
LOGO SANS PEINE

Auto-initiation au LOGO MO5 – TO7/70

Vincent Dureau et Dominique Péré



**cedic
nathan**

Maquette : Alain Dufourcq et Anne Génot
Montage : Michèle Beaucamp

Ce volume porte la référence
ISBN 2-7124-4051-10

Toute reproduction, même partielle, de cet ouvrage est interdite. Une copie ou reproduction par quelque procédé que ce soit, photographie, photocopie, microfilm, bande magnétique, disque ou autre, constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1957 sur la protection des droits d'auteur.

© C.A.D.I.C. 1985
C.A.D.I.C., 6-10 boulevard Jourdan, 75014 Paris
Tél. (1) 369.06.06

SOMMAIRE

CHAPITRE 1. Mode d'emploi

Avant le chargement	9
Le chargement	9
Après le chargement	10
Quelques conseils	10

CHAPITRE 2. L'outil d'initiation "INITIA

Le menu	12
Option 1 : Utilisation du programme et des commandes de la tortue graphique	13
Écriture	14
Exécution	16
Un détour par l'éditeur LOGO	17
Écriture d'une expression LOGO	17
Dites-moi sous quoi vous êtes, je vous dirai ce que vous pouvez faire	18
L'éditeur LOGO	20
Sous LOGO	20
Option 2 : Commandes de la tortue graphique et opérations sur les nombres	22
Commandes	23
Opérations	24
Option 3 : Commandes et opérations sur les mots	26
Données et commandes non graphiques	27
Opérations sur les mots	28
Emboîtements	29
(Re)coller les mots	30
Option 4 : Les mots et les listes	32
La liste	33
Insérer des mots	34
La primitive liste	37
Sous LOGO	38
Option 5 : La commande REPEAT	39
Sous LOGO	42
Option 6 : L'instruction conditionnelle	43
La primitive SI	44

CHAPITRE 3. La commande POUR	
Procédure commande	49
Procédure opération	49
Procédure avec arguments	50
Un exemple : la procédure explose	51
Sauver	54
CHAPITRE 4. L'analyseur syntaxique "SYNTA	
Les différents niveaux du programme	57
Un exemple	58
Le menu	58
Quelques conseils	59
Des exemples pour vous entraîner	60
CHAPITRE 5. L'outil de visualisation "VISUA	
Le menu	65
Un exemple	66
D'autres exemples	68
CHAPITRE 6. Outil de trace "TRACE	
Exemple (pas à pas)	73
Un exemple avec modification du contexte	76
Un exemple avec choix du niveau de contexte	77
CHAPITRE 7. Procédures	
Quelques procédures pour les mots et les listes	85
Annexes	
Réponses aux exercices	89
Lexique	94

PRÉSENTATION

Vous allez utiliser des programmes d'initiation au langage LOGO. Ces programmes sont eux-mêmes écrits en LOGO, et lorsque vous serez devenus des « costauds » du LOGO, vous pourrez comprendre comment ils ont été écrits.

Nous vous conseillons de commencer par *le programme d'initiation par thèmes* et, la première fois que vous l'utiliserez, de choisir les thèmes dans l'ordre croissant.

Lorsque vous aurez compris et manipulé ce programme, vous connaîtrez la moitié des primitives* LOGO. Vous distinguerez les différents types de primitives* et vous connaîtrez les objets* LOGO.

Ensuite, il faudra vous jeter à l'eau.

Le chapitre « POUR » vous aidera alors à créer vos propres procédures.

Puis, à l'aide d'un deuxième outil, *l'analyseur syntaxique*, vous pourrez créer vos propres expressions. Vous aurez alors acquis une plus grande autonomie.

L'outil de visualisation vous montrera pas-à-pas comment LOGO interprète une expression qui lui est destinée. Cet outil repère aussi les erreurs éventuelles qui se seraient glissées dans les expressions.

Lorsque vous aurez utilisé et compris ces trois premiers outils, vous aurez une convivialité suffisante avec LOGO pour aller aussi loin que vous le voudrez avec *l'outil de trace*.

Celui-ci démonte complètement le mécanisme de LOGO, et vous permet d'accéder, à tout moment, à l'environnement* LOGO du programme que vous lui proposerez. La familiarisation avec ces outils représente plusieurs heures d'initiation, puis d'approfondissement en LOGO. Après un premier passage, vous pourrez vous lancer dans la programmation en connaissance de cause. Et, si plus tard, vous avez des problèmes, l'analyseur syntaxique et les outils de visualisation et de trace seront toujours là pour vous aider...

Le dernier chapitre de ce livret comporte des programmes qui vous ouvriront la voie pour construire vous-même votre univers LOGO.

La plupart des exemples utilisent la notation préfixée, mais on peut utiliser les opérations sur les nombres $+$ $-$ $*$ $>$ $<$ dans les expressions écrites dans les programmes.

Un petit lexique est à la fin du livre et concerne les mots suivis de $*$.

CHAPITRE 1

MODE D'EMPLOI



Vous avez à votre disposition quatre programmes sur vos deux cassettes.

1^{ère} cassette • le programme d'initiation "INITIA,
• l'analyseur syntaxique "SYNTA.

2^{ème} cassette • le programme d'évaluation et d'exécution "VISUA,
• le programme de trace "TRACE.

Avant le chargement

Lorsque vous chargerez un de ces programmes, faites attention à la place libre en mémoire. Ces programmes écrits en LOGO sont assez gros et nous vous conseillons d'effacer toute la mémoire avant de les charger en écrivant :

.EFT **ENTREE**

Si vous venez d'allumer votre micro-ordinateur, .EFT n'est pas nécessaire.

Le chargement

Maintenant, vous allez charger en mémoire le premier outil que nous avons prévu pour vous : le programme d'initiation.

Nous vous conseillons d'utiliser les outils dans l'ordre : ils traitent des difficultés de manière croissante.

Lorsque vous avez mis la cassette dans le magnétophone, vérifiez que vous êtes au début de la bande magnétique, puis tapez :

?RAMENE "INITIA **ENTREE**

*Le symbole **ENTREE** signifiera qu'il faut que vous tapiez sur la touche **ENTREE** du clavier, cette touche « valide » la ligne qui a été écrite et LOGO évalue* cette ligne ; en l'occurrence LOGO va aller chercher le « Fichier » INITIA sur la cassette et charger toutes les procédures et les noms de variables dans la mémoire du micro-ordinateur.*

Maintenant, vous lisez sur l'écran la succession des procédures qui se définissent l'une après l'autre :

VOUS VENEZ DE DÉFINIR « NOM-DÉ-PRO »

(si cela ne se produit pas, il est vraisemblable que vous n'avez pas appuyé sur la touche lecture du magnétophone).

S'il y a un problème quelconque au chargement du programme, vous remettez la bande à son départ et vous recommencez !

Après le chargement

Lorsque le point d'interrogation réapparaît sur l'écran, tout est prêt pour le début de votre apprentissage.

Quelques conseils

Lorsque vous verrez une nouvelle primitive ou lorsque vous avez un doute, vous devez vous référer continuellement au manuel de référence. Vous y trouverez un complément aux indications des différents outils ainsi que des primitives nouvelles.

Lorsque vous écrivez sous le langage LOGO, faites bien attention à mettre des blancs entre chaque primitive et entre leurs arguments. Sinon LOGO ne comprend pas :

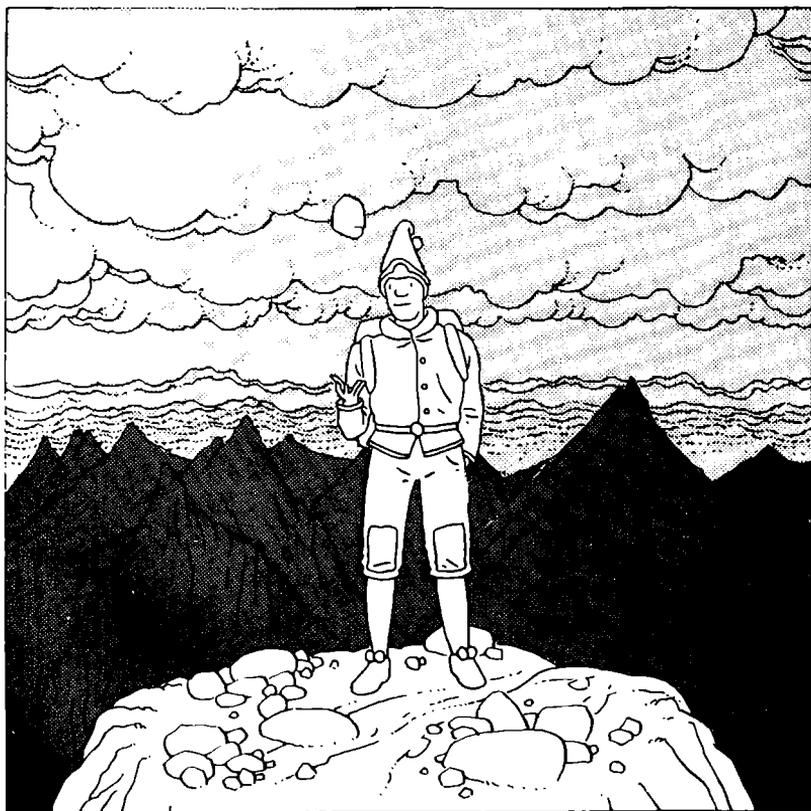
?AV 60 est correct

?AV60 est incorrect et renvoie le message :

COMMENT FAIRE AV60.

