur fichiers.

Ainsi :

```
POUR CARRE
REPETE 4 [AVANCE 40 DROITE 90]
FIN
```

CTRL **C**

```
CARRE EST DEFINI
```

Quelles que soient les commandes utilisées entre temps, en tapant EDITE CARRE vous retrouverez ce texte.

CTRL **G**

CARRE n'est pas défini
Si vous tapez

```
CARRE
```

vous obtenez un message

CARRE N'A PAS ETE DEFINI

Si vous tapez **IMTS**

Le titre CARRE n'apparaît pas. Cependant, si vous tapez

EDITE

vous voyez réapparaître le texte de la procédure (et vous pouvez alors le rendre définitif en tapant CTRL **C**)

Ayant défini (par **CTRL** **C**) cette procédure, vous êtes d'avis de la modifier.

```
EDITE CARRE
POUR CARRE
REPETE 4 [AVANCE 8Ø DROITE 9Ø]
FIN
```

CTRL **C**
La seule procédure CARRE maintenant en mémoire est un carré de 80 pas de côté. C'est ce texte que vous retrouverez en tapant EDITE.

CTRL **G**
Vous avez toujours la précédente procédure CARRE –considérée comme seule définitive – mais aussi, potentiellement, la procédure de carré à 80 pas de côté, que vous pouvez récupérer en tapant EDITE (attention ! *pas* EDITE CARRE qui vous redonnerait le carré de 40 pas). Cependant, si vous ne rendez pas définitive la nouvelle procédure de carré, (en éditant et terminant par un CTRL **C** , au besoin après avoir introduit des modifications supplémentaires, comme un changement dans le titre), cette potentialité sera irrémédiablement perdue après une utilisation de l'écran graphique ou des fichiers sur disquette.

CHAPITRE II

UTILISATION DU SYSTEME LOGO

Le texte que vous tapez dans l'éditeur n'est pas obligatoirement une procédure. Vous pouvez utiliser n'importe quelle commande LOGO, excepté les commandes de graphisme et celles d'opérations sur fichiers (voir page 265).

Pour vous faciliter l'existence, quand vous utilisez l'éditeur, vous avez à votre disposition les commandes suivantes :

[ESC] Efface le caractère situé immédiatement à gauche du curseur, et décale d'un cran vers la gauche le curseur et tout le texte de la ligne en cours situé à droite du curseur.

[→] [←] Déplace le curseur vers la droite ou vers la gauche, sans effacer aucun caractère.

[CTRL] **A** Positionne le curseur au début de la ligne en cours.

[CTRL] **B** Fait apparaître à l'écran la page précédente, ou positionne le curseur au début de la page en cours si celle-ci est la première page.

[CTRL] **C** Sort de l'éditeur, en considérant comme acquis ce qui vient d'y être réalisé.

[CTRL] **D** *Détruit* le caractère situé sous le curseur, c'est-à-dire le caractère qui apparaît par intermittence sous le carré qui clignote.

[CTRL] **E** Positionne le curseur à la fin de la ligne en cours.

[CTRL] **F** Fait apparaître à l'écran la page suivante, ou positionne le curseur à la fin de la page en cours, si celle-ci est la dernière page.

[CTRL] **G** Sort de l'éditeur, sans considérer comme définitivement acquis ce qui vient d'y être réalisé.

[CTRL] **K** Détruit tous les caractères de la ligne en cours situés à droite du curseur.

[CTRL] **N** Positionne le curseur à la ligne suivante.

[CTRL] **O** *Ouvre* une nouvelle ligne à l'endroit où se trouve le curseur. C'est l'équivalent de [RETURN] suivi de [CTRL] **P** ,très utile quand on veut ajouter de nouvelles lignes dans une procédure.

[CTRL] **P** Positionne le curseur à la ligne *précédente.*

• *Contrôle des couleurs*

Si vous avez un moniteur ou un téléviseur couleur (et la carte d'extension correspondante), vous pouvez utiliser le primitif COLOREPLUME (en abrégé CP) pour changer la couleur des lignes tracées par la tortue. Vous pouvez aussi utiliser le primitif COLOREFOND (en abrégé CF) pour déplacer la tortue sur des fonds de couleurs variées. Ces deux primitifs nécessitent un argument d'entrée, qui est un nombre de 0 à 6. La correspondance est la suivante :

nombre	*couleur*
0	noir
1	blanc
2	vert
3	mauve
4	orange
5	bleu

La couleur qui apparaît effectivement sur l'écran peut varier d'une manière très importante suivant la manière dont le téléviseur est réglé. Si vous avez un téléviseur noir et blanc les "couleurs" apparaîtront comme différentes rayures.
COLOREPLUME 6 "inverse" la couleur de tout point sur lequel passe la tortue. La couleur effectivement produite dépend à la fois de la couleur du fond et de la couleur du point ; de toute manière, si l'on inverse la couleur d'un point et qu'on l'inverse à nouveau, on retrouve toujours la couleur initiale.
Si vous ne précisez pas une commande COLOREFOND et COLOREPLUME, les valeurs prises par défaut sont COLOREFOND 0 (noir) et COLOREPLUME 1 (blanc).

Dessin sur fond de couleur.

Quand on dessine sur un fond coloré (valeurs de 2 à 5 pour COLOREFOND), on ne peut utiliser que deux des quatre couleurs simultanément. Ainsi, sur un fond vert ou mauve, on n'aura jamais de bleu ou d'orange : COLOREPLUME 4 tracera en vert (et non orange), et COLOREPLUME 5 en mauve (au lieu de bleu). Il convient également de signaler que si vous tracez un dessin en couleur sur l'écran, puis changez la couleur du fond, la couleur des lignes du dessin peut se trouver modifiée, ou les lignes déformées d'une manière inattendue. Cependant, si vous revenez à la couleur initiale du fond, vous retrouverez l'apparence première du tracé.
Ces effets bizarres sont dus au système couleur sur l'Apple II, qui ne permet pas, par exemple, d'avoir des points verts très proches de points oranges.

Dessin sans utilisation de la couleur.

Pour obtenir des couleurs nettes avec l'Apple, le système LOGO a recours à des traits plus épais qu'il n'est en fait nécessaire. Cela signifie que les dessins n'auront pas l'air aussi précis que si l'on pouvait tracer des lignes plus fines. Si vous n'avez pas besoin de la couleur, les résultats seront plus agréables avec des traits plus fins. Pour cela, commencez par COLOREFOND 6. Sur un fond de "couleur" 6 , qui apparaît noir, COLOREPLUME 0 trace en noir (c'est-à-dire en "lèveplume"), les nombres de 1 à 5 correspondant à un trait "blanc" et 6 assure l'inversion. Nous indiquons "blanc" entre guillemets, car sur un écran couleur, le "blanc" n'est pas toujours blanc. En particulier, des lignes verticales "blanches" sont soit rouges soit vertes suivant leur position !

CHAPITRE II

UTILISATION DU SYSTEME LOGO

- ***Le système de fichiers en LOGO***

En LOGO, le système de fichiers vous permet de sauvegarder les définitions des procédures sur une disquette. Un utilisateur peut avoir plusieurs fichiers sur une même disquette, chacun de ces fichiers ayant un nom. Les noms des fichiers sont répertoriés dans le CATALOGUE de la disquette.

Fichiers sur disquettes.

Quand vous utilisez LOGO, vous devriez normalement avoir une disquette de travail dans le lecteur-enregistreur de disques. Pour préparer une disquette de travail, reportez-vous à la page ci-contre.

Si vous voulez sauvegarder les définitions de vos procédures, utilisez le primitif GARDE. Par exemple

GARDE "MONBOULOT

sauvegarde les définitions des procédures et des noms actuellement disponibles dans l'espace de travail, en en stockant une copie dans un fichier du nom de MONBOULOT. Cela ne sauve pas uniquement la procédure dont le titre est MONBOULOT (d'ailleurs, vous n'avez probablement aucune procédure avec ce titre !).

Attention ! Si vous aviez déjà un fichier du nom de MONBOULOT, l'ancien est détruit et remplacé par un nouveau. Par conséquent, si un fichier MONBOULOT contient des procédures que vous n'avez pas en ce moment dans l'espace de travail et dont vous voulez garder trace, changez de nom pour sauvegarder l'actuel espace de travail. Par exemple, MONBOULOT1.

Le primitif RAMENE a besoin d'un nom de fichier comme argument d'entrée. Il lit le fichier de ce nom sur la disquette, et transfère en mémoire une copie des informations contenues dans ce fichier.

Le système de fichiers, en LOGO, fait appel à la même zone de mémoire que le tracé de dessins ou l'édition de procédures. Par conséquent, si vous utilisez une commande concernant les fichiers pendant que vous êtes en mode DESSINE, vous vous retrouvez en mode ECRIS.

Remarque : le nom d'un fichier utilisé avec GARDE ou RAMENE est toujours précédé du caractère *guillemets.*

Le primitif CATALOGUE affiche à l'écran la liste de tous les fichiers de la disquette. Les fichiers correspondant à la sauvegarde de l'espace de travail sont affichés avec la mention .LOGO. Ainsi le fichier créé par la commande

GARDE "MONBOULOT

sera affiché comme MONBOULOT.LOGO. N'indiquez jamais la mention .LOGO quand vous utilisez les primitifs GARDE ou RAMENE. Pour effacer un fichier de la disquette, utilisez le primitif DETRUIS-FICHIER, suivi du nom du fichier à effacer (n'oubliez pas le caractère *guillemets* au début).

Sauvegarde de dessins.

Outre la sauvegarde des définitions de procédures, LOGO vous permet de garder sur disquettes une image graphique, vous offrant ainsi la possibilité de l'afficher à nouveau à l'écran. Pour cela, on utilise les primitifs GARDEDESSIN et RAMENEDESSIN. GARDE-DESSIN, analogue à GARDE, doit être suivi d'un nom, qui désigne le fichier de l'image. L'image qui est actuellement affichée à l'écran est alors copiée sur la disquette. Bien entendu, cette commande ne s'utilise que lorsqu'on est en mode graphique ! RAMENEDESSIN, suivi du nom de fichier de l'image, reconstitue une copie du dessin sur l'écran. Il est conseillé de cacher la tortue avant de taper GARDE-DESSIN ; sinon, quand vous ramènerez le dessin, vous risquez de vous trouver avec deux tortues : une qui est l'image de la tortue sur le dessin, et l'autre qui est la vraie tortue. Les fichiers de dessin sont reconnaissables sur le catalogue grâce à la mention DESSIN. Là encore, n'indiquez jamais la mention .DESSIN quand vous utilisez les primitifs GARDEDESSIN et RAMENEDESSIN.

Pour effacer un fichier d'image tapez DETRUISDESSIN suivi du nom du fichier d'image à détruire (n'oubliez pas le caractère *guillemets* avant le nom du fichier.)

Préparation d'une disquette de travail.

La sauvegarde des fichiers LOGO s'effectue sur des disquettes standard après une initialisation adéquate. Voici comment procéder :

1 . Insérez la disquette utilitaire dans le lecteur-enregistreur de disquette de votre micro-ordinateur, et allumez la console. (La disquette utilitaire n'est *pas* celle que vous utilisez pour charger LOGO, mais celle qui contient les programmes de démonstration).

2 . Tapez

LOAD HELLO RETURN

Attendez que le lecteur ait fini de tourner (la lampe témoin rouge s'éteint lorsque c'est terminé).

3 . *Retirez* la disquette utilitaire du lecteur, et insérez une disquette *vierge.* Attention : cette disquette va être initialisée (ou ré-initialisée), et toute information préalablement stockée sur cette disquette sera alors *irrémédiablement perdue.*

4 . Tapez

INIT HELLO RETURN

Attendez que le lecteur ait fini de tourner : cela prend environ une minute (là encore, la lampe témoin, en s'éteignant, vous avertit que c'est terminé). La disquette est alors initialisée et peut être utilisée pour stocker des fichiers LOGO.

• *Utilisation de périphériques*

Les opérations usuelles en LOGO font appel au clavier, à l'écran, et à un lecteur-enregistreur de disquettes. Il y a aussi deux primitifs (MANETTE et BOUTON? voir au chapitre 3 la description de ces primitifs) qui permettent de recevoir de l'information d'une à quatre manettes de jeu qui peuvent être reliées à l'Apple.

En plus, LOGO dispose d'un primitif PERIPH qui permet d'envoyer de l'information à des dispositifs divers. Ce primitif doit être suivi d'un nombre qui indique la fente (slot) sur la carte Apple dans laquelle doit être connectée la carte interface correspondante. Le primitif PERIPH a pour effet de transférer sur le dispositif concerné les informations qui seraient normalement affichées *par LOGO* sur l'écran. On revient à l'écran du moniteur grâce à PERIPH Ø. Ce primitif permet également d'appeler une routine définie en langage assembleur par l'utilisateur. Contrairement à ce qui se produit avec la commande PR # du BASIC Apple, le primitif PERIPH ne transfère pas au dispositif périphérique les informations tapées au clavier : ce sont seulement les textes *produits par LOGO* qui arrivent à ce dispositif extérieur. Il n'y a aucune modification de l'éditeur d'écran ni de l'éditeur de ligne.

Ainsi, supposons que vous avez une imprimante connectée à l'Apple sur la fente (slot) 7 et que vous voulez utilisez cette imprimante pour garder une trace écrite d'une procédure CERCLE. Il suffit d'écrire

PERIPH 7

IMPRIME CERCLE

PERIPH Ø

Si votre imprimante, pour fonctionner, doit recevoir certaines commandes préalables, il faut les lui passer avant de charger LOGO.

Par exemple à partir du DOS, tapez (pour une Microline 82 A)

PR# 7 RETURN

CTRL **I N** RETURN

PR# 6 RETURN

pour charger LOGO, puis procédez comme ci-dessus.

Si vous avez une imprimante SILENTYPE vous pouvez obtenir des copies sur papier de dessins d'écran. Pour cela, il vous suffit, une fois que vous avez sur l'écran le dessin à conserver, d'envoyer à l'imprimante un code de contrôle qui est le caractère dont le code ASCII est 17. En admettant que l'imprimante soit branchée sur la fente (slot) 7, voici une procédure LOGO qui résoud votre problème :

```
POUR COPIEDECRAN
PERIPH 7
AFR CAR 17
PERIPH Ø
FIN
```

Le primitif PERIPH transmet uniquement les informations produites après AFFICHE (ou AFR) et les messages d'erreur. Les caractères que vous tapez au clavier continuent de s'afficher à l'écran, mais ne sont pas transmis au dispositif extérieur. Par ailleurs, les commandes d'édition, effaçage, flèches de déplacement, etc... fonctionnent normalement.

Du fait de l'imprimante SILENTYPE, la copie sur papier du dessin n'est pas de très bonne qualité. Si vous voulez interrompre l'exécution, tapez CTRL **C**.

Il convient de signaler que le facteur d'échelle de l'imprimante est souvent différent de celui des moniteurs. Vous aurez donc intérêt à effectuer quelques essais préalables pour déterminer la valeur à donner à .ASPECT (voir la description de ce primitif au chapitre 3) qui donne les meilleurs résultats sur votre matériel.

- ***Pour les utilisateurs d'Apple avec BASIC entier***

La procédure indiquée pour l'initialisation d'une disquette ne convient pas si l'on dispose d'un Apple avec BASIC entier. Dans ce cas, il faut procéder comme suit.

1 . Insérer la disquette utilitaire sans autocollant de protection d'écriture dans le lecteur-enregistreur. Allumer ensuite l'Apple. Sur l'écran s'affiche le message

```
LANGUAGE NOT AVAILABLE
```

et un chevron (">")
Taper alors les lignes suivantes

1Ø PRINT" CTRL **D CATALOG"**

UNLOCK HELLO

DELETE HELLO

SAVE HELLO

LOCK HELLO

2 . *Important :* retirer la disquette utilitaire du lecteur (et remettre en place l'autocollant de protection d'écriture si vous le souhaitez).

3 . Insérer une disquette vierge, puis taper

INIT HELLO

Quand la lampe témoin du lecteur s'éteint, la disquette est initialisée.

CHAPITRE III PRIMITIFS ET MESSAGES D'ERREUR

LOGO est un langage de programmation puissant, qui comporte des commandes graphiques, des opérations sur les nombres, et qui peut traiter des listes. Il dispose également d'un éditeur écran en temps réel, qui peut être utilisé aussi bien pour éditer les lignes d'instructions au fur et à mesure qu'elles sont tapées, que pour éditer les définitions de procédures.

• *Commandes graphiques*

AVANCE en abrégé **AV**

Requiert un nombre comme argument d'entrée. Fait avancer la tortue du nombre de pas indiqué, dans la direction que regarde la tortue. Si la plume est posée, ce déplacement laisse une trace.