CHAPITRE 2

L'OUTIL D'INITIATION "INITIA



Le menu

Vous mettez votre magnétophone en position, comme cela est décrit dans le mode d'emploi.

?RAMENE "INITIA **ENTREE**

Les procédures se définissent.

Puis tapez :

?OUTIL **ENTREE**

Vous obtenez un menu* qui vous propose différentes possibilités.

Avant de choisir, il convient de vérifier que votre lecteur de cassettes est toujours en position lecture, car chaque choix va chercher un complément de programme spécial sur la bande magnétique.

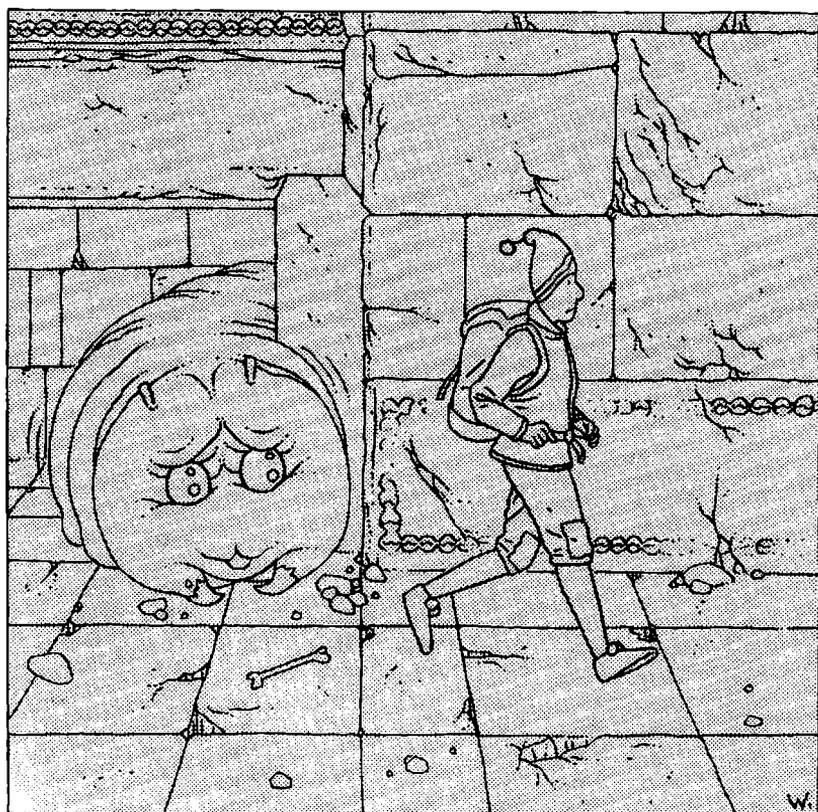
Choisissez une option du menu, puis reportez-vous à la partie correspondante dans cet ouvrage. Nous vous conseillons de choisir d'abord l'option 1.

Si vous avez un problème pendant ce programme, pour le relancer il suffira de taper :

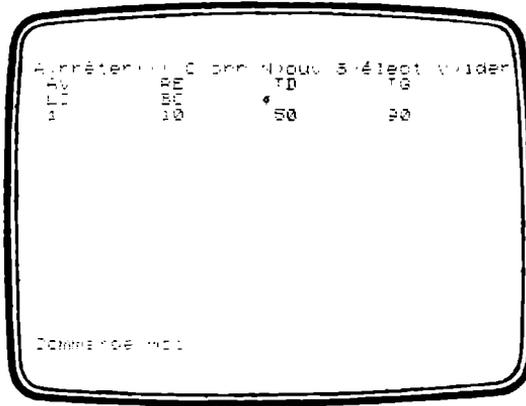
?OUTIL **ENTREE**

Option 1

Utilisation du programme et des commandes de la tortue graphique



Nous sommes dans l'option 1 du programme « outil d'initiation » ; la partie correspondant à ce choix est chargée en mémoire et nous avons maintenant l'écran sous cette forme :



- Si votre écran de téléviseur est différent de la forme ci-dessus, il convient d'arrêter. Pour cela, appuyer sur la touche **A** qui correspond à la fonction A > rrêter (en haut) de l'écran d'écriture (éditeur*). Puis relancer par l'appel :

?OUTIL

et refaire l'option 1 dans le menu*.

- Si l'écran présente bien la forme ci-dessus, nous allons pouvoir faire nos premiers pas en LOGO.

Écriture

Nous sommes d'abord devant l'écran d'écriture. Il permet d'écrire des expressions* LOGO, grâce à l'éditeur* situé dans la zone jaune. L'écran est divisé en cinq parties qui ont toutes une fonction très précise. Nous allons les comprendre et les utiliser avec des exemples.

Le menu en haut sur fond noir donne les commandes possibles. Il suffit de taper la lettre en majuscule :

- A>rrêter!!! pour arrêter le programme et revenir « sous* LOGO » ;
- C>orriger pour revenir en arrière dans la zone d'écriture (la zone jaune du milieu) ;
- S>électionner pour valider le choix fait à l'aide des quatre flèches     , et en face de l'astérisque ;
- V>ider pour revenir à l'écran initial, en effaçant tout ce qui a été écrit.
- N>ouveau chapitre pour faire une nouvelle option.

— *La deuxième partie au-dessous du menu* : c'est la zone des expressions LOGO et de leurs données que l'on peut S>électionner.

— *Dans la zone jaune*, sera écrite la suite des expressions LOGO sélectionnées.

— *Dans la zone bleue*, des messages d'erreur apparaîtront en cas de mauvaise sélection.

— *Dans la zone rouge*, enfin, nous aurons les explications, et les questions correspondantes, aux sélections effectuées.

Au travail donc !

Un coup sur  , et l'étoile se met sur le 50. On sélectionne  , et la zone bleue n'est pas d'accord ! 50 est un nombre, et en LOGO un nombre est une donnée. Pour éviter cette erreur, il faut sélectionner un des éléments écrits en bleu sur fond jaune, ceux écrits en blanc sur fond bleu sont interdits et ne s'écriront pas dans la zone jaune.

Nouveau choix, en manipulant les touches fléchées et en frappant  , sélectionnez REcule. Cette fois-ci, ça marche : dans le jaune le programme écrit, dans le rouge il pose une question.

Une pause : cette partie du programme permet de manipuler des primitives* de la tortue, les autres parties feront avancer dans l'apprentissage d'autres primitives du langage LOGO.

Quelquefois, il sera conseillé de se reporter au manuel de référence pour y lire la définition exacte de la primitive*, sa fonction, le nombre et la nature de ses arguments.

Par exemple : RE prend un argument et c'est forcément un nombre, LC (Lève Crayon) ne prend pas d'argument.

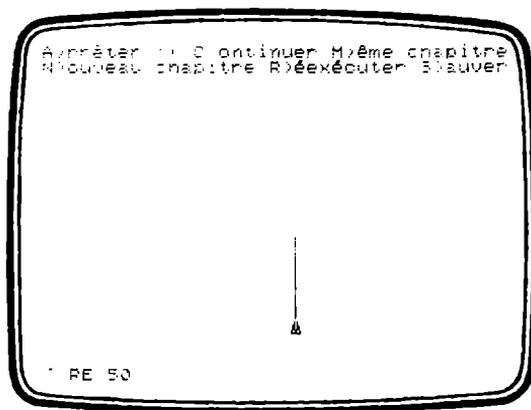
S>électionnez maintenant 50 par exemple.

Cette fois-ci, le commentaire en zone rouge n'est plus une question, mais il se termine par un ! et nous observons, sur le menu* de la première ligne, l'apparition de **ENTREE**.

Cela veut dire que l'on peut maintenant exécuter l'expression LOGO qui vient d'être écrite.

Exécution

Exécutez en appuyant sur cette touche **ENTREE**.



Nous sommes maintenant devant l'écran d'exécution. Il est divisé en trois parties : le centre pour l'exécution des expressions graphiques de la tortue, les quatre lignes du bas pour l'exécution des expressions non graphiques. Derrière le ?, est écrite l'expression exécutée ; enfin un nouveau menu* figure en haut de l'écran (ici la tortue a tracé un trait de 50 pas en reculant).

Les premières lettres des choix du menu permettent différentes possibilités :

- **A** >rrêter pour revenir sous* LOGO en stoppant le programme outil.
- **C** >ontinuer pour revenir à l'écran d'écriture exactement dans l'état où nous l'avions laissé.
- **M** >ême chapitre permet de revenir à l'écran d'écriture vide.
- **N** >ouvel chapitre permet de revenir au menu initial pour choisir une nouvelle option.

- R > exécuter permet de refaire l'expression écrite autant de fois que l'on veut.

Pour le moment nous allons :

- S > auver, c'est-à-dire écrire l'expression sélectionnée dans l'éditeur* de LOGO. Une note de musique indique que cela s'est bien passé.

Puis :

- A > rrêter.

Un détour par l'éditeur LOGO

Maintenant, tapez :

?ED **ENTREE**

L'expression précédemment composée est écrite en blanc sur fond bleu. Nous sommes dans l'éditeur du LOGO. (Nous l'utiliserons dans le chapitre POUR.) Cet « éditeur » va nous permettre de sauvegarder toutes les expressions LOGO qui nous auront paru intéressantes.

Tapez en même temps **CNT** et **C** et l'on revient à LOGO.

Écriture d'une expression LOGO

Revenons maintenant au programme initial :

?OUTIL

Choix : **1**

Écrivez dans l'éditeur d'OUTIL, grâce aux touches fléchées et à la touche **S** :

RE 50 TD 90 AV 10

Nous recommandons de bien lire ce qui s'écrit dans la zone rouge ; cette zone permet de bien connaître les expressions LOGO, ce qui vous rendra autonome le plus vite possible.

Frappez :

ENTRÉE pour exécuter,
et R > exécuter trois fois.

Joli non ?

Maintenant c'est à vous de « jouer » avec la tortue en manipulant les deux zones de programmes (n'oubliez pas de sauver le programme précédent car nous l'utiliserons à la fin du chapitre).

*Dites-moi sous quoi vous êtes,
je vous dirai ce que vous pouvez faire !*

Si vous nous avez bien suivi dans ce chapitre, vous avez, sans vous en rendre compte, travaillé à quatre niveaux différents :

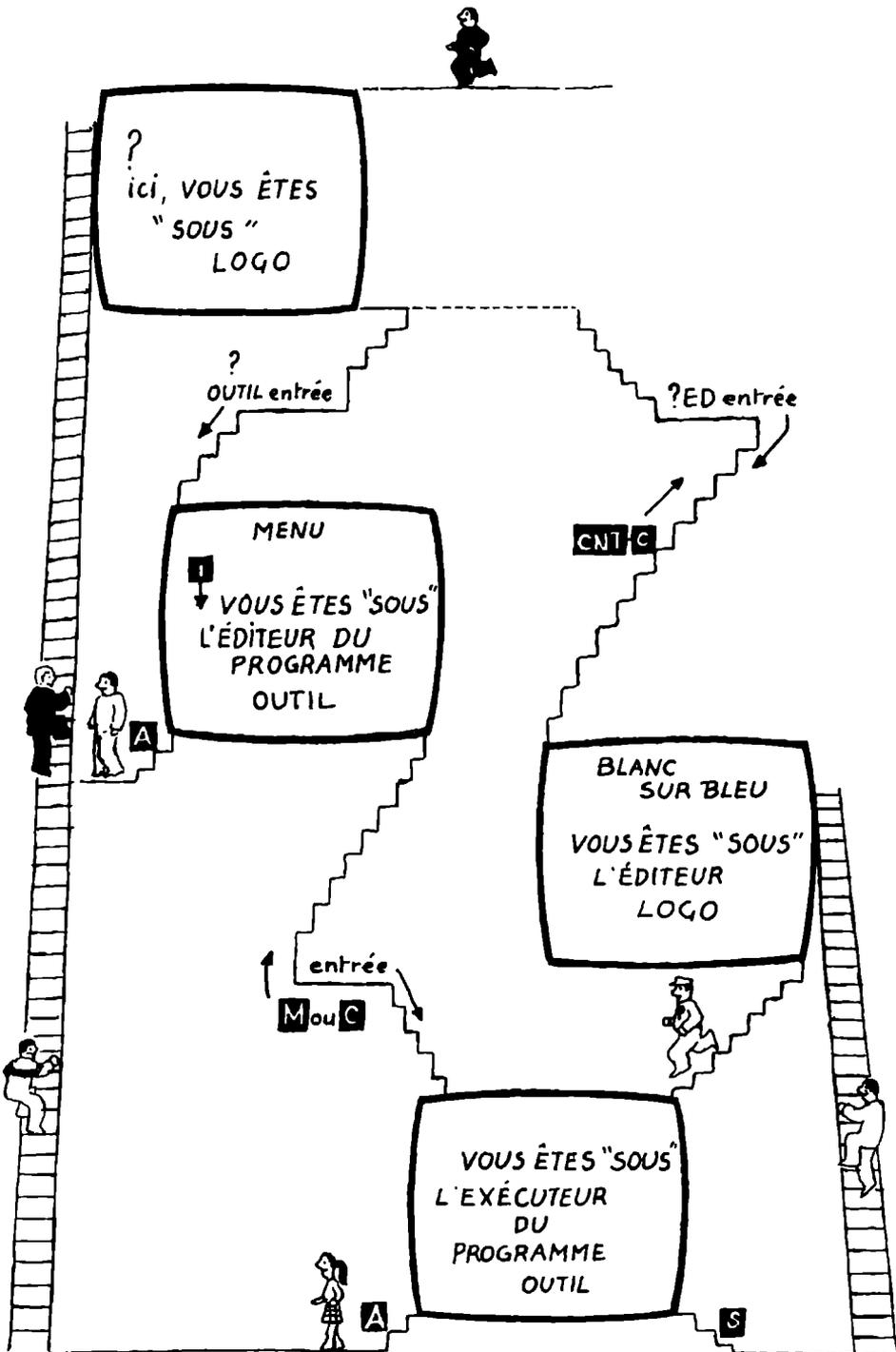
Premier niveau : LOGO, le langage avec ? d'où l'on appelle le programme OUTIL.

Deuxième niveau : éditeur du programme OUTIL.

Troisième niveau : exécution du programme OUTIL.

Quatrième niveau : éditeur de LOGO.

Il est possible que le plan suivant, trouvé dans les bagages de André Deledicq, vous apporte quelques éclaircissements.



EXERCICES

A partir des mots proposés dans l'option 1, construisez des expressions permettant de :

1. dessiner un carré ;



2. dessiner un triangle équilatéral ;



3. dessiner cette forme :



Réponses en page 89.

L'éditeur LOGO

Si vous avez sauvé une expression dans l'éditeur LOGO, nous allons maintenant pouvoir l'utiliser.

A > rrêter

!ED **ENTREE**

Frappez alors :

CNT - Q qui permet de quitter l'éditeur en exécutant l'expression qui y est écrite.

Sous LOGO

Lorsque le point d'interrogation est affiché, vous êtes «sous» le contrôle de LOGO.

Vous pouvez alors écrire quelques expressions (attention aux blancs !).

Si vous vous trompez, vous risquez de tomber sur des messages LOGO, par exemple :

COMMENT FAIRE AV50 ?

Le blanc entre AV et 50 a été oublié.

PAS ASSEZ DE DONNÉES POUR RE ?

Il faut donner un nombre après RE.