Exemple : **AVANCE 100**

La tortue avance de cent pas.

CACHETORTUE en abrégé **CT**

Fait disparaître le triangle qui symbolise la tortue.

CAP Sort sous forme de nombre décimal (compris entre 0 et 359) l'angle que fait la tortue avec le "nord", compté positivement dans le sens des aiguilles d'une montre et mesuré en degrés.

Exemple : **FIXECAP CAP + 10**

est équivalent à DROITE 10

CENTRE Positionne la tortue au centre de l'écran. Si la plume est posée, ce déplacement laisse une trace.

COLOREFOND en abrégé **CF**

Requiert un nombre comme argument d'entrée. Colore le fond de l'écran d'une teinte uniforme. La correspondance nombre-couleur est la suivante :

nombre	couleur
0	noir
1	blanc
2	vert
3	mauve
4	orange
5	bleu
6	noir : permet d'avoir des traits plus fins quand on n'utilise pas la couleur pour les tracés.

Exemple :

COLOREFOND 4

colore le fond de l'écran en orange.

COLOREPLUME en abrégé **CP**

Requiert un nombre comme argument d'entrée. Colore les lignes tracées par la tortue. La correspondance code - couleur sur fond noir est la même que pour COLOREFOND. Pour des fonds colorés, se reporter au manuel d'apprentissage (PREMIERS PAS AVEC EDILOGO, page 149). Ce primitif n'a pas d'effet rétroactif sur les tracés déjà effectués, mais affecte les lignes à venir.

Exemple :

COLOREPLUME 3

Sur fond noir, les tracés effectués dorénavant sont mauves.

DESSINE Fait passer en mode graphique. Nettoie l'écran graphique, remet la tortue au centre et la rend visible. DESSINE est sans influence sur l'option PAGE ou ENROULE, ni sur la couleur du fond.

DROITE en abrégé **DR**

Requiert un nombre comme argument d'entrée. Fait pivoter la tortue du nombre indiqué de degrés dans le sens trigonométrique inversé (sens des aiguilles d'une montre).

Exemple :

DROITE 90

fait pivoter la tortue d'un quart de tour.

ECRANMIXTE Peut également être obtenu grâce à CTRL **S**
En mode graphique, donne un écran dont la partie supérieure est réservée au dessin et les quatre lignes inférieures au texte. C'est le statut normal en mode graphique.

ECRIS en abrégé **EC**

Sort du mode graphique, et donne une page écran de texte vide, le curseur étant positionné en haut à gauche.

ENROULE En mode graphique, c'est le statut normal. La tortue peut alors sortir de l'écran et poursuivre son trajet en rentrant par le bord opposé. On peut éviter l'effet d'enroulement grâce au primitif PAGE. L'option ENROULE ou PAGE n'est pas affectée par DESSINE ou VIDEECRAN.

ETATTORTUE en abrégé **ET**

Sort une liste de quatre éléments, qui sont dans l'ordre :

VRAI ou FAUX selon que la plume est posée ou levée
VRAI ou FAUX selon que la tortue est montrée ou cachée
numéro de couleur du fond
numéro de couleur de la plume

Exemple :

```
AFFICHE ETATTORTUE
VRAI FAUX Ø 2
```

indique que la plume est posée, la tortue invisible, le fond noir et la plume verte.

FIXECAP en abrégé **FCAP**

Requiert un nombre comme argument d'entrée. Fait pivoter la tortue de manière qu'elle regarde le cap indiqué. Le cap 0 est le "nord" de l'écran, les mesures sont données en degrés, et comptées positivement dans le sens des aiguilles d'une montre.

Exemple :

```
FIXECAP 18Ø
```

fait regarder la tortue vers le "sud"

FIXEX Requiert un nombre comme argument d'entrée, met la tortue au point d'abscisse indiquée, sans modifier l'ordonnée ni le cap. Laisse une trace si la plume est posée.

FIXEXY Requiert deux nombres comme arguments d'entrée (si le deuxième est négatif, il convient de le parenthèser). Met la tortue au point dont l'abscisse est le premier nombre indiqué, et dont l'ordonnée est le deuxième nombre indiqué. Laisse une trace si la plume est posée. Ne modifie pas le cap.

FIXEY Requiert un nombre comme argument d'entrée. Met la tortue au point d'ordonnée indiquée, sans modifier l'abscisse ni le cap. Laisse une trace si la plume est posée.

GAUCHE en abrégé **GA**

Requiert un nombre comme argument d'entrée. Fait pivoter la tortue du nombre de degrés indiqués dans le sens trigonométrique direct (sens inverse des aiguilles d'une montre).

Exemple :

```
GAUCHE 18Ø
```

fait pivoter d'un demi-tour.

LEVEPLUME en abrégé **LP**

Relève le traceur (la plume) de la tortue : un déplacement ultérieur s'effectuera sans laisser de trace.

MONTRETORTUE en abrégé **MT**

Fait réapparaître le triangle qui symbolise la tortue. Permet en particulier de visualiser le cap de la tortue.

PAGE En mode graphique, la tortue ne peut plus sortir de l'écran. Toute instruction qui tendrait à lui faire dépasser les limites de l'écran sera refusée, et un message

TORTUE HORS ECRAN

avertira de l'impossibilité d'exécuter cette instruction. On peut sortir du mode PAGE grâce au primitif ENROULE. L'option PAGE (comme l'option ENROULE) n'est pas affectée par DESSINE ou VIDEECRAN.

PLEINECRAN Peut également être obtenu grâce à CTRL **F**

En mode graphique, donne un écran entièrement réservé au dessin, sans aucune ligne de texte.

POSEPLUME en abrégé **PP**

Abaisse le traceur (la plume) de la tortue. Un déplacement ultérieur s'effectuera en laissant une trace.

RECULE en abrégé **RE**

Requiert un nombre comme argument d'entrée. Fait reculer la tortue du nombre de pas indiqué, en conservant la direction que regarde la tortue. Si la plume est posée, ce déplacement laisse une trace.

Exemple :

```
RECULE 50
```

La tortue recule de cinquante pas.

VERS Requiert deux nombres comme arguments d'entrée. Si le deuxième est négatif, il convient de le parenthèser. Ces deux nombres étant interprétés comme l'abscisse et l'ordonnée d'un point de l'écran, le primitif sort le cap du segment orienté ayant pour origine l'endroit où se trouve la tortue, et pour extrémité le point défini par ces coordonnées.

Exemple :

```
FIXECAP VERS 30 80
```

tourne le regard de la tortue vers le point de coordonnées (30,80)

VIDEECRAN en abrégé **VE**

En mode graphique, nettoie l'écran graphique, sans modifier la position ni le cap de la tortue. Rend visible la tortue si elle était cachée. VIDEECRAN est sans influence sur l'option PAGE ou ENROULE.

XCOR Sort sous forme décimale l'abscisse de la position de la tortue

YCOR Sort sous forme décimale l'ordonnée de la position de la tortue.

- ***Mots et listes***

DERNIER en abrégé **DER**

Requiert un argument d'entrée. Si cet argument est une liste, sort le dernier élément de cette ligne. Si cet argument est un mot, sort le dernier caractère de ce mot.

Signale une erreur quand l'argument est le mot vide ou la liste vide.

Exemples :

```
AFFICHE DERNIER 12345
5
AFFICHE DERNIER "CHOCOLAT
T
AFFICHE DERNIER [ SOUS LES LILAS BLANCS ]
BLANCS
```

INSERED en abrégé **ID**

Requiert deux arguments d'entrée, le deuxième étant obligatoirement une liste (éventuellement vide). Sort une liste constituée des éléments du deuxième argument, suivis du premier argument.

Exemples :

```
AFFICHE INSERED "ICI [ IL FAIT BEAU ]
IL FAIT BEAU ICI
AFFICHE INSEREP [ UN DEUX ] [ IL PLEUT ]
IL PLEUT [ UN DEUX ]
```

INSEREP en abrégé **IP**

Requiert deux arguments d'entrée, le deuxième étant obligatoirement une liste (éventuellement vide). Sort une liste, constituée des éléments du deuxième argument, précédés du premier argument.

Exemples :

```
AFFICHE INSEREP "ICI [ IL FAIT BEAU ]
ICI IL FAIT BEAU
AFFICHE INSERED [ UN DEUX ] [ IL PLEUT ]
[ UN DEUX ] IL PLEUT
```

LISTE Requiert deux (ou plus, si l'instruction est parenthèsée) arguments d'entrée. Sort une liste dont les éléments sont les éléments des deux arguments. Certains de ces arguments peuvent être vides.

Exemples :

```
AFFICHE LISTE "BONJOUR "BONSOIR
BONJOUR BONSOIR
AFFICHE LISTE [ BONJOUR TOUT LE MONDE ] ...
                                    ... [ CA VA ? ]
[ BONJOUR TOUT LE MONDE ] [ CA VA ? ]
AFFICHE ( LISTE "SALUT [ LES COPAINS ] ...
                                    ... [ TOUS DANS LE BAIN ] )
SALUT [ LES COPAINS ] [ TOUS DANS LE BAIN ]
```

Il est indispensable de laisser un espace avant la parenthèse fermante.

MOT Requiert deux (ou plus, si l'instruction est parenthèsée) mots, qui peuvent être éventuellement vides, comme arguments d'entrée. Sort un mot formé par concaténation des caractères des deux arguments.

Exemples :

```
AFFICHE MOT "SAL "AMI
SALAMI
AFFICHE MOT 18 75
1875
```

```
AFFICHE ( MOT "DO "MI "NI "QUE )
DOMINIQUE
```

Il est indispensable de laisser un espace avant la parenthèse fermante.

PHRASE en abrégé **PH**

Requiert deux (ou plus, si l'instruction est parenthèsée) arguments d'entrée. Si les arguments sont des listes, combine tous les éléments en une seule liste, et sort cette liste. Si un argument est un mot, cet argument est considéré comme une liste à un seul élément. *Exemples :*

```
AFFICHE PHRASE [ BONJOUR TOUT ] ...
                ... [ LE MONDE ] [ COMMENT CA VA? ]
BONJOUR TOUT LE MONDE COMMENT CA VA?
AFFICHE PHRASE "SALUT [ LES COPAINS ]
SALUT LES COPAINS
AFFICHE ( PHRASE [ UN DEUX ] ...
                ... "TROIS [ ALLONS AU BOIS ] )
UN DEUX TROIS ALLONS AU BOIS
```

Il est indispensable de laisser un espace avant la parenthèse fermante.

PREMIER en abrégé **PR**

Requiert un argument d'entrée. Si cet argument est une liste, sort le premier élément de cette liste. Si cet argument est un mot, sort le premier caractère de ce mot. Signale une erreur si l'argument est la liste vide ou le mot vide.

Exemples :

```
AFFICHE PREMIER 12345
1
AFFICHE PREMIER "CHOCOLAT
C
AFFICHE PREMIER [ SOUS LES LILAS BLANCS ]
SOUS
```

SAUFDERNIER en abrégé **SD**

Requiert un argument d'entrée. Si cet argument est une liste, sort une liste qui contient tous les éléments de la liste donnée, à l'exception du dernier. Si cet argument est un mot, sort un mot formé de tous les caractères du mot donné, à l'exception du dernier. Signale une erreur si l'argument est la liste vide ou le mot vide.

Exemples :

```
AFFICHE SAUFDERNIER 12345
1234
AFFICHE SAUFDERNIER "CHOCOLAT
CHOCOLA
AFFICHE SAUFDERNIER ...
... [ SOUS LES LILAS BLANCS ]
SOUS LES LILAS
```

SAUFPREMIER en abrégé **SP**

Requiert un argument d'entrée. Si cet argument est une liste, sort une liste qui contient tous les éléments de la liste donnée, à l'exception du premier. Si cet argument est un mot, sort un mot formé de tous les caractères du mot donné, à l'exception du premier. Signale une erreur si l'argument est la liste vide ou le mot vide.

Exemples :

```
AFFICHE SAUFPREMIER 12345
2345
AFFICHE SAUFPREMIER "CHOCOLAT
HOCOLAT
AFFICHE SAUFPREMIER ...
... [ SOUS LES LILAS BLANCS ]
LES LILAS BLANCS
```

- ***Définition et édition de procédures***

DEFINIS Requiert deux arguments d'entrée. Le premier doit être un nom, qui est destiné à être le titre de la procédure : en règle générale

il doit être précédé du caractère *guillemets.* Le second argument d'entrée doit être une liste dont les éléments sont eux-mêmes des listes : la première de ces listes est la suite des paramètres de la procédure (dans le cas d'une procédure sans paramètres, cette première liste doit être la liste vide). Chacune des listes suivantes correspond aux lignes de la procédure en cours de définition.

Exemple :

```
DEFINIS "CARRE [[LONGUEUR] [REPETE 4 ...
                ... [AVANCE :LONGUEUR DROITE 90]]]
```

Normalement on utilise les primitifs POUR ou EDITE pour définir des procédures. DEFINIS est surtout utile pour écrire des procédures qui définissent d'autres procédures.

EDITE en abrégé **ED**

Sans argument d'entrée, passe en mode édition, en ramenant le contenu du tampon d'édition écran, c'est-à-dire la (ou les) procédure(s) le plus récemment éditée(s) ou imprimée(s).
Si l'on précise un titre de procédure en argument d'entrée, passe en mode édition en ramenant le texte de cette procédure dans le tampon d'édition. On peut également éditer TOUT, ou les NOMS, ou les PROCEDURES.

FIN Termine la définition d'une procédure. Il n'est pas nécessaire d'indiquer FIN à la fin de la dernière procédure, mais ce mot est indispensable pour séparer les procédures les unes des autres dans le cas d'une définition simultanée (sans validation par CTRL **C** entre temps) de plusieurs procédures.

POUR Premier mot dans la définition d'une procédure. Le nombre d'arguments d'entrées est variable. Sans argument d'entrée, passe en mode éditeur avec un tampon d'édition vide. Avec un ou plusieurs arguments d'entrée, passe en mode éditeur avec la procédure ayant pour titre le premier argument donné ; tout autre argument d'entrée est considéré comme un paramètre de cette procédure (et doit donc être précédé du caractère **:**).

TEXTE Requiert un nom de procédure comme argument d'entrée (ce nom doit être précédé du caractère *guillemets,* sinon LOGO exécute la procédure). Sort le texte de la procédure sous forme d'une liste. S'il n'y a pas de procédure de ce nom déjà définie, TEXTE sort la liste vide : [ ]

• *Création de noms*

CHOSE Requiert un nom comme argument d'entrée, et sort la valeur qui a été attribuée à ce nom. Ce primitif permet une évaluation à un niveau supplémentaire comme le montrent les deux dernières lignes de l'exemple.

Exemple :

```
CREE "CHAT "MINOU
CREE "MINOU "BLANCHETTE
AFFICHE :CHAT
MINOU
AFFICHE CHOSE "CHAT
MINOU
AFFICHE CHOSE :CHAT
BLANCHETTE
```

CREE Requiert deux arguments d'entrée : le premier est obligatoirement un nom (il doit donc en règle générale être précédé du caractère *guillemets*), qui sera le nom du paramètre. Le deuxième argument d'entrée est la valeur associée à ce nom.

Exemple :

```
CREE "VALEUR 485960
AFFICHE :VALEUR
485960
AFFICHE :VALEUR + 10
485970
```

- ***Opérations numériques***

+ (préfixé)
Requiert un nombre (sans signe) comme argument d'entrée. Autre façon d'écrire un nombre positif.

Exemple :

```
AFFICHE + 3.8
3.8
```

+ (infixé)
Requiert deux nombres comme arguments d'entrée et sort la somme de ces deux nombres

Exemple :

```
AFFICHE 7.1E4 + 4.3E5
501000
```

[−] (préfixé)
Requiert un nombre comme argument d'entrée, et sort l'opposé de ce nombre.

Exemple :

```
AFFICHE − 3.8 > 0
FAUX
```

[−] (infixé)
Requiert deux nombres comme arguments d'entrée et sort la différence de ces deux nombres.

Exemple :

```
AFFICHE −5 − (−4)
−1
```

[*] (infixé)
Requiert deux nombres comme arguments d'entrée, et sort le produit de ces deux nombres.

Exemple :

```
AFFICHE 8.2E4 * 2
164000
```

[/] (infixé)
Requiert deux nombres comme arguments d'entrée, et sort le quotient décimal de ces deux nombres.

Exemple :

```
AFFICHE 4 / 3
1.33333
AFFICHE 35 / 7
5.
```

= (infixé)
Requiert deux arguments d'entrée. Les compare et sort "VRAI s'il s'agit de nombres égaux, "FAUX sinon.

Exemple :

```
AFFICHE 3 + 8 * 4 = 35
VRAI
```

ARRONDI Requiert un nombre comme argument d'entrée. Sort le nombre entier (relatif) le plus proche du nombre donné.