Voici quelques expressions que vous pouvez écrire sous LOGO :

?AV 50 TG 120 AV 50 TG 120 AV 50 **ENTREE**

?LC AV 10 BC AV 10 LC AV 10 BC AV 10 **ENTREE**

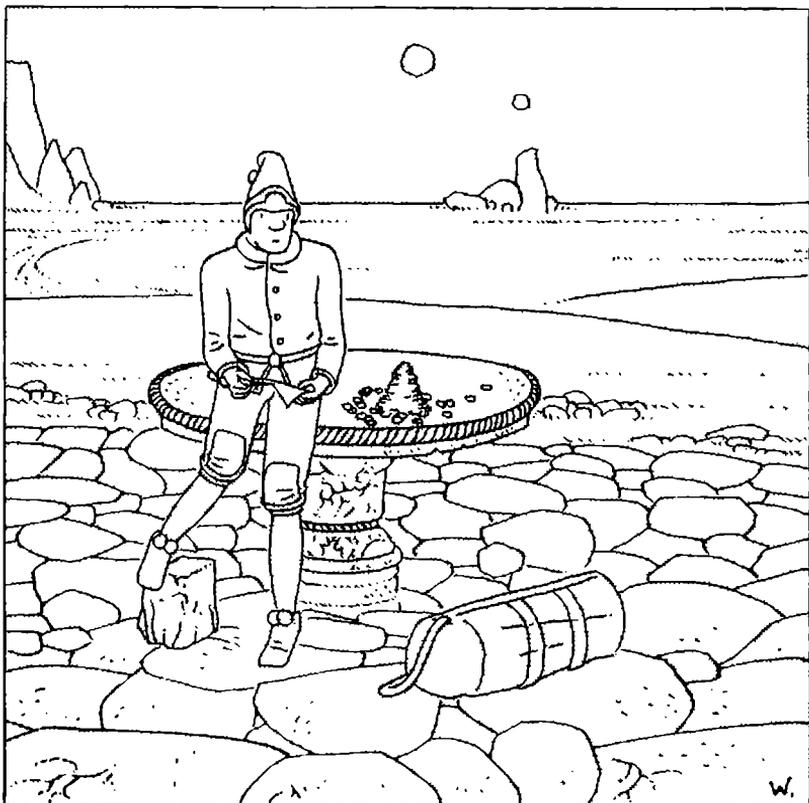
?AV 50 RE 30 TG 90 AV 50 RE 30 TD 90 **ENTREE**

EXERCICES

1. Dessiner un rectangle.
2. Dessiner un hexagone.
3. Dessiner un losange.

Option 2

Commandes de la tortue graphique et opérations sur les nombres



Opérations

Nous allons entrer dans le second monde des primitives LOGO : les *opérations**.

Vous avez remarqué que les opérations SOMME et DIFF (+ et -) sont inaccessibles directement ; ce sont des opérations et elles doivent donc suivre une commande.

Écrivez par exemple :

AV SOMME 20 10

Exécutez.

C'est bien sûr équivalent à AV 30, on peut dire que SOMME 20 10 REND 30 pour AV ; AV s'exécutera donc après l'évaluation* de SOMME 20 10.

C> continuez

AV SOMME 20 10 TG DIFF 90 20

(lisez attentivement la zone rouge).

Exécutez.

La tortue AVance de 30 pas et Tourne à gauche de 70 degrés.

C> continuez

AV SOMME 20 10 TG DIFF 90 20 RE SOMME 10 DIFF 90 20

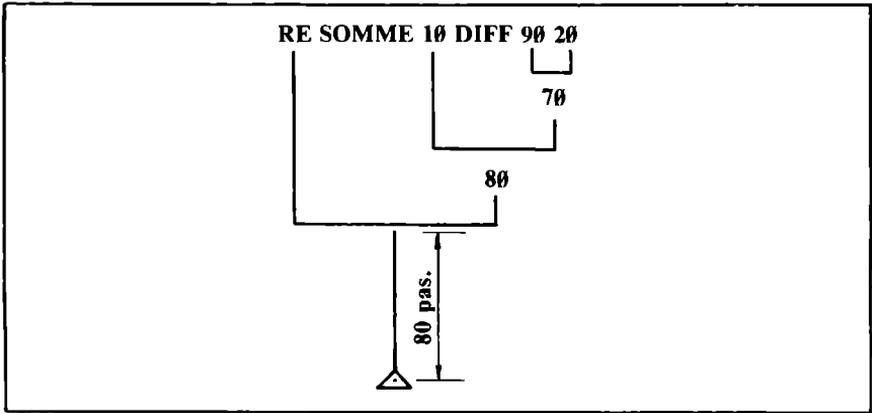
Exécutez.

RE a besoin d'un nombre pour argument.

Somme REND un nombre, et a besoin lui-même de deux nombres 10 et DIFF 90 20.

Le processus est donc celui-ci : RE évalue son argument Somme qui évalue ses deux arguments 10 et DIFF qui évalue ses deux arguments, 90 et 20, donc DIFF REND 70 à SOMME qui REND 80 à RE et la tortue RAvance de 80 pas. Ouf !

Le processus d'évaluation s'effectue à partir de la commande un peu comme une vague qui vient récupérer jusqu'aux derniers arguments.



M > ême chapitre

Écrivez

AV CC

Exécutez.

La couleur du crayon est blanche, donc c'est la couleur n° 7, CC REND la couleur du crayon (cette information est pratique, on évite ainsi par exemple de dessiner en blanc sur fond blanc dans un programme !).

La tortue a donc avancé de sept pas.

EXERCICES

En n'utilisant que les commandes et les opérations prévues dans OUTIL, option 2, construire les expressions qui permettent de :

1. Faire AVancer la tortue de 80 pas.
2. Faire REculer la tortue de 110 pas.
3. Faire AVancer la tortue de 9 pas.
4. Faire REculer la tortue de 78 pas.

Sauvegardez ces expressions dans l'éditeur LOGO et essayez d'écrire des expressions sous LOGO.

Exemples d'expressions :

?AV SOMME 55 RE DIFF 10 5 **ENTREE**

?FCB 0 FCC 2 FCFG 0 CT AV 100 TG 120 AV DIFF 50 100 **ENTREE**

Option 3

Commandes et opérations sur les mots



C> continuez

FCFT 2 FCT 1 EC "SALUT TAPE "LES EC "COPAINS

Exécutez.

TAPE écrit sans aller à la ligne.

M> ême chapitre

EC et TAPE sont des commandes, elles peuvent donc être suivies d'opérations et, bien sûr, d'opérations sur les mots.

Opérations sur les mots

Écrivez

EC PREM "LES

Exécutez.

Cela écrit la première lettre du mot **LES** !

Nous pouvons dire de **PREM REND** la première lettre du mot qui lui est associé.

C> continuez

EC PREM "LES TAPE DER "COPAINS

Exécutez.

DER REND la dernière lettre du mot donné en argument.

C> continuez

EC' PREM "LES TAPE DER "COPAINS EC SP 79

Nous faisons deux constatations :

- **SP** est aussi une opération qui rend le mot de son argument sans la première lettre.
- Un nombre est aussi un mot en **LOGO**.

M> ême chapitre

EC' SD "COPAINS

Exécutez.

SD REND donc le mot de son argument sans sa dernière lettre.

EXERCICES

1. Écrire, avec les possibilités offertes par l'écran, option 3 :

BONJOUR

LE

COPAIN

2. Écrire :

SLC

Emboîtements

Revenons au M> ême chapitre

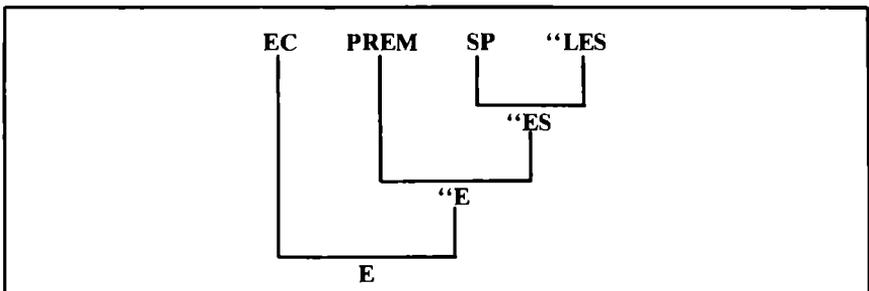
Nous avons déjà vu que les opérations peuvent être utilisées les unes dans les autres ;

Écrivez

EC PREM SP "LES

Exécutez.

C'est bien la première lettre de ES qui est écrite. EC a évalué PREM qui a évalué SP (Sauf Premier), qui lui-même a évalué LES et rendu ES à PREM qui a rendu E à EC qui l'a écrit.



EXERCICES

Avec les possibilités offertes par l'écran, essayez de :

1. écrire un **P** ;
2. écrire **ALU** ;
3. écrire **OU**.

(Re)coller les mots

Nous venons de voir des primitives LOGO de « démontage » de mot.

Essayons une nouvelle opération qui, au contraire, permet de construire des mots.

Écrivez

EC MOT "LES "COPAINS

Exécutez.

La primitive MOT colle deux mots.

M > ême chapitre

V > idez

Écrivez

EC MOT "BONJOUR MOT "LES "COPAINS

MOT a deux arguments. Si l'on veut coller trois mots, il faut faire attention à ce que l'on écrit.

EC MOT MOT "BONJOUR "LES "COPAINS

est équivalent.

EXERCICES

Proposition d'expressions à construire :

1. Écrire SALUTLECOPIAIN (sans utiliser TAPE).
2. Écrire LESBONBONS (sans utiliser TAPE).
3. Écrire LEJOURNAL (sans utiliser TAPE).
4. Écrire en rouge sur fond jaune un chiffre compris entre 0 et 78 collé à JOURS et R > exécuter plusieurs fois pour observer l'effet de HASARD.
5. N'oubliez pas de vous servir de l'éditeur LOGO, et d'écrire quelques expressions sous LOGO, par exemple :

?EC MOT "PREM "DER

?TAPE "PREM "DER

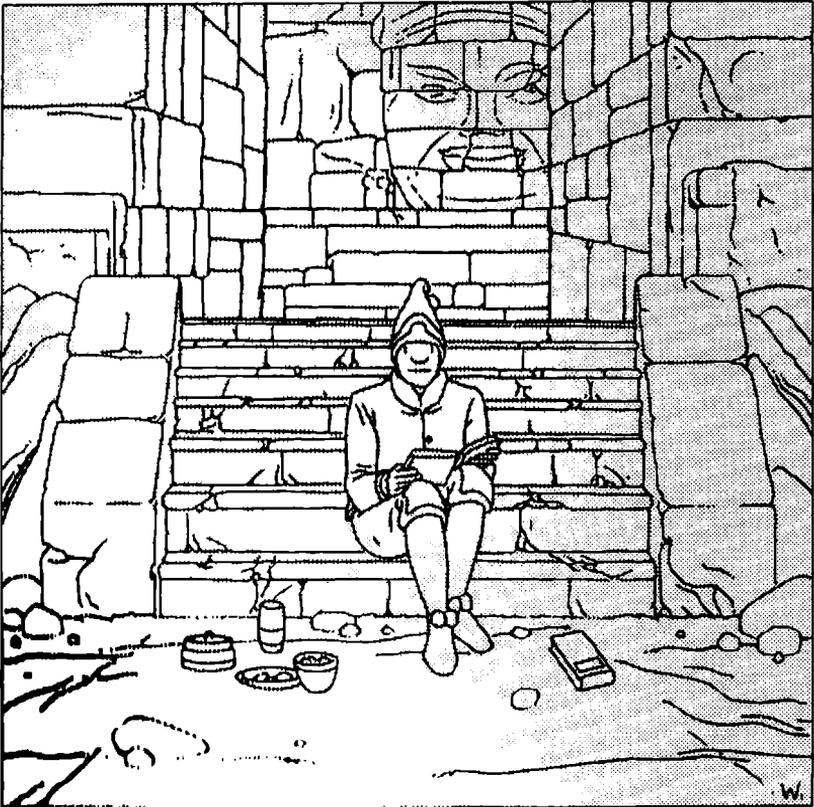
?TAPE "PREM EC "DER

?EC SP SP SP "TOTO EC SD SD SD "TITI

?EC MOT "SP "TOTO SD "TAT

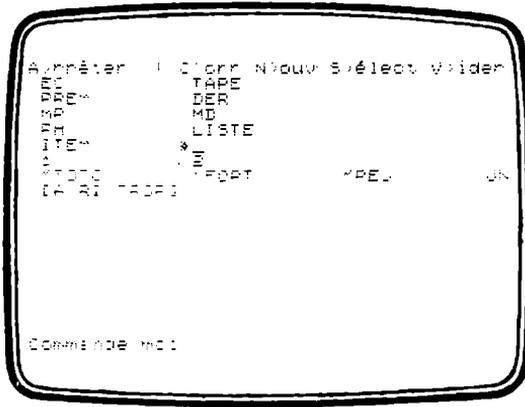
Option 4

Les mots et les listes



Nous avons, dans l'option 3, manipulé l'objet* LOGO *mot* : nous allons faire connaissance maintenant avec l'objet *liste*.

Un objet LOGO, c'est une donnée que prend LOGO comme argument de ses primitives. Il y a deux sortes d'objets : le mot et la liste.



La liste

[A RI TROP] est une liste au sens LOGO, c'est une suite d'objets LOGO séparés par un espace et mise entre crochets.

[A RI TROP] est une liste composée des trois mots "A, "RI et "TROP.

Les opérations de démontage d'objets LOGO, PREM, DER, SP, SD agissent sur les listes comme sur les mots.

Écrivez

TAPE MP "TOTO [A RI TROP]

Exécutez.

MP est une opération de construction de liste, Mets Premier.

C>ontinuez

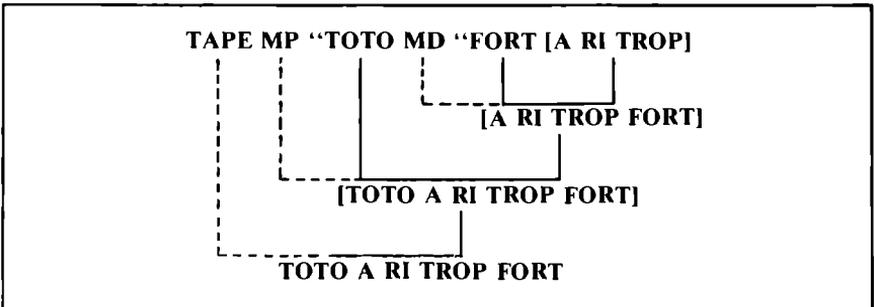
C>orrigez

S>électionnez

TAPE MP "TOTO MD "FORT [A RI TROP]

Exécutez.

MD (Mets Dernier) est le pendant de MP. Ces deux primitives prennent comme premier argument un mot ou une liste et comme deuxième argument une liste.



Insérer des mots

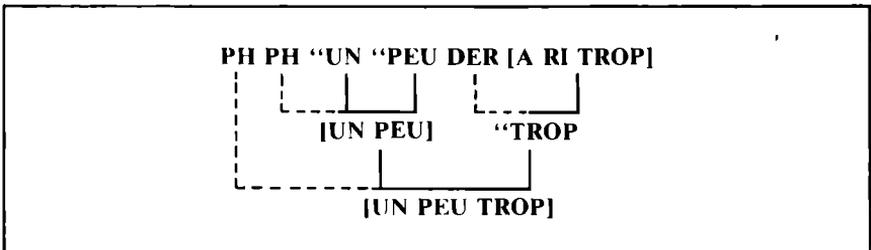
A partir de la « phrase » précédente, nous allons essayer d'écrire :
`TOTO A RI UN PEU TROP FORT`

Pour insérer les deux mots UN et PEU, il y a plusieurs solutions ; en voici une :

Nous savons avec MP et MD mettre `"TROP` au début et `"FORT` à la fin. Le problème est de construire `[A RI UN PEU TROP]`, c'est-à-dire mettre « UN PEU » dans `[A RI TROP]`. Une opération est toute désignée pour cela, c'est PH (PHrase). Elle construit une liste à partir d'objets LOGO.

Ainsi, nous pouvons construire `[A RI UN PEU TROP]` avec PH `[A RI] [UN PEU TROP]`.

Pour construire `[UN PEU TROP]`, c'est facile :



Le problème (avec les outils que nous nous sommes donnés), c'est d'aller pêcher "RI dans [A RI TROP] sans la primitive SP.

Pour cela, nous disposons de la primitive ITEM dont les arguments sont un nombre et une liste. ITEM est une opération qui va pêcher le n^{ième} objet LOGO dans une liste LOGO.

Ainsi : ITEM 2 [A RI TROP] REND "RI.