Exemple :

```
AFFICHE ARRONDI 3.2
3
AFFICHE ARRONDI -172.8
-173
```

ATG Requiert deux nombres comme arguments d'entrée. Sort l'arc tangente du quotient de ces deux nombres. L'angle est mesuré en degrés entre 0 et 360. Le quadrant de l'angle est déterminé par les signes des deux nombres donnés.

AUHASARD Sans argument d'entrée, initialise au hasard la prochaine séquence de nombres au hasard.

Avec un nombre comme argument d'entrée, fixe au nombre donné le générateur de nombres aléatoires, ce qui permet de retrouver ultérieurement une même suite de nombres aléatoires. Dans ce cas, il convient d'écrire cette instruction entre parenthèses.

Exemple :

```
(AUHASARD 20) REPETE 4 [ AFFICHE HASARD 50 ]
14
11
38
6
(AUHASARD 20) REPETE 4 [ AFFICHE HASARD 50 ]
14
11
38
6
```

COS Requiert un nombre comme argument d'entrée. Sort le cosinus de ce nombre, considéré comme un angle mesuré en degrés.

Exemple :

```
AFFICHE COS 60
.5
```

ENTIER Requiert un nombre comme argument d'entrée. Sort la partie entière, ignorant la partie décimale.

Exemple :

```
AFFICHE ENTIER 7.9
7
AFFICHE ENTIER -83.7
-83
```

HASARD Requiert un entier naturel *n* comme argument d'entrée. Sort un entier aléatoire compris entre 0 et *n* − 1. Quand LOGO vient d'être chargé, des séquences identiques d'appels au primitif HASARD produisent les mêmes sources de nombres aléatoires, à moins de faire précéder par AUHASARD la séquence d'appels à HASARD.

QUOTIENT Requiert deux nombres comme arguments d'entrée. Sort le quotient entier de ces deux arguments. Dans le cas où les arguments ne sont pas entiers, il y a d'abord arrondi aux entiers les plus proches.

Exemple :

```
AFFICHE QUOTIENT 8.4 2.3
4
```

RCAR Requiert un nombre positif comme argument d'entrée. Sort la racine carrée du nombre donné.

Exemple :

```
AFFICHE RCAR 6400
80
```

RESTE Requiert deux nombres comme arguments d'entrée. Sort le reste dans la division euclidienne du premier nombre par le second. Dans le cas où les arguments ne sont pas entiers, il y a d'abord arrondi aux entiers les plus proches.

Exemple :

```
AFFICHE RESTE 60 7
4
```

SIN Requiert un nombre comme argument d'entrée. Sort le sinus de ce nombre, considéré comme un angle mesuré en degrés.

Exemple :

```
AFFICHE SIN 45
.707107
```

- ***Conditionnels***

ALAFOIS Requiert deux (ou plus, si l'expression est parenthèsée) arguments d'entrée. Sort "VRAI si chacun des arguments a la valeur "VRAI, "FAUX si au moins l'un des arguments a la valeur "FAUX.

Exemples :

```
AFFICHE ALAFOIS 1 + 1 = 2 3 * 4 = 14 - 2
VRAI
AFFICHE (ALAFOIS 3 = 2 + 1 Ø = Ø 4 * 5 = 9)
FAUX
```

Dans le cas où il y a plus de deux arguments, il est indispensable de laisser un espace entre le dernier argument et la parenthèse fermante.

ALORS Utilisé dans SI...ALORS...SINON...

NON Requiert un argument d'entrée. Sort "VRAI si l'argument a la valeur FAUX et "FAUX si l'argument a la valeur VRAI.

Exemple :

```
SI NON 3 < 2 AFFICHE "LOUPE
LOUPE
```

SI Le format standard est SI { condition } ALORS { expression 1 } SINON { expression 2 } . La { condition } est évaluée ; si sa valeur est VRAI, l' { expression 1 } est exécutée. Si sa valeur est FAUX, l' { expression 2 } est exécutée. Le mot ALORS est facultatif, et la partie SINON { expression 2 } peut être omise. La { condition } doit être une expression LOGO qui sort "VRAI ou "FAUX, par exemple les primitifs terminés par un caractère ? ou les fonctions de test habituelles , =, >, <, NON etc...

SIFAUX en abrégé **SIF**

Exécute le reste de la ligne seulement si le résultat du dernier TESTE effectué est "FAUX. Dans le cas contraire, ignore le reste de la ligne.

SINON Utilisé dans SI ... ALORS ... SINON ...

SIVRAI en abrégé **SIV**

Exécute le reste de la ligne seulement si le résultat du dernier TESTE était "VRAI. Dans le cas contraire, ignore le reste de la ligne.

TESTE Requiert une condition (expression qui sort VRAI ou FAUX) en argument d'entrée. Vérifie la valeur de vérité de cette condition. Si la valeur trouvée est VRAI, exécute l'expression qui suit SIVRAI. Si la valeur trouvée est FAUX, exécute l'expression qui suit SIFAUX.

Exemple :

```
POUR DING
TESTE "DING = "DING
SIVRAI AFFICHE "DONG
SIFAUX AFFICHE "PASDING
FIN
```

Produit à l'exécution :

```
DING
DONG
```

En effet, TESTE trouve "VRAI et exécute la ligne SIVRAI dans la procédure DING ; d'où affichage du mot "DONG.

UNDE Requiert deux (ou plus si l'expression est parenthèsée) arguments d'entrée. Sort "VRAI si au moins l'un des arguments a la valeur VRAI, "FAUX si tous les arguments ont la valeur FAUX.

Exemple :

```
AFFICHE UNDE 1 + 1 = 2 3 * 2 = 14 - 4
VRAI
AFFICHE ( UNDE 3 = 2 3 > 5 3 * 9 = 54 )
FAUX
```

Dans le cas où il y a plus de deux arguments, il est indispensable de laisser un espace entre le dernier argument et la parenthèse fermante.

- ***Prédicats utilisés avec les conditionnels***

[=] Requiert deux arguments d'entrée.

Si les deux arguments sont des nombres, compare ces valeurs et sort "VRAI s'il y a égalité, "FAUX sinon.

Si les deux arguments sont des mots, compare les deux suites de caractères et sort "VRAI si elles sont identiques, "FAUX sinon.

Si les deux arguments sont des listes, compare les deux suites de leurs éléments, et sort "VRAI si elles sont identiques, "FAUX sinon. Sort "FAUX si les arguments ne sont pas de même nature. (*rappel* : un nombre est un mot particulier).

Exemple :

```
AFFICHE 2 = 7 - 5
VRAI
AFFICHE 2 = "2
VRAI
AFFICHE "ABC = "BAC
FAUX
AFFICHE "MARI = MOT "MA "RI
VRAI
AFFICHE "BONJOUR = [BONJOUR]
FAUX
```

[<] Requiert deux nombres comme arguments d'entrée. Les compare et sort "VRAI si le premier argument est inférieur au second, "FAUX sinon.

Exemple :

```
AFFICHE 12 < 8 + 7
VRAI
```

[>] Requiert deux nombres comme arguments d'entrée. Les compare et sort "VRAI si le premier argument est supérieur au second, "FAUX sinon.

BOUTON? Requiert un argument d'entrée, qui doit être un nombre entier compris entre 0 et 2. Sort "VRAI ou "FAUX selon que le bouton situé sur la manette correspondante est pressé ou non. Remarque : la manette (paddle) numéro 3 sur l'Apple n'a pas de bouton.

Exemple :

```
SI BOUTON? 1 ="VRAI ALORS ECRIS
```

CHOSE? Requiert un argument d'entrée qui doit être un mot, et qui doit donc, en règle générale, être précédé du caractère *guillemets*.
Sort "VRAI si l'argument est un nom de paramètre (un nom auquel a été attribuée une valeur).

Exemple : (voir CHOSE p. 217)

```
AFFICHE CHOSE? "CHAT
VRAI
AFFICHE CHOSE? :CHAT
VRAI
AFFICHE CHOSE? "BLANCHETTE
FAUX
```

LC? Sort "VRAI si une touche du clavier a été frappée (c'est-à-dire si le tampon des caractères du clavier contient une valeur, quelle qu'elle soit). Sort "FAUX si le tampon des caractères du clavier est vide.

LISTE? Requiert un argument d'entrée. Sort "VRAI si l'argument est une liste, "FAUX sinon.

Exemple :

```
AFFICHE LISTE? PHRASE "RI "ME
VRAI
```

MOT? Requiert un argument d'entrée ; sort "VRAI si l'argument est un mot, "FAUX sinon.

Exemple :

```
AFFICHE MOT? PREMIER PHRASE ...
... [ IL FAIT BEAU ] [ CE MATIN ]
VRAI
AFFICHE MOT? PREMIER LISTE ...
... [ IL FAIT BEAU ] [CE MATIN ]
FAUX
```

NOMBRE? Requiert un argument d'entrée ; sort "VRAI si l'argument est un nombre, "FAUX sinon.

Exemple :

```
SI NOMBRE? DERNIER "COTE1 AFFICHE ...
                          ... [ VARIABLE INDEXEE ]

VARIABLE INDEXEE
```

- ***Contrôle***

EXECUTE Requiert une liste comme argument d'entrée. Exécute la liste comme s'il s'agissait d'une suite d'instructions tapées au clavier. Attention, la liste ne doit pas comporter plus de 255 caractères.

Exemple :

```
CREE "MULT [ AFFICHE 5 * ]
EXECUTE PHRASE :MULT 8
40
```

NIVEAUSUP Ne s'utilise qu'à l'intérieur d'une procédure. Arrête l'exécution de la procédure en cours, et celle de toutes les procédures appelantes, et rend le contrôle à l'utilisateur. C'est un primitif assez peu utilisé. On notera la différence entre NIVEAUSUP et STOP: alors que NIVEAUSUP arrête l'exécution de tout, STOP au contraire arrête seulement l'exécution de la procédure en cours, en transférant l'exécution à la procédure appelante.

REPETE Requiert deux arguments d'entrée. Le premier doit être un nombre, et le deuxième argument une liste. EXECUTE la liste donnée le nombre de fois indiqué.

Exemples :

```
REPETE 4 [ AVANCE 20 DROITE 90 ]
```

trace un carré et remet la tortue dans sa position de départ (point et cap inchangés).

```
REPETE 3 [ AFFICHE "COUCOU ]
COUCOU
COUCOU
COUCOU
```

SORS Requiert un argument d'entrée. Ne s'utilise qu'à l'intérieur d'une procédure. Provoque l'arrêt de la procédure en cours d'exécution,

et transmet l'argument d'entrée à la procédure appelante, ou au niveau supérieur si on est dans une procédure de niveau 1.
Si l'argument d'entrée doit être évalué, c'est le résultat de l'évaluation qui est transmis.

Exemple :

```
POUR CARRE :N
SORS :N * :N
FIN
AFFICHE CARRE 21
441
POUR CUBE :N
SORS :N * CARRE :N
FIN
AFFICHE CUBE 10
1000
```

STOP Ne s'utilise qu'à l'intérieur d'une procédure. Arrête l'exécution de la procédure en cours, et rend le contrôle à la procédure appelante (ou à l'utilisateur, si STOP se trouve dans une procédure de niveau 1). On notera la différence entre STOP et FIN. FIN indique (principalement pour la gestion dans l'éditeur) où se termine le texte d'une procédure.

Exemple :

```
POUR REBOURS :NOMBRE
SI :NOMBRE < 0 STOP
AFFICHE :NOMBRE
REBOURS :NOMBRE - 1
FIN
REBOURS 4
4
3
2
1
0
```

VA Requiert un argument d'entrée, qui doit être une étiquette. Ne s'utilise qu'à l'intérieur d'une procédure, et transfère l'exécution à la ligne indiquée par l'étiquette. (Une étiquette est un mot dont le dernier caractère est obligatoirement le signe **:**)

Exemple :

```
POUR BOUCLE
AFFICHE [ PREMIER TOUR ]
AFFICHE [ C'EST FAIT ]
ETIQUETTE: AFFICHE [ ENCORE UN TOUR ]
AFFICHE [ AU SUIVANT ]
VA "ETIQUETTE
FIN
```

Ce primitif est rarement utilisé en LOGO, car il est beaucoup plus efficace et compréhensible d'utiliser REPETE ou de construire des procédures, le cas échéant récursives. (Harold ABELSON, dans son manuel, note que "parsemer des procédures LOGO de VA, c'est comme verser du ketchup sur du caviar" !) L'utilisation de VA, bien que parfois précieuse, conduit le plus souvent à des procédures peu lisibles, et en général difficiles à "débugger".

- ***Entrées - Sorties***

AFFICHE en abrégé **AF**

Requiert un (ou plus, si l'expression est parenthèsée) argument(s) d'entrée. Affiche le ou les argument(s) sur l'écran, et positionne le curseur au début de la ligne suivante. Les mots sont affichés sans le premier caractère *guillemets* et les listes sans leurs crochets externes. Dans le cas de plusieurs arguments, affiche ces arguments sur une même ligne en les séparant par des espaces. Dans le cas où l'argument est une procédure, ce n'est pas le texte de la procédure qui sera affiché (voir TEXTE ou IMPRIME), mais la procédure sera exécutée : si elle sort un argument, c'est cet argument de sortie qui sera affiché, sinon une erreur sera signalée.

Exemples :

```
AFFICHE "SALUT
SALUT
(AFFICHE "SALUT "LES "COPAINS )
SALUT LES COPAINS
```

```
AFFICHE [ SALUT LES COPAINS ]
SALUT LES COPAINS
```

AFR (mis pour Affiche et Reste)

Requiert un (ou plus, si l'expression est parenthèsée) argument(s) d'entrée. Affiche les arguments sur l'écran, sans laisser d'espace ni retourner à la ligne entre deux arguments. Les listes sont affichées *sans* leurs crochets extérieurs.

Exemple :

```
( AFR [IL FAIT BEAU ] [CE MATIN ] )
IL FAIT BEAUCE MATIN
```

(Voir AFFICHE)

ASCII Requiert un caractère comme argument d'entrée, et sort le nombre qui est le code ASCII de ce caractère.

Exemples :

```
AFFICHE ASCII "A
65
AFFICHE ASCII "!
33
AFFICHE ASCII "4
52
```

BOUTON? Requiert un argument d'entrée qui doit être un nombre entier compris entre 0 et 2. Sort "VRAI ou "FAUX selon que le bouton situé sur la manette correspondante est pressé ou non. Remarque : la manette (paddle) numéro 3 sur l'Apple n'a pas de bouton.

CAR Requiert un nombre entier comme argument d'entrée. Sort le caractère dont ce nombre est le code ASCII.

Exemple :

```
AFFICHE CAR 83
S
```

CURSEUR Requiert deux entiers naturels comme arguments d'entrée. Le premier doit être compris entre 0 et 39 et indique la colonne où

sera positionné le curseur. Le deuxième doit être compris entre 0 et 23 et indique la ligne où sera positionné le curseur.
CURSEUR 23 39 positionne le curseur en bas à droite de l'écran.

LC? Sort "VRAI si une touche du clavier a été frappée (c'est-à-dire si le tampon des caractères du clavier contient une valeur, quelle qu'elle soit). Sort "FAUX si le tampon des caractères du clavier est vide.

LISCAR en abrégé **LC**

Sort le premier caractère situé dans le tampon des caractères du clavier. Si le tampon est vide, arrête l'exécution de la procédure jusqu'à ce qu'une touche ait été pressée.
Voir des exemples d'utilisation dans le programme PICOLO, qui figure sur la disquette utilitaire.

LISLIGNE en abrégé **LL**

Arrête l'exécution de la procédure, jusqu'à ce que l'utilisateur ait tapé quelque chose au clavier, et validé sa réponse en appuyant sur RETURN . Sort cette ligne (comme une *liste*).

Exemple :

```
AFFICHE [ VOULEZ-VOUS CONTINUER: OUI - NON ]
SI PREMIER LISLIGNE = "OUI ...
... ALORS SUITE SINON TERMINE
```

MANETTE Requiert un nombre entier compris entre 0 et 3 comme argument d'entrée. Sort un nombre compris entre 0 et 255, correspondant à la position du cadran sur la manette de jeu (paddle) concernée.

PERIPH Requiert un argument d'entrée, qui doit être le numéro d'un connecteur (slot) de la carte Apple ; transfère alors sur le périphérique branché sur ce connecteur tous les affichages et messages d'erreur, au lieu de les envoyer sur l'écran. PERIPH 0 ramène le transfert à l'écran. Un "numéro de connecteur" supérieur à 8, est interprété comme l'adresse d'une routine rédigée par l'utilisateur en langage assembleur (voir chapitre 5).