Donc : PH PREM [A RI TROP] ITEM 2 [A RI TROP] REND [A RI].

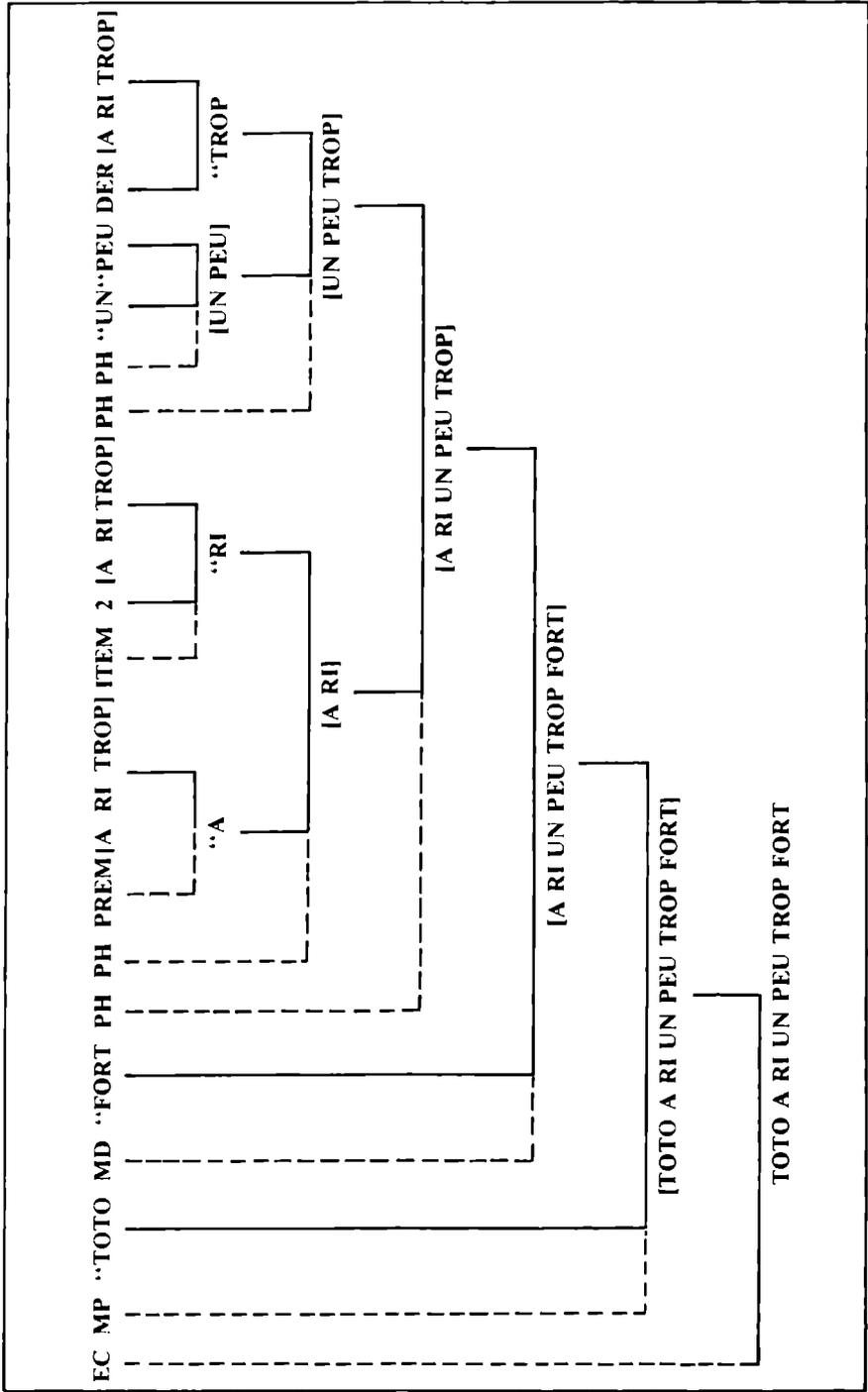
Cela donne finalement :

Écrivez :

```
EC MP "TOTO MD "FORT PH PH PREM[A RI TROP][ITEM 2[A RI TROP]PH  
PH "UN "PEU DER[A RI TROP]
```

Exécutez.

Ouf !



Voilà, nous avons construit une phrase grâce à des montages et des démontages d'objets LOGO.

M> ême chapitre

La primitive liste

Nous connaissons maintenant : les objets listes, les primitives PH et ITEM.

Nous allons voir l'effet de la primitive LISTE.

Écrivez :

EC LISTE "TOTO [A RI TROP]

Exécutez.

Tiens ! les crochets sont restés ! En réalité, LISTE REND à EC [TOTO [A RI TROP]], donc LISTE construit des listes.

C> continuez

EC LISTE "TOTO [A RI TROP] EC PH "TOTO [A RI TROP]

Exécutez.

Nous voyons la différence : PH ne respecte pas les crochets (il détruit le premier niveau des listes), alors que LISTE les impose.

M> ême chapitre

EC LISTE MD [A RI TROP] [A RI TROP] MD "FORT [A RI TROP]

Exécutez.

LISTE a rendu à EC :

[[A RI TROP [A RI TROP]] [A RI TROP FORT]]

Donc, un élément d'une liste peut être une liste.

Une liste composée de mots est appelée une *liste plate*.

Une liste composée de listes peut être décrite par sa *profondeur*.

Ainsi, la profondeur de la liste précédente est 3.

3 est le nombre maximum d'emboîtements des listes de cette liste.

EXERCICES

Avec les mots et primitives donnés sur l'écran de l'option 4, écrivez :

1. A I O U

2. TOTO RI TROP PEU

3. TOTO A TROP RI [UN PEU]

Sous LOGO

En travaillant sous LOGO, essayez ces quelques exemples d'expression :

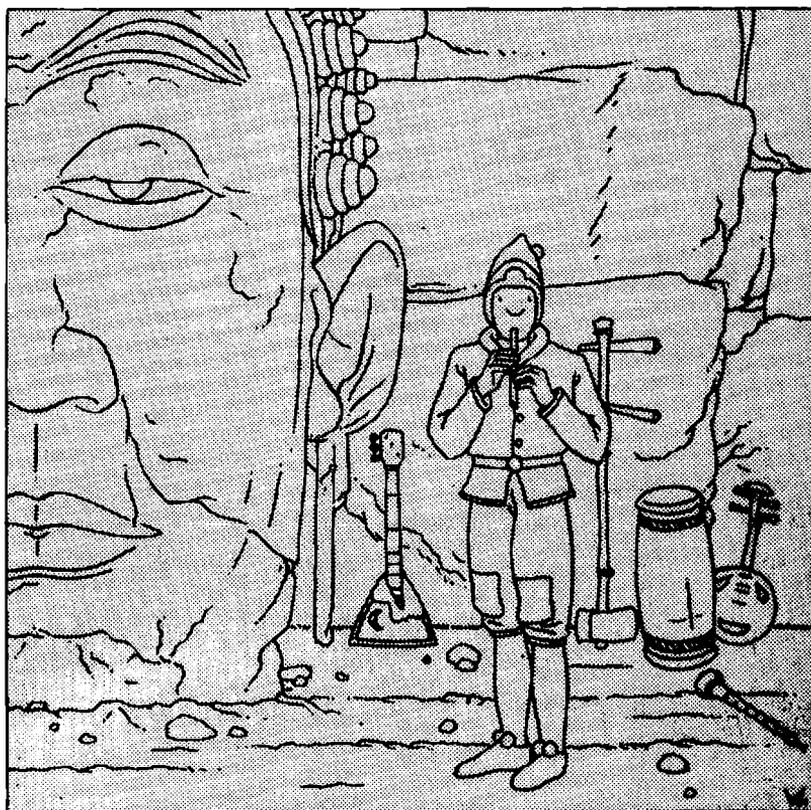
?EC PH LISTE "A "TA "SANTE

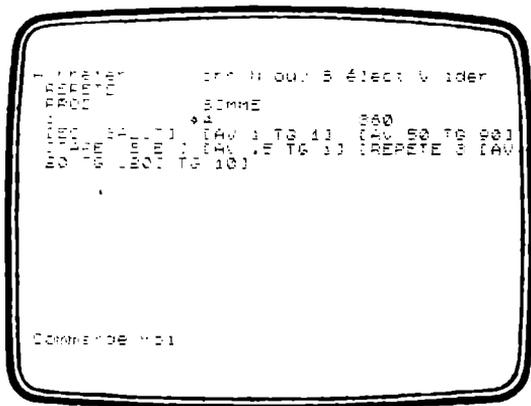
?EC LISTE PH ITEM 2 [MA TA SA] LISTE "MAIN [EST FROIDE]

?EC MP "TOTO LISTE PH "MANGE "DES [PATATES FRI- TES]

Option 5

La commande REPETE





Écrivez

REPETE 4 [AV 50 TG 90]

Exécutez.

Voilà une façon élégante de faire un carré ! REPETE a deux arguments : le premier est un nombre, le second une liste de commandes LOGO.

M> 8^{ème} chapitre

Écrivez

REPETE 360 [AV 1 TG 1]

Voilà une façon élégante de faire un cercle.

Au passage, nous avons le « théorème » du tour complet : c'est simple ; pour faire revenir la tortue à son départ, on doit commander : REPETE n [AV 10 TG angle] avec $n \times \text{angle} = 360$ et la tortue revient à son point de départ.

Ainsi, pour le carré, nous avons :

$$4 \times 90 = 360$$

Pour le cercle :

$$1 \times 360 = 360$$

M> 8^{ème} chapitre

Écrivez

REPETE PROD SOMME 4 4 SOMME 1 4 [AV 1 TG 1]

Sous LOGO

Essayez de créer vos propres cercles, ou parties de cercles, ou de faire des polygones.

— L'hexagone

REPETE 6[AV 20 TG 60]

— Le dodécagone

REPETE 12[AV 10 TG 30]

Option 6

L'instruction conditionnelle



M > ême chapitre

Écrivez

SI EGAL? HASARD 2 1 [EC "GAGNE] [EC "PERDU]

Exécutez.

HASARD 2 est égal à l'un des deux nombres 0 ou 1 ; à chaque fois que HASARD 2 rend 1, la condition EGAL? est vraie, donc on effectue ECris? GAGNE, sinon elle est fausse, donc on écrit PERDU.

M > ême chapitre

Écrivez

SI PLP? HASARD 2 1 [EC"GAGNE] []

Exécutez.

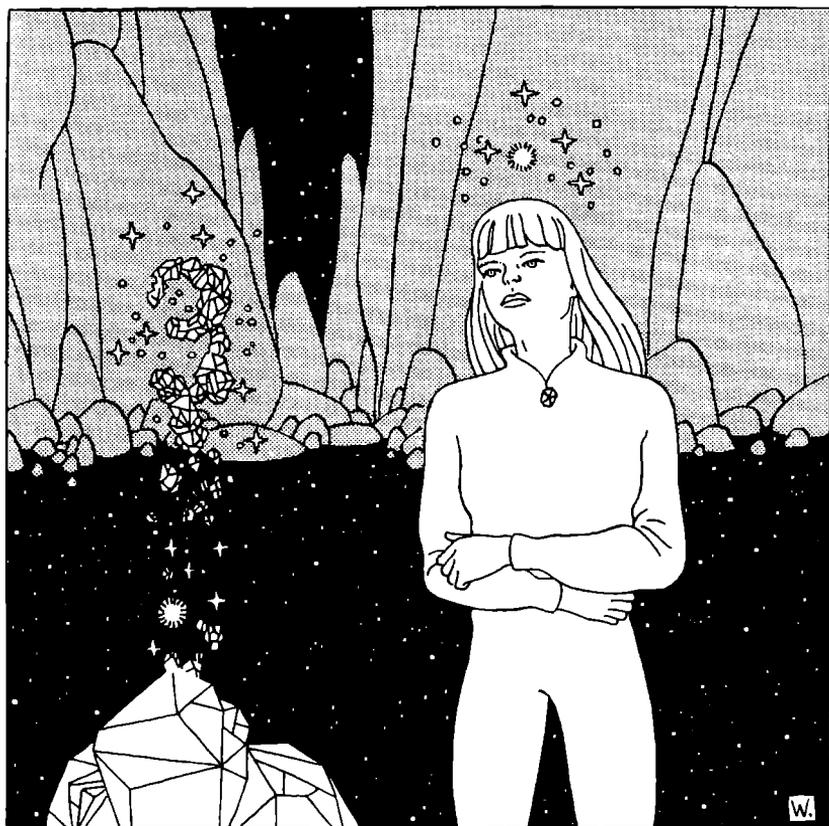
Cette expression n'écrit GAGNE que lorsque HASARD 2 est égal à 0 (Plus Petit que 1), sinon elle ne fait rien. En fait, elle exécute une expression neutre [] qui est la liste vide ; rien, en LOGO, peut être un objet vide.

EXERCICES

1. Écrivez une expression qui écrit PERDU ou GAGNE selon que le nombre choisi au hasard entre 0 et 1 est plus grand ou non que 0.
2. Écrivez une expression qui teste si un nombre est plus grand que 1 et, dans ce cas, écrit GAGNE.
3. En travaillant sous LOGO, essayez les expressions suivantes :
?SI PLG? HASARD 3 1 [AV 50] [RE 50]
?SI EGAL? HASARD 2 0 [] [EC "TOTO AV 50 TAPE "A "GAGNE]
?SI EGAL? MOT "TO "TO "TOTO [EC "C'EST EC "TOTO]

CHAPITRE 3

LA COMMANDE POUR



Nous allons maintenant programmer directement en LOGO. Les procédures que nous allons définir pourront servir pour les prochains OUTILS.

Procédure commande

La primitive POUR est une commande, elle permet de définir une procédure en lui donnant un nom. Exemple simple et courant : nous connaissons l'expression LOGO qui produit un carré (voir OUTIL 1), nous allons en faire une procédure :

```
?POUR CARRE ENTREE  
> REPETE 4[AV 50 TG 90] ENTREE  
> FIN ENTREE  
VOUS VENEZ DE DEFINIR CARRE
```

POUR indique à LOGO que le prochain MOT sera le nom d'une procédure et que le corps de la procédure sera tout ce qui est écrit jusqu'à FIN.

Le nom de la procédure peut rejoindre le contenu de l'environnement* LOGO, et la procédure se manipulera comme une primitive.

?CARRE

dessine un carré à partir de la position courante de la tortue. CARRE est une procédure commande, elle se suffit à elle-même pour être utilisée.

Procédure opération

Comme il existe des primitives opérations, on peut définir des procédures opérations ; on indique qu'elles doivent rendre un résultat, avec la commande RENDS*.

```
?POUR OPERATION ENTREE
```

·RENDS SOMME 3 7 **ENTREE**

·FIN **ENTREE**

VOUS VENEZ DE DEFINIR OPERATION

?EC OPERATION **ENTREE**

10

?EC SOMME 5 OPERATION **ENTREE**

15

Procédure avec arguments

Ces deux procédures n'ont pas d'arguments, tout comme les primitives VE ou CC, mais on peut décider de mettre des arguments aux procédures-commandes et aux procédures-opérations.

Pour mettre un argument à CARRE, nous allons nous servir de l'éditeur LOGO.

?ED [CARRE] **ENTREE**

(Rappelons que le crochet [se frappe **CNT** - **A** et le crochet] se frappe **CNT** - **Z**.)

Nous voyons écrit en blanc sur bleu :

POUR CARRE

REPETE 4 [AV 50 TG 90]

FIN

Ajoutons l'argument :COTE en plaçant le curseur un espace après CARRE à l'aide de la flèche  et en tapant :COTE.

POUR CARRE :COTE

REPETE 4 [AV 50 TG 90]

FIN

Maintenant, toujours à l'aide des flèches, allons placer le curseur sous le 5 de 50 et, en appuyant sur EFF, effaçons 50, puis tapons :COTE pour donner l'argument :COTE à la commande AV.

POUR CARRE :COTE

REPETE 4 [AV :COTE TG 90]

FIN

Nous avons ainsi changé le texte de la procédure CARRE. Il faut maintenant la redéfinir : appuyons simultanément sur **CNT** et **Q**.

CARRE est redéfini, l'ancienne procédure est oubliée.

?CARRE 40 **ENTREE**

dessine un carré de 40 pas de côté

?CARRE 60 **ENTREE**

un autre de 60 pas de côté, etc.

?ED **ENTREE**

Permet de revenir dans l'éditeur avec le contenu précédent.

Si on ne veut pas redéfinir la procédure CARRE, on peut quitter l'éditeur avec **CNT**-**C**.