VIDETAMPON Vide le tampon des caractères du clavier. Cela est utile pour éviter qu'une frappe des caractères trop rapide par rapport à la vitesse d'exécution de la procédure donne l'impression que les réponses de la machine sont systématiquement en retard par rapport à l'utilisateur.

VIDETEXTE Nettoie l'écran de tout l'affichage du texte et ramène le curseur au début de la première ligne disponible (haut de l'écran en mode ECRIS, ligne supérieure des quatre lignes du bas de l'écran en mode DESSINE).

- ***Gestion des fichiers et de l'espace de travail***

AUREVOIR Nettoie l'espace de travail et ré-initialise LOGO. Ne vide pas l'espace alloué à l'utilisateur pour ses procédures en langage machine.

CATALOGUE Affiche le nom des fichiers présents sur la disquette actuellement en position dans le lecteur. Précise pour chacun de ces fichiers la nature du contenu : procédures et noms sont repérés par la mention .LOGO à côté du titre du fichier, les copies de dessins d'écran par .DESSIN, les procédures en langage machine par .BIN.

DETRUISDESSIN Requiert un argument d'entrée qui doit être un nom de fichier de copie d'écran. Détruit ce fichier et son contenu.

DETRUISFICHIER Requiert un argument d'entrée qui doit être un nom de fichier de procédures et variables. Détruit ce fichier et son contenu.

EFFACE en abrégé **EF**

Requiert un argument d'entrée.

Si cet argument est le nom d'une procédure, fait disparaître la définition de cette procédure de l'espace de travail (si cette procédure n'a pas été sauvegardée auparavant dans un fichier sur disquette, elle est définitivement perdue).

Exemple :

```
EFFACE CARRE
```

Si cet argument est PROCEDURES, fait disparaître toutes les définitions des procédures de l'espace de travail.

Si cet argument est NOMS, fait disparaître toutes les variables de l'espace de travail.

Si cet argument est TOUT, fait disparaître à la fois les procédures et les variables.

EFNOM Requiert un argument d'entrée, qui doit être le nom d'une variable actuellement disponible dans l'espace de travail. Fait disparaître cette variable de l'espace de travail. Contrairement à ce qui se passe avec les procédures, il y a évaluation de l'argument. Aussi pour effacer la variable BIDULE qui a été créée par

```
CREE "BIDULE "MACHIN
```

on tapera EFNOM "BIDULE. Si vous tapez EFNOM BIDULE, BIDULE est considéré comme une procédure, et LOGO tentera de l'exécuter.

Exemple :

```
POUR CRI
CREE "TOUTOU "OUAH
SORS :TOUTOU
FIN
AFFICHE CRI
OUAH
IMPRIME NOMS
"TOUTOU EST OUAH
"BIDULE EST MACHIN
EFNOM CRI
IMPRIME NOMS
"BIDULE EST MACHIN
```

GARDE Requiert un nom comme argument d'entrée. Ce nom sera le nom du fichier. Effectue une copie du contenu complet de l'espace de travail (noms et procédures) et la garde dans un fichier sur la disquette. Utilisé en mode DESSINE, détruit l'image actuellement affichée sur l'écran.

Exemple :

```
GARDE "MONBOULOT
```

S'il existait un fichier du nom MONBOULOT sur la disquette, celui-ci est détruit et remplacé par le nouveau fichier. Il convient donc d'être extrêmement prudent en utilisant des noms de fichiers déjà affectés.

GARDEDESSIN Requiert un nom comme argument d'entrée. Ce nom sera le nom du fichier. Effectue une copie de l'image actuellement affichée à l'écran, et la garde dans un fichier sur la disquette.

Exemple :

```
GARDEDESSIN "PAYSAGE
```

IMPRIME en abrégé **IM**

Utilisé seul, affiche à l'écran le texte de la dernière procédure définie, éditée ou imprimée.

Suivi d'un nom de procédure comme argument, affiche à l'écran le texte de cette procédure.

Exemple :

```
IMPRIME CRI
```

```
POUR CRI
  CREE "TOUTOU "OUAH
  SORS "TOUTOU
FIN
```

Suivi de TOUT, affiche à l'écran à la fois les procédures et les variables.
Suivi de PROCEDURES, affiche à l'écran tous les textes de procédures, mais pas les variables.
Suivi de NOMS, affiche à l'écran tous les noms de variables et leur valeur.

Exemple :

```
IMPRIME NOMS
"BIDULE EST MACHIN
"TOUTOU EST OUAH
```

Suivi de TITRES (on peut écrire en un seul mot en abrégé IMTS), affiche à l'écran tous les titres des procédures actuellement disponibles dans l'espace de travail.

RAMENE Requiert un nom de fichier comme argument d'entrée. Lit ce fichier sur la disquette, et en ramène une copie dans l'espace de travail. Utilisé en mode DESSINE, détruit la page graphique actuellement affichée à l'écran.

Exemple :

```
RAMENE "MONBOULOT
```

RAMENEDESSIN Requiert un nom de fichier de dessin comme argument d'entrée. Lit ce fichier sur la disquette, et en ramène sur l'écran une copie qui se substitue à l'image actuellement affichée.

- ***Aide au dépistage d'erreurs (debugging)***

CONTINUE en abrégé **CO**
Reprend l'exécution après une pause (PAUSE ou CTRL **Z**)

DETRACE Sort du mode TRACE

PAUSE peut aussi s'obtenir par CTRL **Z**
Ne s'emploie qu'à l'intérieur d'une procédure. Suspend l'exécution de la procédure, affiche à l'écran le niveau où s'est produite l'inter-

ruption (c'est-à-dire le nombre de procédures actuellement en cours d'exécution), et permet à l'utilisateur de faire exécuter des instructions.

L'exécution de la procédure reprend en tapant CONTINUE (ou CO en abrégé), suivi d'une validation par touche [RETURN]. [CTRL] **G** arrête définitivement l'exécution et rend le contrôle au niveau supérieur.

[TRACE] Produit une exécution ligne par ligne de toutes les procédures appelées ultérieurement. Pour chaque procédure, il y a affichage à l'écran du titre de la procédure, avec la valeur de ses arguments d'entrée, puis affichage de chacune des lignes de texte de la procédure : l'exécution de cette ligne est provoquée par la frappe d'une touche quelconque du clavier, et la ligne suivante est alors affichée.

[CTRL Z] Produit une pause (voir PAUSE)

[CTRL G] Arrête l'exécution de la procédure et rend le contrôle au niveau supérieur.

- ***Commandes diverses***

[.APPEL] Requiert deux arguments d'entrée ; le premier est un nombre qui désigne une adresse, et donne le contrôle à cette adresse, pour l'exécution d'une routine conçue par l'utilisateur et écrite en langage assembleur ; le deuxième argument est stocké en mémoire de manière à pouvoir être examiné par la routine. Ce primitif permet à l'utilisateur de réaliser ses propres primitifs et de les utiliser via LOGO.
(voir chapitre 5)

[.ASPECT] Requiert un nombre décimal (voisin de 1) comme argument d'entrée. Permet de modifier le rapport d'échelle entre les "points" verticaux et les "points" horizontaux. En France, 1 est en général une assez bonne valeur pour donner une impression d'isotropie de l'écran. Au Québec, .8 donne un résultat assez satisfaisant. Il convient d'opérer différents essais pour chaque téléviseur ou moniteur, de légères variations pouvant être constatées d'un appareil à un autre. C'est aussi avec .ASPECT qu'on obtiendra une copie sur papier respectant l'isotropie (car les imprimantes ont elles aussi une personnalité !).

Avec des valeurs très éloignées de 1, on risque d'avoir quelques aberrations sur la position de la tortue par rapport à son tracé. Il faut noter que si l'on change la valeur de .ASPECT, les valeurs nécessaires pour amener la tortue aux bords supérieur et inférieur de l'écran sont modifiées.

[.DEPOSE] Requiert deux nombres comme arguments d'entrée. Le premier désigne une adresse et le deuxième une valeur. Dépose la valeur donnée à l'adresse donnée.
(voir chapitre 5)

DOS Requiert un argument d'entrée, qui est une commande (et non un primitif LOGO) sous forme de mot ou de liste. Transmet cette commande pour exécution. Accepte la plupart des commandes DOS.

Exemple :

```
DOS [ UNLOCK MONBOULOT.LOGO ]
```

.ESPACE Sort le nombre de NODES (quatre octets) actuellement disponibles. Pour obtenir une évaluation correcte, effectuer un recyclage auparavant.

Exemple :

```
.RECYCLE
AFFICHE .ESPACE
1734
```

.EXAMINE Requiert un nombre comme argument d'entrée. Sort la valeur contenue à l'adresse indiquée par l'argument.
(voir chapitre 5)

.PAT (mis pour Point d'Arrêt)
Sort de LOGO.

Pour retrouver LOGO, taper CTRL **Y** RETURN . Ce primitif est utilisé en cas de pépin système. Deux adresses utilisées :

"cold start" en $1BF9 que l'on peut utiliser quand on a quitté LOGO.

"warm start" en $1BFC, pour essayer de récupérer LOGO après un pépin système. Si l'on rentre dans LOGO via l'adresse "cold start", le contenu de l'espace de travail est perdu : cela a le même effet que de taper AUREVOIR. L'adresse "warm start" laisse les variables et les procédures intactes.

.RECYCLE Met en route un recyclage ; LOGO examine dans la mémoire tous les NODES (quatre octets) disponibles, soit parce qu'ils sont effectivement vacants, soit parce que l'information qui y est contenue n'est plus utilisée. Le recyclage s'effectue automatiquement quand c'est nécessaire, mais il peut être utile de prévoir un recyclage avant certaines procédures qu'on ne souhaite pas voir ralentir par un recyclage intempestif.

; Tout le texte écrit à la droite de ce caractère est ignoré lors de l'exécution de la ligne. Permet d'insérer un commentaire.

- ***Messages d'erreur***

: EST MAL PLACE DANS ...

Il est probable que le caractère ":" qui indique la valeur d'une variable a été recollé au mot qui le précède. Remettre l'espace séparateur au bon endroit.

Exemple :

AFFICHE:N au lieu de **AFFICHE :N**

Peut également provenir de la tentative d'utiliser une étiquette hors d'une procédure.

'ALORS' EST MAL PLACE

Probablement SI ... ALORS ... SINON ... n'est pas situé sur une même "ligne ordinateur". Effacez les espaces séparateurs en trop.

Exemple :

SI "MOT = "MIT

ALORS AFFICHE "IDEM

SINON AFFICHE "DIFFERENT

au lieu de

SI "MOT = "MIT ALORS AFFICHE ...

... "IDEM SINON AFFICHE "DIFFERENT

Si le texte n'arrive pas à tenir en une seule ligne (256 caractères) vous pouvez transformer en procédures les expressions qui suivent ALORS et SINON ou utiliser la formulation TESTE ... SIVRAI ... SIFAUX ... qui doit être inscrite en trois lignes distinctes.

A QUOI SERVENT LES PARENTHESES ?

Les parenthèses sont (probablement) inutiles, ou mal positionnées.

Exemple :

(AFFICHE ("A "B)

ARRET !

LIGNE ... AU NIVEAU ... DE ...

Interruption due à CTRL **G**

L'interruption se situe alors que *n* (nombre indiqué après NIVEAU) procédures sont en cours d'exécution. La ligne où se produit l'inter-

ruption est précisée, ainsi que la procédure où se trouve cette ligne. Le niveau 0 est celui d'une commande tapée directement au clavier. Le niveau 1 indique que l'instruction fautive est sur une ligne d'une procédure appelée au niveau 0. Le niveau 2 signifie que l'instruction fautive est sur une ligne d'une procédure appelée au niveau 1, etc...

... – – COMMANDE DOS INCONNUE

Vous avez utilisé un terme inadéquat après le primitif DOS.

S'IL TE PLAIT, EFFACE QUELQUECHOSE

L'espace de travail est saturé. Il convient soit de le sauvegarder sur fichier, et ne garder que les procédures réellement en cours d'utilisation, soit d'éliminer de l'espace de travail un certain nombre d'objets (procédures ou noms) devenus sans utilité.

– – ERREUR DISQUE

Etes-vous sûr d'avoir une disquette dans le lecteur ? Avez-vous vérifié que la disquette est bien une disquette de travail initialisée pour Apple ? La disquette est-elle dans le bon sens ?

... EST DEFINI

Indique que la procédure est enregistrée comme effectivement définie, quand on quitte le mode éditeur avec CTRL **C.**

... EST UN PRIMITIF LOGO

Vous essayez probablement d'utiliser comme titre de procédure un primitif LOGO. Choisissez un autre titre.

– – FICHIER PROTEGE

Le fichier que vous essayez de détruire ou de sauvegarder porte le même titre qu'un fichier protégé (qu'on peut ramener, mais non effacer). Changer de titre pour votre sauvegarde est sans doute une solution. Tout fichier marqué d'un astérisque dans le catalogue est protégé.

'FIN' : SEULEMENT DANS L'EDITEUR

Vous avez essayé d'utiliser FIN dans une ligne de procédure (vous vouliez probablement dire STOP), ou dans la dernière liste des lignes après DEFINIS.

ICI COMMENCE ...

Signale le début d'une procédure en mode TRACE. La valeur des arguments d'entrée est mentionnée.

ICI FINIT ...

Signale la fin d'une procédure en mode TRACE.

IL MANQUE QUELQUECHOSE APRES ...

Le primitif ou la procédure est suivi d'un nombre insuffisant d'arguments d'entrée : il n'y a pas assez d'information pour que l'instruction soit exécutée.

Exemple :

```
AFFICHE
IL MANQUE QUELQUECHOSE APRES AFFICHE
```

IL MANQUE QUELQUECHOSE AVANT ...

Typique d'une opération numérique ou d'une utilisation d'un prédicat en notation préfixée, alors qu'il s'agit de primitifs infixés.

Exemples :

```
TESTE = 4
IL MANQUE QUELQUECHOSE AVANT =
AFFICHE / 8
IL MANQUE QUELQUECHOSE AVANT /
```

IL MANQUE QUELQUECHOSE ENTRE ( )

Ceci provient d'une instruction située à l'intérieur des parenthèses, et qui n'a pas reçu un nombre suffisant d'arguments d'entrée ou d'une absence de parenthèse ouvrante.

Exemple :

```
LEVEPLUME (FIXEXY 40 )
IL MANQUE QUELQUECHOSE ENTRE ( )
```

IL N'Y A PAS DE PROCEDURE ...

Vous essayez d'imprimer ou d'effacer une procédure qui n'existe pas. Probablement, vous vous êtes trompé de nom, ou vous avez peut-être fait une faute d'orthographe. Essayez IMTS, pour revoir tous les titres des procédures actuellement disponibles dans votre espace de travail.

IL N'Y A PAS D'ETIQUETTE ...

Vous utilisez VA avec une étiquette qui ne figure nulle part dans la procédure. Vous pouvez éviter ce message en n'utilisant pas VA. Si vous tenez à VA, n'oubliez pas d'indiquer l'étiquette au début de la ligne à laquelle vous voulez rendre le contrôle.

IL N'Y A RIEN A GARDER

Vous essayez de copier sur disque (avec GARDE) votre espace de travail, mais celui-ci ne contient rien que LOGO puisse garder (procédures ou variables)

LA PARENTHESE VA TROP PROFOND

LA PROCEDURE VA TROP PROFOND

L'EVALUATION VA TROP PROFOND

SI-ALORS VA TROP PROFOND

Ceci est un oiseau rare ! Vous avez excédé les limites acceptables (plus de 200). Nous aimerions bien avoir une copie de votre procédure pour notre musée.

LE DISQUE EST PLEIN

Il n'y a plus assez d'espace sur la disquette pour sauvegarder votre fichier. Quelques suggestions pour vous tirer d'affaire :
- taper CATALOGUE, essayez de trouver un fichier dont vous n'avez plus besoin, effacez-le avec DETRUISFICHIER
- Chercher parmi vos disquettes de travail une disquette moins remplie.
- Tapez IMTS pour vous remettre en mémoire tous les noms de procédures actuellement disponiles dans l'espace de travail. Certaines ne sont peut-être pas indispensables. Effacez ces procédures ; avec un peu de chance cela diminuera la taille de votre fichier suffisamment pour qu'il puisse être sauvegardé sur cette disquette.

LE DISQUE EST PROTEGE

Vous avez essayé d'effectuer une sauvegarde en fichier sur un disque qui est protégé en écriture. C'est peut-être parce que la disquette actuellement dans le lecteur n'est pas votre disquette de travail personnelle.

LE NOMBRE NE CONVIENT PAS APRES ...

Le nombre que vous avez utilisé comme argument d'entrée est trop grand ou trop petit. Essayez une valeur différente.

LIGNE TROP LONGUE APRES DEFINIS

LIGNE TROP LONGUE APRES EXECUTE

LIGNE TROP LONGUE APRES REPETE

La liste ne doit pas dépasser 256 caractères. Pourquoi ne pas définir quelques sous-procédures pour effectuer une partie des instructions utilisées.

... N'AIME PAS RECEVOIR ...

L'argument d'entrée que vous avez fourni à ce primitif ou à cette procédure est d'une nature inadéquate. Vous avez mis un mot ou une liste à la place d'un nombre ou vice versa.

Exemple :

```
AVANCE "BIDULE
```

AVANCE N'AIME PAS RECEVOIR "BIDULE

Provient souvent d'une utilisation du nom (ici "BIDULE) au lieu de la chose (qui serait notée :BIDULE).
Ce message est également produit quand on essaie d'utiliser les primitifs d'extraction de caractères ou d'éléments de listes sur des objets vides.

Exemple :

```
AFFICHE SAUFPREMIER [ ]
```

SAUFPREMIER N'AIME PAS RECEVOIR [ ]

... N'AIME PAS RECEVOIR ... LA DONNEE DOIT ETRE 'VRAI' OU 'FAUX'

Variante du message précédent, quand on utilise un conditionnel ou un prédicat. Ces primitifs doivent recevoir comme argument d'entrée des expressions dont l'évaluation donne VRAI ou FAUX. Vous avez probablement oublié une partie du texte.

Exemple :

```
SI 2 ALORS 4
```

SI N'AIME PAS RECEVOIR 2

LA DONNEE DOIT ETRE 'VRAI' OU 'FAUX'

... N'A PAS ETE CREE

Il n'y a pas de chose correspondant au nom que vous avez donné. Cela se produit par exemple si dans une procédure vous utilisez un nom de paramètre qui n'a pas été mentionné sur la ligne de titre de cette procédure. (ou qui a été désigné par un nom différent). Ou bien vous utilisez un paramètre qui était local à une procédure.

Exemple :

```
POUR CARRE :COTE
REPETE 4 [ AVANCE :COTE DROITE 90 ]
FIN
AFFICHE :COTE
COTE N'A PAS ETE CREE
AFFICHE :C
C N'A PAS ETE CREE
```

... N'A PAS ETE DEFINI

Il n'y a pas de procédure portant le titre que vous venez de taper. C'est très souvent une faute de frappe (CQRRE au lieu de CARRE, par exemple), ou l'oubli d'un espace (AVANCE100 au lieu de AVANCE 100)

... N'A RIEN SORTI

Il manque un argument d'entrée pour un primitif, et cet argument n'est pas fourni par les instructions figurant sur le reste de la ligne.

Exemple :

```
AVANCE RECULE 50
RECULE N'A RIEN SORTI
```

Cela provient souvent d'une procédure qui effectue des calculs, mais ne *sort* pas le résultat pour le rendre utilisable.

Exemple :

```
POUR CARRE :N
AFFICHE :N * :N
FIN
AVANCE CARRE 4
```

CHAPITRE III

PRIMITIFS ET MESSAGES D'ERREUR

16

CARRE N'A RIEN SORTI

... NE PEUT ETRE UTILISE DANS L'EDITEUR

Encore un oiseau rare. En mode éditeur, vous avez tapé un texte qui, lorsque vous sortez de l'éditeur, interfère avec le reste du texte dans le tampon d'édition.

... NE PEUT ETRE UTILISE QU'APRES 'IMPRIME', 'EFFACE' OU 'EDITE'

Concerne les primitifs TOUT, TITRES, NOMS, PROCEDURES. Vous avez probablement utilisé AFFICHE au lieu de l'un des trois primitifs autorisés.

... NE S'EMPLOIE QUE DANS UNE PROCEDURE

Concerne les primitifs SORS, STOP, VA, que vous avez essayé d'utiliser dans une ligne d'instructions directement tapée au clavier.

... - - PAS DE FICHIER DE CE NOM.

Plusieurs causes possibles :
- faute de frappe dans le nom du fichier (exemple RAMENE "NUSIQUE)
- le fichier n'est pas sur la disquette actuellement dans le lecteur
- vous avez un problème avec le lecteur.

Essayez de taper CATALOGUE [RETURN] : la liste des fichiers disponibles doit vous permettre de vous tirer d'affaire dans les deux premiers cas. Si CATALOGUE produit l'affichage de la liste vide [ ], c'est un problème avec le lecteur.

PAUSE

Interruption due à [CTRL] **Z** ou au primitif PAUSE dans le cours d'un programme. Tapez **CONTINUE** [RETURN] pour poursuivre.

PLUS D'ESPACE DISPONIBLE !

- suivi de **LIGNE ... AU NIVEAU ... DE ...** : votre procédure produit une boucle folle, due à une récursion non terminale sans test d'arrêt
- message isolé : il y a trop de procédures dans votre espace de travail. Effacez-en quelques unes.

QUE DOIS-JE FAIRE AVEC...

Vous avez tapé au clavier une instruction demandant d'effectuer un calcul sans préciser à quoi doit servir le résultat (vous vouliez probablement le faire AFFICHER, mais LOGO a beau être très doué, il ne devine pas –encore– ce genre d'intention) .

Exemple :

```
3 + 4
QUE DOIS-JE FAIRE AVEC 7
```

QUE FAUT-IL FAIRE AVEC ...

Même message que précédemment, mais l'instruction est située à l'intérieur d'une procédure, et non tapée au clavier.

Exemple :

```
POUR CARRE :NOMBRE
:NOMBRE * :NOMBRE
FIN
CARRE 5
QUE FAUT-IL FAIRE AVEC 25, LIGNE
    :NOMBRE * :NOMBRE
AU NIVEAU 1 DE CARRE
```

'SINON' EST MAL PLACE

Probablement, SI ... ALORS ... SINON ... n'est pas situé sur une même "ligne ordinateur" (voir 'ALORS' EST MAL PLACE), ou SINON est utilisé hors contexte

Exemple :

```
AFFICHE 4 SINON AFFICHE "QUATRE
'SINON' EST MAL PLACE
```

TIENS ! FIN EN TROP

Vous àvez probablement tapé deux fois le primitif FIN dans l'éditeur. Cela n'empêche pas l'exécution de la procédure.

TIENS ! UN CROCHET FERMANT EN TROP

Il y a davantage de crochets fermants que de crochets ouvrants. Ceci n'empêche pas l'exécution des instructions.

TORTUE HORS ECRAN

En mode PAGE, la tortue devrait sortir de l'écran, si l'instruction était exécutée, aussi ne bouge-t-elle pas.

Exemple :

AVANCE 200

TORTUE HORS ECRAN

TRACE COMMENCE

Début du mode TRACE. Tous les appels de procédures ultérieurs seront exécutés en PAS A PAS.

TRACE TERMINE

Sortie du mode TRACE.

TROP DE PARAMETRES

Vous avez dépassé le nombre de paramètres autorisé (plus de 200) Ça ne devrait pas arriver très souvent.

TU ESSAIES DE DIVISER PAR ZERO

Dans l'utilisation de / , QUOTIENT ou RESTE, le deuxième argument d'entrée est nul. Prévoyez d'insérer un test sur la valeur de ce paramètre, pour écarter la division lorsqu'il devient nul.

UN MOMENT, MERCI ...

La procédure est en cours d'enregistrement après CTRL **C**

Une utilisation judicieuse des instructions .EXAMINE et .DEPOSE autorise toutes sortes de possibilités. Prenons par exemple le problème de l'animation en LOGO. Le LOGO de l'Apple II n'est pas conçu pour produire des effets d'animation. Le meilleur mouvement que l'on peut obtenir sur l'écran est celui du déplacement de la tortue. En effet, si vous voulez avoir un cercle qui se déplace sur l'écran, ce sera très long de dessiner et d'effacer maintes et maintes fois un cercle en le déplaçant petit à petit. Ce que vous pouvez faire, par contre, c'est changer la forme de la tortue elle-même, de manière qu'elle ressemble à un cercle. Vous aurez alors un cercle qui se déplace sur l'écran, simplement en déplaçant la tortue.

Le dessin de la tortue fait appel au mécanisme "SHAPE" (forme) de l'Apple, qui permet de spécifier des formes par des tables de vecteurs de 2 ou 3 bits : vous en trouverez la description dans ie manuel de référence de l'Applesoft. Vous pouvez faire vos propres formes LOGO et les faire mouvoir sur l'écran à la place de la tortue. Pour établir votre propre table de formes, mettez l'emplacement du premier élément de la table à l'adresse USHAPE (voir le paragraphe "adresses mémoire utiles" pour l'explication des adresses LOGO.) L'adresse SSIZE contient le code de la taille sur un octet que vous pouvez changer. La valeur par défaut, 1, est la meilleure pour la tortue normale, toutefois des valeurs supérieures ou égales à 2 rendent souvent plus visibles les formes définies par l'utilisateur. A la différence du mécanisme général d'Apple pour les formes, l'interface LOGO permet aux formes définies par l'usager d'être représentées seulement à des caps de 0 , 90 , 180 , 270 degrés. Le cap de la forme représentée est déterminé par le quadrant que regarde la tortue et peut être changé en tournant la tortue avec les commandes usuelles GAUCHE et DROITE. Le format des tableaux de formes est celui qui est décrit dans le manuel de référence de l'Applesoft, à ceci près qu'il faut omettre l'information de tête dans le tableau de formes. Commencez chaque tableau de forme directement avec les vecteurs. Terminez normalement.

Vous pouvez établir un tableau de formes à la main, et utiliser .DEPOSE pour le stocker dans l'espace LOGO réservé pour le code de l'utilisateur, et ensuite initialiser USHAPE et SSIZE. Remarquez que vous pouvez créer plus d'une forme et changer ensuite en changeant USHAPE.

- ***L'éditeur de formes LOGO***

L'établissement de tableaux de formes est un travail fastidieux. Un des programmes contenus sur la disquette LOGO est un éditeur de formes. C'est un programme LOGO, qui vous permet de créer une forme, simplement en la dessinant sur l'écran. Ce programme assemble alors automatiquement la forme en un tableau de formes.

CHAPITRE IV

CHANGER LA FORME DE LA TORTUE

L'éditeur de formes a été écrit par Henry Minsky. Remarquez que l'éditeur de formes est lui-même un ensemble de procédures LOGO qui utilisent .EXAMINE et .DEPOSE selon le schéma indiqué ci-dessus. Vous pouvez lire ces procédures et les utiliser comme guide pour écrire des fonctions similaires.

Pour utiliser l'éditeur de formes, ramenez le fichier EDIT.FORME Le fichier contient un éditeur de formes en temps réel, et permet de changer la forme de la tortue et sa taille.

Donnez un nom à la forme à créer. Par exemple

RONDS

Taper à nouveau la commande CREEFORME permet de définir une nouvelle forme.

Vous ne pouvez éditer les formes prédéfinies. Si vous souhaitez effacer toutes les formes et tout recommencer, taper DEBUT. Les commandes suivantes, semblables à celles de l'éditeur, vous permettent de construire les formes :

U LEVEPLUME
D POSEPLUME

CTRL **P** remonte (et dessine une ligne verticale si la plume est en position baissée)
CTRL **N** descend (et dessine une ligne verticale si la plume est en position basse)
← et → déplacent dans la direction de la flèche (en dessinant une ligne horizontale si la plume est baissée)
CTRL **C** sort de l'éditeur de forme et définit effectivement la forme
CTRL **G** sort de l'éditeur de forme sans définir la forme de manière définitve. Vous pouvez utiliser cette forme dans l'état où elle est pour en "habiller" la tortue, mais quand vous reprendrez la définition d'une forme, celle que vous venez de quitter par CTRL G sera perdue. Cette commande s'utilise principalement quand on veut recommencer une forme.

ESC efface les quelques commandes précédentes. (Ce qui se produit exactement est l'effacement de l'octet précédent dans le tableau de formes, de sorte que le dernier segment ou les deux ou trois segments précédents sont effacés.)

1...9 Taper un chiffre modifie la taille de la forme. Taper 3 équivaut à exécuter TAILLE 3.

Une fois que vous avez défini une forme pour la tortue, (ROND dans cet exemple), vous pouvez faire prendre cette forme à la tortue : pour cela tapez FIXEFORME :ROND

CHANGER LA FORME DE LA TORTUE

Vous pouvez changer la taille des formes en utilisant la procédure TAILLE (TAILLE 1 par défaut). La commande FIXEFORME nécessite un argument d'entrée et fait prendre à la tortue la forme correspondante désignée par cet argument. Cette commande cache d'abord la tortue, puis la montre. Si vous voulez redonner à la tortue sa forme triangulaire initiale, tapez FIXEFORME 0. La procédure interne que FIXEFORME utilise est .FORME . Elle aussi nécessite la donnée d'un argument d'entrée, mais ne commence pas par cacher la tortue. Ceci est utile car LOGO efface la tortue tout simplement en dessinant la forme de la tortue en mode inverse. Cela implique que si vous donnez une première forme à la tortue, puis la forme vide, et qu'ensuite vous cachez la tortue et la déplacez, l'image initiale de la tortue réapparaîtra sur l'écran, car cacher la forme vide n'efface rien du tout.
.FORME 1 donne la forme vide à la tortue. La procédure suivante utilise cette méthode pour faire apparaître une forme à des endroits aléatoires sur l'écran.

```
POUR COUCOU :ROND
PLEINECRAN .FORME :ROND
MONTRETORTUE
.FORME 1
CACHETORTUE LEVEPLUME
FIXEXY (– 120 + HASARD 240) (– 110 + HASARD 220)
COUCOU :ROND
FIN
```

- ***Sauvegarder les formes***

Pour sauvegarder sur disquettes toutes les formes que vous avez définies, utilisez la procédure GARDEFORMES, Il faut un argument, qui est le nom du fichier. Si vous tapez GARDEFORMES"RONDS cela créera deux fichiers : RONDS.FORMES contenant le tableau de formes, et RONDS.AUX.LOGO, contenant les noms des formes et quelques autres procédures. En plus de FIXEFORME, .FORME et TAILLE, ce fichier auxilliaire contient une procédure appelée INITFORMES. Si vous tapez RAMENE"RONDS.AUX et exécutez ensuite la procédure INITFORMES , le tableau des formes sera automatiquement ramené.
Sur la disquette utilitaire se trouve un programme de démonstration appelé FUSEE . Tapez RAMENE "FUSEE, puis DEBUT . La procédure FUSEE charge le fichier FUSEE.AUX et exécute INITFORMES . Tapez IM FUSEE pour voir comment cela marche. Pour l'exécuter une nouvelle fois sans avoir à recharger le fichier, il suffit de taper TABLEAU.

CHAPITRE V

L'ASSEMBLEUR EDI-LOGO

Le système EDI-LOGO pour l'Apple a été conçu pour être à la fois puissant et facile à utiliser. Dans la plupart des cas, les primitifs décrits au chapitre 3 devraient suffire pour concevoir et exécuter les programmes LOGO. Toutefois, il y a des situations dans lesquelles il est souhaitable d'accroître les possibilités du système, en accédant directement au langage machine.

Attention : Ce chapitre ne sera utile et compréhensible qu'aux personnes déjà familiarisées avec l'assembleur de l'Apple II.

L'assembleur LOGO possède un certain nombre de "points d'ancrage" qui permettent d'accéder directement à la mémoire ou d'interfacer des programmes LOGO et des programmes écrits en langage d'assemblage 6502. La disquette utilitaire contient un assembleur langage machine 6502 qui aide à gérer ces "points d'ancrage". Un autre "point d'ancrage" implanté dans LOGO vous permet de créer des effets d'animation simples, en donnant une nouvelle forme à la tortue LOGO. Un autre "point d'ancrage" vous permet de modifier le fonctionnement de l'éditeur LOGO de manière à pouvoir l'utiliser en éditeur ordinaire plutôt qu'en éditeur de procédures, et accéder aux fichiers sur disquette pour des applications non standard (Exemple : ne sauvegarder sur disquette qu'une procédure, alors que l'espace de travail en contient plusieurs).

• *.EXAMINE et .DEPOSE*

Ces deux commandes sont à peu de chose près les routines connues PEEK et POKE de l'Apple. La principale différence est que les adresses doivent toujours être spécifiées par des entiers positifs en LOGO, alors que pour PEEK et POKE il convient d'utiliser des nombres négatifs pour les adresses situées au-delà de 32 K. .EXAMINE requiert une adresse en argument d'entrée, et sort la valeur (numérique) contenue dans cette adresse.

.DEPOSE requiert deux arguments d'entrée : une adresse et une valeur numérique. Ce primitif charge la valeur donnée dans le registre spécifié par l'adresse. Ces deux commandes sont utiles pour communiquer avec des dispositifs particuliers d'entrée/sortie, par exemple quand l'utilisation de PERIPH ne suffit pas. Bien entendu, une utilisation arbitraire de .DEPOSE n'importe où dans la mémoire peut provoquer une mise hors service du système LOGO ou d'autres effets déplaisants. Il est à remarquer que les adresses manipulées par ces commandes sont des nombres LOGO ordinaires, c'est-à-dire exprimés en base décimale, même si l'on a plutôt l'habitude de considérer une adresse Apple par sa représentation hexadécimale.

Pour beaucoup d'applications, il serait utile d'écrire une routine de conversion de l'hexadécimal en décimal. De cette façon, vous pourriez taper

```
.EXAMINE HEX "9E
```

au lieu de

```
.EXAMINE 158
```

Nous adoptons dans ce chapitre la convention suivante : un nombre en hexadécimal est précédé du symbole $. Ainsi, $9E est le décimal 158.

Lorsque LOGO est chargé, les adresses ROM sont déplacées dans l'extension mémoire, qui contient aussi une partie du système LOGO. Les anciennes adresses ROM ne sont donc plus disponibles (masquées par l'extension). C'est pourquoi utiliser .EXAMINE (ou .DEPOSE) pose certains problèmes pour les adresses de l'extension. La carte 16 K est implantée de $CØ8Ø à $CØ8F (49280 à 49295).

• *Comment écrire vos propres routines en langage machine*

Vous pouvez interfacer vos routines personnelles et LOGO en utilisant le primitif .APPEL, qui requiert deux arguments d'entrée. Le premier argument est l'adresse de la routine, et le second, un entier que la routine peut examiner. La routine peut, ou non, sortir un résultat (nombre entier). Le primitif .APPEL requiert *toujours* deux arguments, que la routine de l'utilisateur ait effectivement besoin de prendre en compte le second ou non.

.APPEL donne le contrôle à la routine implantée à l'adresse spécifiée par le premier argument. Naturellement, avant ceci, vous devez écrire la routine appropriée et l'implanter à l'adresse adéquate. La mémoire disponible pour vos routines en code machine s'étend de $99AØ à $9AAØ (39328 à 39584). Vous pouvez assembler manuellement et stocker la routine en utilisant .DEPOSE, mais vous trouverez plus commode d'utiliser l'assembleur LOGO décrit ci-dessous.

Quand l'exécution de votre routine commence, les mémoires NARG1 et NARG1 + 1 en page zéro contiennent le premier argument de .APPEL, qui est justement l'adresse de la routine elle-même. NARG1 + 2 et NARG1 + 3 contiennent obligatoirement zéro à l'instant où la routine est appelée. La routine peut utiliser temporairement les mémoires NARG1 à ANSN4 + 3 sans se soucier de les restaurer au moment du retour. Ces mémoires de stockage sont volatiles, c'est-à-dire que LOGO peut changer ces mémoires entre des appels successifs de votre routine. Les mémoires de USERPZ à $FF ne sont pas utilisées par LOGO et vous pouvez donc les utiliser pour garder les valeurs constantes de vos routines en page zéro. Les adresses en page zéro sont dans le fichier ADRESSES.

NARG2 à NARG2 + 3 contiennent le second argument de .APPEL sous forme de quatre octets complément à 2. Ainsi, .APPEL 39328 3 donnerait l'organisation de mémoire suivante :

NARG2	NARG2 + 1	NARG2 + 2	NARG2 + 3
3	Ø	Ø	Ø
NARG1	NARG1 + 1	NARG1 + 2	NARG1 + 3
$AØ	$99	Ø	Ø

Si on substitue (– 1) à la place de 3, les registres NARG2 à NARG2 + 3 contiennent $FF

Pour sortir un entier, stocker cet entier (en utilisant le format ci-dessus) dans les quatre mémoires NARG2 à NARG2 + 3 et sauter en OTPFX2. Si le nombre est stocké dans quelque autre ensemble de quatre variables page-zéro consécutives (tel que NARG1), charger le registre Y avec l'adresse et sauter en OTPFIX.

Pour avoir le mot LOGO ''VRAI, sauter en OTPTRU. De même, en sautant en OTPFLS, vous provoquerez un affichage du mot ''FAUX par votre routine.. Si vous ne voulez pas sortir de résultat, finissez votre routine avec une instruction RTS.

Voici un exemple qui lit l'état du port de la cassette (en adressant l'emplacement input cassette: $C∅6∅) et donne la réponse ''VRAI ou ''FAUX selon qu'il y a du son ou non. Le code est ici écrit en assembleur 6502 standard. Pour l'utiliser, il faudra l'assembler à la main, et implanter les instructions en mémoire (voir plus loin).

```
            ORG $99A∅
CIN         EQU $C∅6∅
OTPTRU      EQU (voir Adresses.LOGO)
OTPFLS      EQU (voir Adresses.LOGO)
LISTEN:     LDA CIN
            BMI ON      ;voir manuel de référence de l'Apple II
            JMP OTPFLS
ON:         JMP OTPTRU
            END
```

Vous pouvez maintenant initialiser la variable LISTEN à l'adresse de l'étiquette LISTEN et exécuter ce nouveau ''primitif'' en tapant:
.APPEL :LISTEN ∅
(Notez qu'on a besoin d'un input, même si on n'en tient pas compte). Ceci marchera exactement comme un primitif normal ou une procédure normale.

AFFICHE .APPEL :LISTEN ∅
affichera VRAI ou FAUX.

Lorsqu'une routine en langage machine a rencontré une condition d'erreur qui perturberait le retour à la procédure LOGO appelante, elle saute à PPTTP, qui a le même effet que le primitif LOGO NIVEAUSUP.

• *L'assembleur LOGO*

L'assembleur LOGO est un assembleur 6502 qui est écrit en langage LOGO. Ce programme a été conçu et écrit par Leigh Klotz. Cet assembleur vous est fourni sur la disquette utilitaire, dans le fichier ASSEMBLEUR. (Le programme ASSEMBLEUR à son tour lit des données stockées sur cette même disquette dans les fichiers auxi-

liaires AMODES et OPCODES). Pour utiliser l'assembleur, ramenez simplement ce fichier comme vous le feriez pour n'importe quel fichier LOGO normal, et exécutez ensuite la procédure DEBUT:

```
RAMENE "ASSEMBLEUR          RETURN
DEBUT
```

Pour assembler une routine, vous écrivez la routine comme une procédure LOGO au moyen de l'éditeur LOGO. Ainsi, l'exemple de la cassette que nous venons de décrire serait écrit de la manière suivante :

```
POUR CASSETTE.CODE
[ CREE "CIN $ "C060 ]
[ CREE "OTPFLS 7188 ]
[ CREE "OTPTRU 7194 ]
LISTEN: LDA CIN
BMI ON
JMP OTPFLS
ON: JMP OTPTRU
FIN
```

Remarquez les différences de syntaxe entre l'input accepté par l'assembleur LOGO et l'assembleur standard 6502. La syntaxe du code pour l'assembleur est expliquée un peu plus loin.

Une fois définie la procédure, on assemble en tapant

```
ASSEMBLE "CASSETTE.CODE
```

Cela assemble les instructions, et les implante par défaut en ($ 99AØ). Ainsi, chaque étiquette du code, comme LISTEN dans l'exemple précédent, est définie comme une variable LOGO. Vous pouvez maintenant appeler la routine par

```
.APPEL :LISTEN Ø
```

• *Utiliser l'assembleur pour écrire des routines d'entrée/sortie*

Bien qu'il soit possible d'utiliser les primitifs .EXAMINE et .DEPOSE pour la plupart des dispositifs périphériques, des routines en langage machine sont nécessaires pour d'autres. Si le périphé-

rique est conçu en ''driver'', vous pouvez utiliser le primitif PERIPH. Ce primitif requiert un nombre de 1 à 7 comme argument d'entrée (numéro du connecteur périphérique), et toutes les routines d'entrée-sortie seront alors dirigées vers le connecteur périphérique spécifié. Il se peut que certaines unités périphériques aient besoin de routines particulières pour gérer leur entrée-sortie. Si vous donnez à PERIPH un argument supérieur à 8, celui-ci sera interprété comme l'adresse d'une routine qui devra être appelée à la place de la routine normale LOGO d'entrée-sortie.

De nombreux périphériques utilisent une technique de ''handshaking'' qui permet à l'ordinateur de ne pas essayer d'envoyer (ou recevoir) trop rapidement des données via ce périphérique. Le programme ci-dessous donne un exemple pour une telle unité.

Nous supposons que STATUS est l'emplacement mémoire input-output de la carte périphérique qui indique l'état de l'unité. Dans ce cas, le bit 7 est à 1 si l'unité est prête à recevoir un caractère. DATA est l'adresse où les octets sont envoyés pour être mémorisés. Une fois que vous avez assemblé cette routine, vous pouvez accéder au périphérique en exécutant PERIPH :TYOWAIT

```
POUR CODE
[ CREE "STATUS      (adresse) ]
[ CREE "DATA        (adresse) ]
TYOWAIT: LDX STATUS
BPL TYOWAIT
STA DATA
RTS
FIN
```

Cette routine de sortie de caractères, qui est censée être appelée via PERIPH, doit trouver dans le registre A l'octet à sortir.

Voici un autre exemple de routine de sortie. Celle-ci transforme l'affichage des caractères ! en les remplaçant par des espaces:

```
POUR IOCODE
XCLOUT: CMP # "!
BNE OUTCHAR
LDA # 32
OUTCHAR: UMP COUT
FIN
```

- ***Syntaxe d'entrée de l'Assembleur***

Pour tirer parti de certains aspects du langage LOGO, l'Assembleur LOGO utilise un format légèrement différent de la plupart des assembleurs. Chaque programme en Assembleur est mémorisé comme une procédure LOGO, bien que cette procédure ne puisse être exécutée directement. Les paragraphes suivants décrivent la syntaxe d'assemblage. Il sera plus aisé de comprendre, au vu des exemples, comment écrire des programmes en assembleur via LOGO. Les étiquettes à l'intérieur d'un programme sont repérées par un suffixe "**:**" (exemple: ETIQUETTE:) Les références aux cases mémoire en page zéro qui ne sont pas indirect indexé (LDA (FOO , x)) ou indexé indirect (LDA (FOO), Y) doivent avoir un point d'exclamation avant l'étiquette ou l'expression qui est en page zéro. Si vous oubliez le point d'exclamation, l'instruction sera comprise comme une référence absolue et occupera un octet de plus. Il doit y avoir un espace avant chaque "!" (indiquant la référence à la page zéro) et chaque **#** (indiquant le mode immédiat) et après chaque étiquette ou référence à une étiquette. L'opérande d'une instruction peut être un mot (une référence à une étiquette) , un nombre, une liste ou un mot d'une seule lettre précédé de " **:** " dans ce dernier cas, l'opérande est le code ASCII de la lettre.
Tout élément intérieur à une liste est évalué comme une expression LOGO. Si la liste est la première chose de la ligne, il ne lui est pas permis de sortir une valeur, et elle est alors seulement évaluée comme une étiquette. Si c'est un opérande (qui suit le nom d'une instruction), il est prévu de sortir quelque chose. Ainsi, les expressions arithmétiques telles que :FOO + 3 , où FOO est une étiquette ou un symbole LOGO normal, peuvent être employées à condition d'être écrites entre crochets. Bien sûr, les références aux valeurs des étiquettes entre crochets seront précédées de "**:**", et les espaces ont leur signification habituelle. Toutes les étiquettes sont des variables LOGO. DOT est une variable LOGO dont la valeur représente l'adresse de la case mémoire en cours d'assemblage.

Les procédures HI8 et LO8 qui sortent respectivement les huit bits de poids fort et les huit bits de poids faible d'un nombre sont souvent utiles à l'intérieur de listes. Utilisez-les comme suit :

```
LDA # [ LO8 :SOURCE ]
STA ! DEST
LDA # [ HI8 :SOURCE ]
STA ! [ :DEST + 1 ]
```

La procédure $ transforme un argument donné en entrée sous sa représentation hexadécimale en un nombre décimal. De plus, les nombres hexa peuvent être admis dans des programmes en insérant un appel à $ à l'intérieur d'une liste. Utiliser le primitif CREE pour donner des valeurs aux étiquettes.

Si vous utilisez souvent des nombres en représentation octale ou binaire vous aurez intérêt à changer la valeur de la variable LOGO $BASE. C'est la base qui est utilisée par la procédure $ (en fait par $1, elle-même appelée par $). La changer en 2 vous donnera du binaire, etc... Vous pouvez faire ceci à l'intérieur d'un programme en assembleur avec

```
[ CREE "$BASE 2 ]
```

Vous pouvez assembler des octets quelconques pour faire un code résultant en plaçant le nombre en début de ligne (ou, si elle existe, après l'étiquette).

Voici un programme simple dans la syntaxe normale de l'assembleur

```
ORG    EQU $99A0
NARG2 EQU $9E
BELL   EQU $1C40
PASS:  LDA NARG2
       CMP # $04
       BEQ YES
       RTS
YES:   JUMP BELL
END
```

et la syntaxe de l'assembleur LOGO

```
POUR CODE
PASS: LDA ! NARG2
CMP # 4
BEQ YES
RTS
YES: JMP BELL
FIN
```

; ! indique la page zéro. Attention à l'espace après le !

Pour assembler ce programme, chargez l'assembleur, tapez DEBUT puis

```
RAMENE "ADRESSES
```

Ce qui suit assemblera le programme ci-dessus à l'adresse $99AØ (origine par défaut) ; pour assembler à partir d'une autre adresse, modifier la valeur de la variable ORG. Tapez

ASSEMBLE "CODE

Ceci produira un affichage sur l'écran du code engendré (en décimal) et l'implantera en mémoire. Les étiquettes sont utilisables comme des symboles LOGO avec .APPEL, .DEPOSE, .EXAMINE
Pour faire exécuter la routine précédente, tapez :

.APPEL :PASS 4

qui provoque un bip, et

.APPEL :PASS autre chose que 4

pour n'avoir pas de son.

Si vous essayez d'assembler de longs programmes, il se peut que vous dépassiez la capacité mémoire. Une façon d'obtenir plus de place mémoire consiste à charger seulement les instructions que votre programme utilise. Après avoir rechargé LOGO, ramenez le fichier instructions OPCODES, qui est fourni sur la disquette utilitaire, et effacez les instructions (en utilisant par exemple EFNOM "BIT) dont vous n'avez pas besoin. Puis recopiez le contenu dans un nouveau fichier instructions OPCODES. Bien sûr, vous ne devez pas faire cette copie sur le disque original ; faites ces manipulations sur des disquettes supplémentaires.

- ***Comment sauver sur disque vos routines assemblées***

Avec le primitif DOS, vous pouvez sauver le code machine réel que l'assembleur engendre. Ce qui suit sauvera vos routines assemblées, dans un fichier ROUTINES (par exemple) :

DOS [BSAVE ROUTINES.BIN, A$99AØ,L$1ØØ]

Et pour charger les routines, tapez

DOS [BLOAD ROUTINES.BIN]

BIN est l'abréviation de BINAIRE, et devrait vous aider à vous souvenir que le fichier est un fichier langage-machine. Retenez bien qu'en plus de sauver le code machine réel, vous devrez sauvegarder les variables LOGO qui définissent les adresses utilisées par .APPEL. Une façon d'opérer consiste à taper EDITE NOMS, à sortir de l'éditeur par CTRL **G** , puis à exécuter EFFACE TOUT Repassez en mode édition et éditez les définitions pour ne conserver

que celles qui vous intéressent. Sortez alors de l'éditeur par CTRL **C** . Gardez le fichier en tapant GARDE "ROUTINES. Si vous voulez recharger votre routine, tapez

```
RAMENE "ROUTINES
DOS [ BLOAD ROUTINES.BIN ]
```

- ***Exemple : génération musicale***

Ce paragraphe présente un exemple d'application d'une extension de LOGO en langage assembleur. Bien que cette version de LOGO ne possède pas de primitifs pour jouer de la musique, vous pouvez utiliser le primitif .APPEL pour "interfacer" une routine langage machine destinée à produire une note pour le haut-parleur de l'Apple II. Celui-ci émet un son bref chaque fois que son adresse est évoquée (cette adresse est $C030 , ou 49200). Essayez de relire à plusieurs reprises cette adresse en utilisant .EXAMINE et vous obtiendrez un bruit.

Par exemple :

```
POUR BIP
REPETE 100 [ CREE "B .EXAMINE 49200 ]
```

Pour produire des notes, un programme doit examiner cette adresse plusieurs fois par seconde. Le nombre de "clics" produits par seconde est appelé la fréquence. Pour que les notes soient régulièrement espacées, le rapport entre les fréquences successives doit être constant, c'est-à-dire que les fréquences doivent être en progression géométrique. Dans la musique occidentale qui comporte douze notes par octave, cette constante doit être telle que la fréquence double après douze notes. Le rapport entre deux notes est donc la racine douzième de deux, soit approximativement 1,05946.

Le concept de période est intimement lié à celui de fréquence. La période d'une note est tout simplement l'inverse de sa fréquence. Connaissant la période, on peut jouer la note correspondante en engendrant une succession rapide de clics suivie d'un temps de silence jusqu'à la fin de la période. Pour ce faire, le programme doit être écrit en langage machine, compte tenu de la vitesse d'exécution requise.

Pour faire de la musique en LOGO, il est nécessaire d'écrire quelques procédures. Nous aurons une procédure appelée JOUE, qui requiert en arguments d'entrée une liste de hauteurs et une liste de durées correspondantes. Les hauteurs seront des entiers spécifiant le nombre de pas chromatiques au dessus ou au dessous d'une note

centrale. Les durées devront être les intervalles de temps durant lesquels la note doit être jouée, 1 étant la durée la plus courte, 2 correspondant à un intervalle deux fois plus long, et ainsi de suite.

```
POUR JOUE :HAUTEURS :DUREES
SI :HAUTEURS = [ ] STOP
SI :DUREES = [ ] STOP
JOUE.NOTE (PREMIER :HAUTEURS ) ...
                    ... (PREMIER :DUREES )
JOUE (SAUFPREMIER :HAUTEURS ) ...
                    ... ( SAUFPREMIER :DUREES )
FIN
```

Même si nous ne sommes pas certains de la façon dont seront jouées les notes, nous pouvons être sûrs que JOUE.NOTE joue réellement une note (connaissant la hauteur de la note et sa durée), parce que tel est notre but !

Puisque les notes sont jouées par un programme en langage machine demandant la période de la note, nous devons trouver un moyen d'associer les périodes aux notes telles qu'elles sont représentées dans JOUE. Une recherche en table est une assez bonne solution. Pour chaque note, il y a un point d'entrée dans la table qui contient la période. Nous construisons notre table comme des mots LOGO, les périodes étant les choses qui leur sont associées. Nous choisissons un nom arbitraire pour cette table, la représentation de chaque note sera ce nom suivi du numéro de la note. Donc si nous appelons la table "#", la période de la note numéro 3 sera la variable LOGO dénommée "# 3". Nous donnerons une valeur particulière pour le silence : nous la notons "# S". Ainsi S produit un silence dans la procédure JOUE. De plus, il serait utile de pouvoir spécifier les notes au-dessus ou au-dessous de l'octave de référence par une notation appropriée. Nous avons choisi de postfixer par + ou − les différentes octaves. Si l'argument de JOUE est 4, cela signifie qu'il s'agit de la quatrième hauteur par rapport au ton central ,4+ indiquant qu'il s'agit du quatrième ton dans l'octave supérieure, et 4− signifiant le quatrième ton dans l'octave inférieure. Vous pouvez réfléchir à une notation plus appropriée dans le cas d'une extension à plus de trois octaves.

La procédure CREE.NOTES suivante associe à chaque note sa période. Les arguments sont le numéro de la plus haute octave et la période de la plus haute note de cette octave :

```
POUR CREE.NOTES :PERIODE
CREE.OCTAVE 11 "+ :PERIODE
```

```
CREE.OCTAVE 11 " :PERIODE * 2
CREE.OCTAVE 11 "– :PERIODE * 4
CREE " # S 16384
FIN
POUR CREE.OCTAVE :HAUTEUR :OCTIND :PERIODE
SI :HAUTEUR = Ø STOP
CREE (MOT " # :HAUTEUR :OCTIND ) :PERIODE
CREE.OCTAVE :HAUTEUR – 1 :OCTIND ...
                              ... :PERIODE * 1.Ø5946
FIN
```

Tout ce qui précède correspond au maximum de ce que nous pouvions faire en LOGO sans nous préoccuper des implémentations spécifiques des routines de génération musicale. Ces routines en langage machine engendrent une note d'une période donnée pour une certaine durée. Comme nous l'avons signalé plus haut, on engendre un son sur le haut-parleur de l'Apple II en produisant un "clic" suffisamment souvent et suffisamment longtemps pour permettre la sortie d'un son.

Le cœur de notre routine en langage machine va être une sous-routine CLIC, voir plus loin. Cette routine est appelée à plusieurs reprises avec la période, codée sur 16 bits dans les registres PER.H et PER.L ; elle recopie ailleurs les valeurs de ces registres pour pouvoir les réutiliser à un nouvel appel de CLIC. Une façon d'agir sur la durée des notes serait d'appeler CLIC un certain nombre de fois, comme par exemple l'appeler deux fois plus afin d'obtenir une note deux fois plus longue. Ça marche si on n'a besoin que d'une seule période (note). Hélas, la routine CLIC, par sa nature, s'exécute en un temps différent pour chaque période, c'est-à-dire chaque note. Donc la routine qui joue des notes doit convertir la durée voulue en nombre de clics à engendrer.

Si nous choisissons une note de base, et si nous étalonnons la durée des autres notes en fonction de celle-ci, notre problème sera résolu. Le nombre d'appels à la routine CLIC pour une durée et une période données est :
:DUREE * (:BASE.PERIODE / :PERIODE). Cette graduation est appelée normalisation. Comme il est plus facile de multiplier et de diviser en langage LOGO qu'en langage machine, nous calculerons le facteur d'échelle en LOGO. Le programme machine appellera autant de fois CLIC que ce nombre passé en argument,

Ci-dessous le programme en langage machine pour jouer les notes :

```
POUR MCODE
[ CREE "SPKR $ "C030 ]
[ CREE "DUR.L :NARG2 ]
[ CREE "DUR.H :NARG2 + 1 ]
[ CREE "PER.L :NARG2 + 2 ]
[ CREE "PER.H :NARG2 + 3 ]
[ CREE "COMPTE.L :USERPZ ]
[ CREE "COMPTE.H :USERPZ + 1 ]
TON: LDA ! DUR.L
     ORA ! DUR.H
     BEQ EXIT ; une durée de 0 signifie pas de note
     LDA ! DUR.L
     SEC
     SBC # 1
     STA ! DUR.L
     LDA  ! DUR.H
     SBC # 0
     STA ! DUR.H
     JSR CLIC
     JMP TON
EXIT: RTS
CLIC: LDA SPKR
     LDA ! PER.L
     STA ! COMPTE.L
     LDA ! PER.H
     STA ! COMPTE.H
```

```
PDLOOP: LDA ! COMPTE.L
        ORA ! COMPTE.H
        BEQ EXIT
        LDA ! COMPTE.L
        SEC
        SBC # 1
        STA ! COMPTE.L
        LDA ! COMPTE.H
        SBC # 0
        STA ! COMPTE.H
        JMP PDLOOP
FIN
```

Pour assembler ce programme, ne conservez que les OPCODES et ADRESSES indispensables, sinon vous n'aurez pas assez d'espace.

Les boucles qui constituent le corps des deux sous-routines CLIC et TON ont une propriété intéressante : toutes les itérations s'exécutent en des temps rigoureusement égaux. Certaines méthodes d'écriture de ces boucles feraient que l'exécution est plus ou moins rapide si DUR.H (ou PER.H) est nul. Ces méthodes auraient pour effet qu'une durée de 400 clics ne serait pas deux fois plus longue qu'une durée de 200 clics. Remarquons que la routine CLIC contient plusieurs instructions qui sont en dehors de la boucle, et donc sont exécutées une seule fois pour une période donnée. Le temps d'exécution des boucles a une incidence sur la note produite. Ceci est équivalent à ajouter une petite quantité pour chaque période. Ce facteur est la constante FUDGE.

Nous avons besoin de certaines méthodes pour interfacer des routines en langage machine avec des procédures en langage LOGO. Le primitif .APPEL fournit un moyen de le faire, mais permet de transmettre un unique argument. Or nous avons besoin de transmettre à la fois la période (PER.H et PER.L) et la durée (DUR.H et DUR.L). Un examen attentif montrera que cela fait un total de 32 bits, et que cela correspond au nombre de bits que .APPEL peut transmettre à la routine en langage machine. Si nous organisons les emplacements

mémoire de telle sorte que DUR.L / PER.L soient les deux octets de poids faible de l'argument de .APPEL et DUR.H / PER.H les deux octets de poids fort, la procédure suivante donnera les deux arguments nécessaires à l'appel d'une routine en langage machine.

```
POUR .APPEL.2 :ADR :INPUT1 :INPUT2
.APPEL :ADR (ENTIER :INPUT2) + ...
                    ... (ENTIER :INPUT1) * 65536
FIN
```

Remarquez l'utilisation du primitif ENTIER. S'il n'était pas utilisé, le résultat de la multiplication ne serait pas un multiple de 65536, puisque les périodes ne seraient pas des nombres entiers, et cela aurait des répercussions sur la durée. La procédure JOUE.NOTE utilise cette procédure et le facteur de normalisation mentionné plus haut. .APPEL.2

```
POUR JOUE.NOTE :PERIODE :DUREE
CREE "PERIODE CHOSE MOT " # :PERIODE
.APPEL.2 :TON :PERIODE - :FUDGE ...
          ... (:DUREE * (PERIODE.BASE / :PERIODE )
FIN
```

Remarque : comme elle est, la procédure JOUE ne comprend pas les silences. Comme il serait difficile d'écrire une procédure LOGO qui provoque un retard d'une durée convenable, imaginez un moyen d'écrire une routine en langage machine qui ferait ceci sans produire de son. Cela sera possible avec une basse fréquence (période longue). Vous pouvez mettre le nombre représentant cette fréquence dans la variable LOGO S ; ensuite vous pourrez utiliser la note S comme argument pour jouer un silence.

- ***Le programme de démonstration de musique***

Il y a trois fichiers relatifs à la musique dans la disquette des utilitaires. Deux sont des fichiers LOGO appelés MUSIC.SRC et MUSIQUE, et l'autre est un fichier appelé MUSIC.BIN, qui contient un programme en langage machine. Pour essayer la démonstration de MUSIQUE, tapez RAMENE "MUSIQUE. Toutes les procédures montrées ici sont utilisées.

- ***Adresses mémoire utiles***

Vous trouverez ici une brève description d'adresses du programme LOGO qui sont utilisables comme points d'ancrage pour modifier LOGO grâce à .EXAMINE et .DEPOSE, et qui permettent d'adapter des programmes en langage assembleur via LOGO, comme nous l'avons montré tout au long de ce chapitre. Les valeurs effectives des adresses sont contenues dans le fichier ADRESSES, qui est fourni sur la disquette utilitaire. Ces adresses peuvent varier d'une version de LOGO à une autre. Si vous tapez RAMENE "ADRESSES, vous définissez les adresses comme des variables LOGO ordinaires, leurs valeurs sont des entiers. Quand vous utilisez une adresse, il convient de la faire précéder du caractère ":", comme dans .EXAMINE :EPOINT

EPOINT Adresse du caractère actuellement dans le tampon d'édition. Utilisé par l'éditeur et la routine EDOUT.

ENDBUF Adresse qui suit le dernier caractère du tampon d'édition. La routine de sauvegarde sur disquette (voir SAVMOD) copie de $2ØØØ jusqu'à cette adresse. Voir un exemple au chapitre suivant.

SAVMOD Si le contenu de cette case mémoire est zéro, les primitifs RAMENE et GARDE fonctionnent normalement. Si ce n'est pas zéro, GARDE copie tout ce qui se trouve dans le tampon d'édition (et qui peut être du texte –autre que des procédures et des noms –), et RAMENE recopie le fichier dans le tampon d'édition sans en évaluer le contenu (Voir chapitre suivant).

BKTFLG Si cette case mémoire contient 1, LOGO imprime les objets demandés tout en gardant la possibilité de les ramener. C'est particulièrement utile quand vous envoyez sur le périphérique EDOUT (voir chapitre suivant). Toutes les listes seront imprimées avec leurs crochets extérieurs et non sous forme de phrases. IM NOMS donnera les variables LOGO et leur valeur sous la forme CREE "VARIABLE (3) au lieu de "VARIABLE EST 3.

NOINTP Contrôle l'action des caractères spéciaux d'interruption tels que [CTRL] **F** , [CTRL] **G** , [CTRL] **S** , [CTRL] **T** et [CTRL] **W** . Si la case mémoire contient zéro (valeur par défaut), ces touches produisent l'effet habituel en mode DESSINE ou ECRIS. Si elle contient 1, ces touches n'ont plus de signification particulière, et pourront être reconnues par LISCAR. [CTRL] **Z** et [CTRL] [SHIFT] **M** gardent leur rôle. Pour leur enlever leur signification, déposer 255 dans NOINTP.

CH , CV Ces adresses contiennent la position actuelle du curseur, colonne et ligne respectivement. .EXAMINE :CH sort la position du curseur sur la ligne.

OTPDEV Contient l'adresse de la routine actuellement utilisée pour sortir des caractères.

INPDEV Similaire à OTPDEV, mais pour entrer des caractères.

USHAPE Pointeur pour les formes de tortue définies par l'usager (voir chapitre 4).

SSIZE Taille de la tortue ou d'une forme créée par l'usager. La valeur par défaut est 1.

INVFLG Détermine si les caractères seront blancs sur fond noir (valeur de INVFLG = 255), noirs sur fond blanc (valeur = 0) ou clignotants (valeur = 1). La valeur par défaut est 255.

NARG2 Premier des quatre octets qui contiennent le second argument de .APPEL.

NARG1 Première adresse de la mémoire utilisable. Toute la mémoire à partir de cette adresse jusqu'à ANSN4 + 3 est utilisable pour les routines de l'usager.

ANSN4 Dernière adresse disponible pour l'utilisateur. ANSN4, ANSN4 + 1, ANSN4 + 2, ANSN4 + 3 sont quatre octets libres.

USERPZ Première adresse mémoire en page zéro disponible pour l'utilisateur. La mémoire s'étendant de USERPZ à $FF n'est pas utilisée par LOGO.

HIMEM (.EXAMINE :HIMEM) + 256 * (.EXAMINE :HIMEM + 1) sort l'adresse la plus grande disponible pour les programmes écrits en langage machine par l'utilisateur. Déterminé par MAXFILES 1 à. $9AA5. Voir VZZZZZ.

OTPFX2 Sauter à cette adresse permettra à .APPEL de sortir l'entier mémorisé dans les quatre octets NARG2 à NARG2 + 3.

OTPFIX Analogue à OTPFX2, mais sort l'entier mémorisé dans les quatre octets d'une case mémoire page zéro pointée par le registre Y.

OTPTRU Sort "VRAI.

OTPFLS Sort "FAUX.

GETRM1 Référencer cette mémoire deux fois, par exemple LDA GETRM1, LDA GETRM1, permet de charger LOGO dans l'extension mémoire, et de court-circuiter la ROM.

KILRAM Référencer cette adresse permet de disposer de la ROM moniteur Apple. C'est le cas pendant l'exécution de .APPEL sauf mention explicite du contraire.

PPTTP Autre possibilité de sortie pour les routines écrites en langage machine par l'utilisateur. Sauter à cette adresse fait exécuter le

primitif LOGO NIVEAUSUP. Il est utile de revenir de cette manière à LOGO quand s'est produite quelque condition d'erreur qui rend inacceptable la poursuite de l'exécution de la procédure .APPELante.

COUT La routine normale de sortie de caractères sur l'écran par LOGO. Affiche à l'écran le caractère contenu dans A.

EDOUT Routine qui place dans le tampon d'édition le caractère contenu dans le registre A. Dépose A à la case mémoire pointée par EPOINT et incrémente EPOINT. Revient sans rien effectuer si EPOINT est supérieur à $3FFF. Peut être utilisé avec PERIPH pour forcer LOGO à placer du texte directement dans le tampon d'édition. Voir exemple au chapitre suivant.

PNTBEG Routine pour remettre EPOINT au début du tampon d'édition. A utiliser avant le premier appel à la sortie vers le tampon d'édition (avec PERIPH :EDOUT) Voir exemple au chapitre suivant.

ENDPNT Routine pour envoyer ENDBUF à EPOINT. A utiliser à la fin de l'affichage dans le tampon d'édition. Voir PNTBEG , EPOINT , ENDBUF.

BELL Routine pour actionner le bip. Utiliser AFR CAR 7 pour actionner le bip sous LOGO.

HOME Remet le curseur au début de la page et vide l'écran.

CLREOP Vide la partie comprise entre le curseur et le bas de l'écran.

SCROLL Produit un déroulement de l'affichage.

CLREOL Vide la fin de la ligne.

FILLEN Contient la longueur du dernier fichier chargé.

FILBEG Adresse de début du dernier fichier chargé.

VZZZZZ (.EXAMINE :VZZZZZ) + 256 ✳(.EXAMINE :VZZZZZ + 1)sort la plus petite adresse disponible pour les programmes en langage machine écrits par l'utilisateur. Bien que cette adresse puisse être inférieure à $99AØ, il est déconseillé de placer du code dans la région intermédiaire, car des versions ultérieures de LOGO sont susceptibles d'utiliser cet espace mémoire.

Nota Bene
Même en effaçant ADRESSES et OPCODES inutiles, vos programmes en assembleur ne tiendront pas toujours en mémoire. Vous pouvez alors générer ces programmes hors de Logo, puis les conserver comme fichiers binaires et les appeler depuis Logo.

• *Utilisation du système LOGO comme éditeur de texte*

Le système LOGO est conçu pour lire et sauvegarder des fichiers qui contiennent des procédures LOGO. En modifiant le système, on peut utiliser LOGO pour lire et sauvegarder des fichiers de texte (et ainsi se servir de l'éditeur LOGO comme un simple éditeur de texte), se servir des disquettes pour une sauvegarde temporaire, et concevoir des programmes auto-chargeants en LOGO.

Normalement le primitif GARDE sauvegarde les noms et procédures se trouvant dans l'espace de travail en les plaçant d'abord dans le tampon d'édition, puis en recopiant le tampon sur une disquette. Si vous voulez écrire un texte quelconque sur disquette pour qu'il puisse être rechargé dans le tampon d'édition sans être évalué, vous pouvez vous servir du drapeau du mode de sauvegarde : ce drapeau contrôle l'action des primitifs RAMENE et GARDE. En temps normal, l'emplacement mémoire désigné par SAVMOD contient la valeur zéro. Quand vous mettez toute autre valeur dans cet emplacement, l'instruction GARDE sauvegarde le contenu du tampon d'édition au lieu de sauvegarder l'espace de travail, et l'instruction RAMENE n'évaluera pas le contenu du tampon d'édition après l'avoir lu sur la disquette. Seul le primitif LOGO AUREVOIR a pour effet de réinitialiser ce drapeau. L'adresse effective de ce drapeau est indiquée dans le fichier ADRESSES sur la disquette utilitaire.

Si vous voulez utiliser l'éditeur de procédures LOGO comme éditeur de texte pendant toute une session, vous pouvez taper

```
.DEPOSE :SAVMOD 1
```

pour que RAMENE et GARDE traitent le tampon d'édition comme du texte. Si alors vous voulez ramener ou sauvegarder des procédures ou des noms, il suffit de taper

```
.DEPOSE :SAVMOD Ø
```

pour que les choses redeviennent normales.

• *Affichage de fichiers*

S'il n'y avait aucun moyen d'afficher des fichiers, l'éditeur LOGO serait inutile pour l'édition de texte. Les procédures suivantes vont afficher le contenu du tampon d'édition sur le périphérique connecté au slot SLOT. Avant de commencer, il est nécessaire de déposer à l'adresse :ENDBUF + 1 une valeur inférieure à 64 (la valeur de ENDBUF est dans le fichier ADRESSES).

```
POUR HARDCOPIE
PERIPH :SLOT
```

```
PRINTMEM 8192 256 * (.EXAMINE :ENDBUF + 1) ...
                              ... + .EXAMINE :ENDBUF
PERIPH ∅
FIN
```

avec

```
POUR PRINTMEM :DEPUIS :JUSQUA
SI :DEPUIS > :JUSQUA STOP
AFR CAR .EXAMINE :DEPUIS
PRINTMEM :DEPUIS + 1 :JUSQUA
FIN
```

- ***Fichiers autochargeants***

Vous pouvez utiliser SAVMOD pour ajouter un texte arbitraire à la fin des procédures et des noms que vous allez sauvegarder sur disquette. Ceci est surtout utile si vous voulez qu'un programme particulier s'exécute à chaque fois qu'un certain fichier est chargé. Par exemple, supposons que vous vouliez qu'une procédure appelée DEBUT s'exécute automatiquement à chaque fois que vous ramenez le fichier JEU; ceci peut se faire en arrangeant les choses de telle sorte que la commande DEBUT soit exécutée automatiquement à chaque fois que JEU est ramené. Pour cela, définissez toutes les procédures nécessaires pour JEU, tapez EDITE TOUT pour mettre l'espace de travail en entier dans le tampon d'édition. Allez alors à la fin du tampon (en utilisant CTRL **F**) et insérez les commandes qui doivent être exécutées directement par exemple DEBUT. Puis tapez CTRL **C** pour sortir de l'éditeur, puis aussitôt après CTRL **G** et enfin tapez

```
.DEPOSE :SAVMOD 1
GARDE "JEU
.DEPOSE :SAVMOD ∅
```

Maintenant, à chaque fois que le fichier JEU est chargé, les procédures seront définies, et l'instruction que vous avez ajoutée à la fin du tampon sera exécutée. Toutefois un fichier dans lequel la commande RAMENE est utilisée ne peut pas être autochargeant.

- ***Ecrire dans des fichiers sur disquette***

SAVMOD vous permet également d'écrire des programmes LOGO qui gèrent leur propres entrées-sorties sur disquettes.

Voici une série de procédures LOGO qui permettent d'écrire dans les fichiers disques. Pour cela exécutez OUVRIR avec comme argument d'entrée le nom du fichier, puis utilisez la commande INSCRIRE pour écrire dans le tampon du fichier. Pour fermer le fichier, tapez FERMER. Cela sauvegardera sur la disquette le contenu du fichier qui vient d'être ouvert.

Vous ne pouvez pas utiliser l'éditeur, ni le mode graphique, quand un fichier est ouvert. Vous ne pouvez pas non plus écrire plus de 8192 caractères. Tout caractère dépassant cette limite sera ignoré.

```
POUR OUVRIR :FICHIER
CREE "FICHIER.OUVERT :FICHIER   ; SAVMOD , PNTBEG , EDOUT,
.APPEL :PNTBEG 0                  EPOINT ,  ENDBUF ,
FIN                               et ENDPNT se trouvent
                                  dans le fichier ADRESSES

POUR INSCRIRE :MACHIN
PERIPH :EDOUT
AFFICHE :MACHIN
PERIPH Ø
FIN

POUR FERMER
.APPEL :ENDPNT Ø   ; actualise le pointeur de fin de tampon
.DEPOSE :SAVMOD 1
GARDE :FICHIER.OUVERT
EFNOM "FICHIER.OUVERT
.DEPOSE :SAVMOD Ø
FIN

POUR RAJOUTER :FICHIER
CREE "FICHIER.OUVERT :FICHIER
.DEPOSE :SAVMOD 1
RAMENE :FICHIER
```

```
.DEPOSE :SAVMOD Ø
.DEPOSE :EPOINT .EXAMINE :ENDBUF
.DEPOSE :EPOINT + 1 .EXAMINE :ENDBUF + 1
FIN
```

Pour les programmeurs en assembleur : la case mémoire ENDBUF est le pointeur LOGO qui pointe sur la fin du tampon d'édition. (Le tampon d'édition commence à l'emplacement $2ØØØ et va jusqu'à $3FFF.)

PNTBEG réinitialise ENDBUF à l'adresse $2ØØØ. EDOUT est une routine qui prend un caractère dans le registre A, le place dans la case mémoire pointée par EPOINT et incrémente EPOINT. La routine de sauvegarde sur disquette effectue une copie à partir de $2ØØØ jusqu'à ENDBUF, donc FERMER doit actualiser ENDBUF grâce à EPOINT en appelant la routine ENDPNT..

Si vous êtes en train d'écrire un fichier que LOGO devra être en mesure de ramener ou d'exécuter ou d'interpréter comme des données, alors vous apprécierez propablement que des crochets soient affichés autour des listes de niveau supérieur. Autrement dit, alors que LOGO fonctionne normalement ainsi :

```
AFFICHE [1 2 [Q B] 3]
1 2 [Q B] 3
```

vous préfèrerez sans doute qu'il fonctionne comme suit :

```
AFFICHE [1 2 [Q B] 3]
[1 2 [Q B] 3]
```

Pour obtenir ce résultat, déposez la valeur 1 à l'emplacement BKTFLG; pour revenir à l'utilisation normale, remettez la valeur 0 d'origine. (Mise à part cette utilisation d'écriture de fichiers sur disquette, vous pouvez trouver pratique que LOGO affiche des crochets extérieurs quand vous effectuez du traitement de liste. De cette manière une liste seulement constituée d'un mot unique ne sera pas affichée de la même manière que le mot lui-même). Les modifications de BKTFLG par .DEPOSE doivent s'effectuer au début et à la fin de la routine INSCRIRE pour ne pas interférer avec les autres opérations d'affichage de LOGO.

Par ailleurs, BKTFLG contrôle l'action des instructions IMPRIME et EDITE NOMS. Quand BKTFLG contient la valeur 1, les noms seront affichés sous le format

```
CREE "NUM (259)
```

- ***Différents paramètres du système***

Ce paragraphe contient diverses informations sur LOGO et sur cette implémentation particulière. Il n'est absolument pas nécessaire de connaître ce qui est présenté ici pour pouvoir utiliser LOGO. Ceci est plutôt destiné à satisfaire la curiosité de quelques uns.

L'écran graphique.

Quand elle regarde vers le haut, la tortue peut avancer de 121 pas à partir du centre, avant de "déraper" jusqu'au bas de l'écran (effet d'enroulement). Elle peut reculer de 120 pas à partir du centre en allant vers la gauche ou vers la droite. Si vous changez le facteur d'échelle (voir la description du primitif .ASPECT au chapitre 3), les valeurs maxima et minima pour les ordonnées seront modifiées, mais les abscisses resteront sans changement.

Les nombres.

Le plus petit nombre avec lequel on peut effectuer des opérations est 10^{-38} (noté en LOGO 1N38), et le plus grand 9.9999×10^{38} (noté en LOGO 9.9999E38). Le plus grand entier positif est 2147483647, le négatif de plus grande valeur absolue est −2147483647.

Les valeurs ASCII

Il y a une correspondance entre les caractères utilisables en LOGO et les nombres de 0 à 255. Le primitif ASCII, si on lui donne une lettre, sort le nombre associé à cette lettre. Le primitif CAR fait l'inverse, le résultat étant sorti sous forme d'un mot d'une seule lettre. Le caractère associé à 0 (souvent appelé NULL), est particulier : il représente le mot vide. De même que le primitif PHRASE ignore les listes vides comme argument d'entrée, MOT ignore le mot vide. Il est impossible de créer un mot à partir du mot vide, si ce n'est justement le mot vide.

Le primitif LISCAR (en abrégé LC) lit une touche tapée sur le clavier et sort cette information sous forme d'un mot d'une seule lettre. Il y a certaines touches d'interruption qui ne peuvent jamais être transmises par LC. Il s'agit de CTRL **F** , CTRL **S** , CTRL **T** , CTRL SHIFT **M** , CTRL **W** , CTRL **Z** et CTRL **G** . Les fonctions produites par ces touches sont utilisables, que LOGO soit en mode DESSINE ou en mode ECRIS. Les valeurs ASCII sont données dans PREMIERS PAS AVEC EDI-LOGO, page 130.

Pour retrouver les valeurs ASCII de n'importe quelle touche, tapez

AFFICHE ASCII LISCAR
et ensuite tapez la touche en question.

Par exemple

Touches		ASCII
ESC		27
←		8
CTRL	P	16
CTRL	N	14

Il est parfois très utile de pouvoir annuler l'effet de CTRL **G** ou de l'utiliser pour autre chose que la fonction qui lui a été assignée. Pour cela, LOGO fournit un point d'ancrage pour détourner les significations spéciales des touches mentionnées plus haut (à l'exception de CTRL **Z** et CTRL SHIFT **M)** . Déposer la valeur 1 dans la case mémoire NOINTP inhibe ces caractères d'interruption. Il faut déposer la valeur 0 pour les refaire fonctionner normalement. (*Attention :* si vous déposez 255 dans cette case mémoire, vous inhibez *tous* les caractères d'interruption. C'est à manier avec beaucoup de précautions). Voyez à la fin du chapitre précédent les adresses mémoires particulières.

Quand les caractères d'interruption sont inhibés, le primitif LISCAR sortira toute touche que vous frapperez, excepté bien entendu CTRL **Z** et CTRL SHIFT **M**.

Autre occasion où l'inhibition des interruptions est utile : dans les procédures qui effectuent des tâches qui doivent être annulées avant de revenir au niveau supérieur. Si l'utilisateur appuie sur CTRL **G**, pendant l'exécution d'une procédure qui change temporairement PERIPH pour sortir sur un autre périphérique, toutes les informations sorties à partir de ce moment là seront envoyées sur le périphérique choisi, sauf à modifier à nouveau PERIPH, ou utiliser CTRL SHIFT **M** . Ainsi, la procédure suivante peut-elle être exécutée sans craindre d'envoyer des messages ARRET ! ou PAUSE au périphérique en question :

```
POUR TCMD :CMD :ARG
.DEPOSE :NOINTP 255
PERIPH 1 ; périphérique connecté au slot 1
(AFFICHE :CMD :ARG )
PERIPH 0
.DEPOSE :NOINTP 0
FIN
```

Longueur des lignes

En LOGO les lignes ne peuvent pas comporter plus de 256 caractères. De plus, les listes arguments de EXECUTE et REPETE et toutes les sous-listes du deuxième argument de DEFINIS doivent obéir à la même restriction.

Par contre, les lignes tapées sous l'éditeur (après POUR "nom de procédure") peuvent être de n'importe quelle longueur. Bien sûr, la longueur est quand même limitée par la longueur du tampon d'édition. Il en est de même pour les lignes de fichiers sur disquettes. Le tampon d'édition peut contenir 8192 caractères.

Le stockage en LOGO.

LOGO stocke les procédures de manière beaucoup plus efficace que la plupart des autres langages. Chaque procédure est stockée comme une liste de lignes, les lignes étant des listes de listes ou de mots. A chaque utilisation, chaque mot occupe la même quantité d'espace, quel que soit le nombre de caractères qui le compose. Par conséquent, il n'y a pratiquement aucune pénalisation si l'on utilise des noms longs et descriptifs pour les procédures ou les variables. Quand LOGO a utilisé tout l'espace de stockage disponible, il met en route un processus appelé recyclage. Cela signifie simplement que LOGO cherche quels emplacements de la mémoire ne sont pas utilisés, et en fait une liste. Ensuite, quand LOGO a besoin d'utiliser un de ces emplacements, il le retire de cette liste.

Quand LOGO est en train de chercher un emplacement mémoire au cours d'un recyclage, il ne peut rien faire d'autre (en particulier, il ne peut pas exécuter votre procédure). Ce processus peut donc gêner certains programmes, où il est important d'avoir les réponses en temps réel. Si cela doit être gênant, effectuez des appels au primitif .RECYCLE à des pauses naturelles dans votre programme.

• *Organisation de la mémoire*

Voici la description de la manière dont le système LOGO utilise la mémoire de l'Apple II (voir page suivante)

Nul :	$0000 - $0003 : $	4 octets	(4 octets) La liste vide
Divers :	$0004 - $07FF : $	7FC octets	(2044 octets) tampons divers
Piles :	$0800 - $1BF5 : $	13F6 octets	(5110 octets soit 2555 mots) PDL. VDPL
Vecteurs :	$1BF6 - $1BFF : $	A octets	(10 octets) Adresses de retour
Autre code :	$1C00 - $1FFF : $	400 octets	(1K octets) Sous-routines d'entrée-sortie
Tampon :	$2000 - $3FFF : $	2000 octets	(8K octets) Tampon édition-graphique
LOGO :	$4000 - $999F : $	59A0 octets	(22944 octets) Code LOGO
Utilisateur :	$99A0 - $9AA5 : $	106 octets	(262 octets) Mémoire libre
DOS :	$9AA6 - $BFFF : $	255A octets	(9562 octets) Code DOS, tampons
Entrée-Sortie :	$C000 - $CFFF : $	1000 octets	(4K octets) Portes d'entrée-sortie mémoire
Tableau Nodes :	$D000 - $F65F : $	2660 octets	(9824 octets soit 2456 nodes) Espace Nodes
Tableau Types :	$F660 - $FFF7 : $	998 octets	(2456. octets) Codes des types des nodes
Mem. fantôme :	$D000 - $DFFF : $	1000 octets	(4K octets) Stockage statique commutable
Libre :	$FFF8 - $FFF9 : $	2 octets	(2 octets)
Interruptions :	$FFFA - $FFFF : S	6 octets	(6 octets soit 3 vecteurs) Vecteurs d'interruption

Imprimé en France par l'imprimerie Fontaine S.A. Poitiers - Code n° 710.006-8