Nous pouvons aussi définir des procédures opérations avec des arguments.

?POUR CUBE :NOMBRE **ENTREE**

>RENDS PROD :NOMBRE PROD :NOMBRE :NOMBRE **ENTREE**

>FIN **ENTREE**

VOUS VENEZ DE DEFINIR CUBE

?EC CUBE 2

8

?EC CUBE 3

27

Un exemple : la procédure explose

Nous allons étudier un autre exemple de procédure qui réalise une opération assez pratique.

Les primitives MD et MP ne fonctionnent que si elles ont une liste comme deuxième argument.

Nous allons créer des équivalents de ces primitives pour les mots.
 Nous les appellerons : MPMOT et MDMOT.

Il y a plusieurs solutions ; l'une d'entre elles consiste à faire exploser le mot en une liste plate, lui ajouter un caractère ou un mot, puis faire impluser la liste plate pour récupérer un mot.

Dans l'éditeur, écrivez

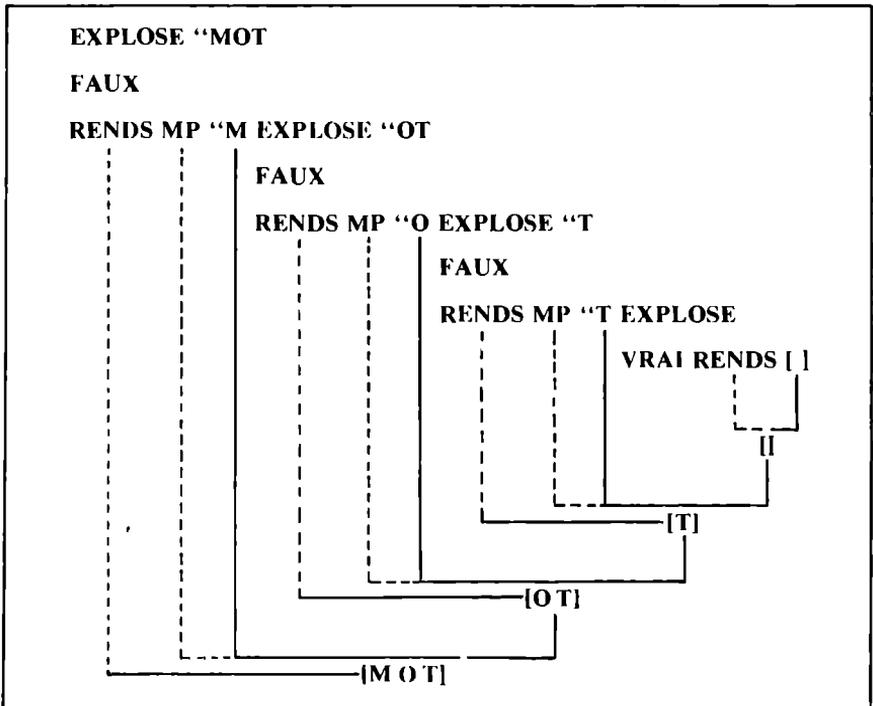
```
POUR EXPLOSE :MOT
SI VIDE? :MOT [RENDS[ ]
RENDS MP PREM :MOT EXPLOSE SP :MOT
FIN
```

Définissez par **CNT - Q**

```
?EC EXPLOSE "TARTESOFT
[T A R T E S O F T]
```

En rappelant, dans la procédure EXPLOSE, la procédure elle-même, chaque lettre du mot subit le même traitement !

Voici la simulation du fonctionnement du processus de résolution.



Maintenant, toujours dans l'éditeur LOGO, écrivons en dessous du texte de la procédure EXPLOSE :

```
POUR MPMOT :CARACTERE :MOT
RENDS IMPLOSE MP :CARACTERE EXPLOSE :MOT
FIN
```

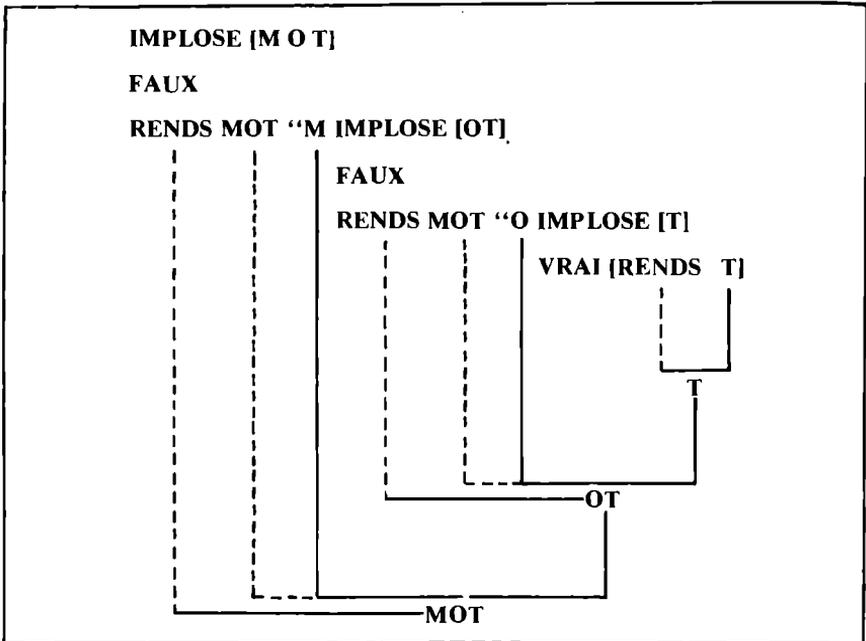
Notre procédure MPMOT fait appel à la procédure IMPLOSE (qui fait l'inverse de EXPLOSE) que nous n'avons pas encore écrite. La voici :

```
POUR IMPLOSE :LISTEPLATE
SI VIDE? SP :LISTEPLATE [RENDS PREM :LISTEPLATE]
RENDS MOT PREM :LISTEPLATE IMPLOSE SP :LISTEPLATE
FIN
```

Définissez **CNT-Q**

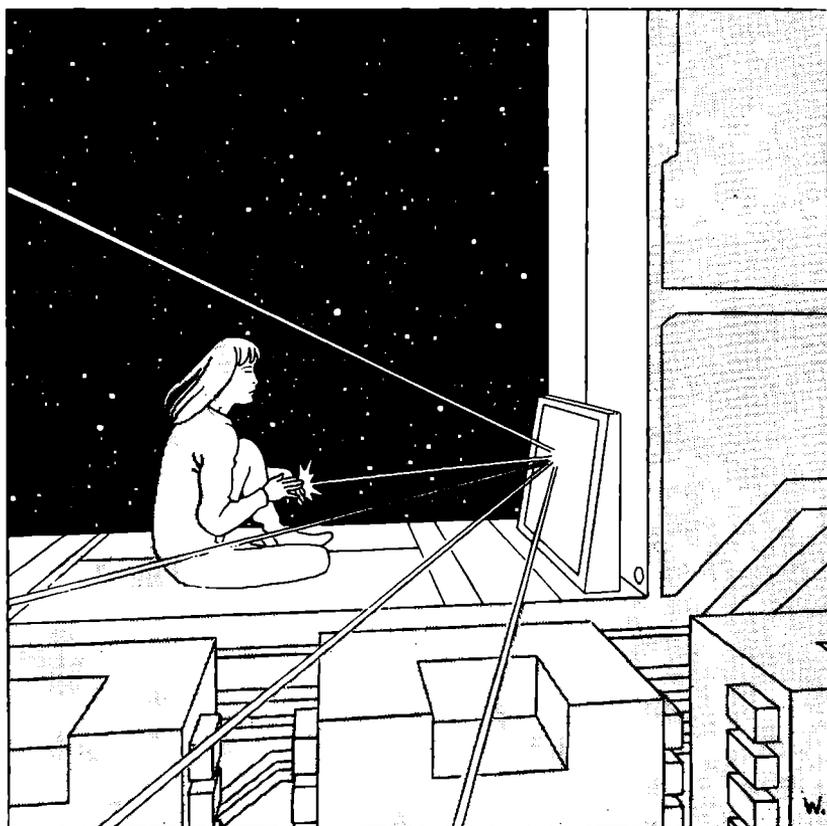
```
?EC IMPLOSE [T A R T E H A R D]
TARTEHARD
```

Fonctionnement de la procédure :



CHAPITRE 4

L'ANALYSEUR SYNTAXIQUE "SYNTA

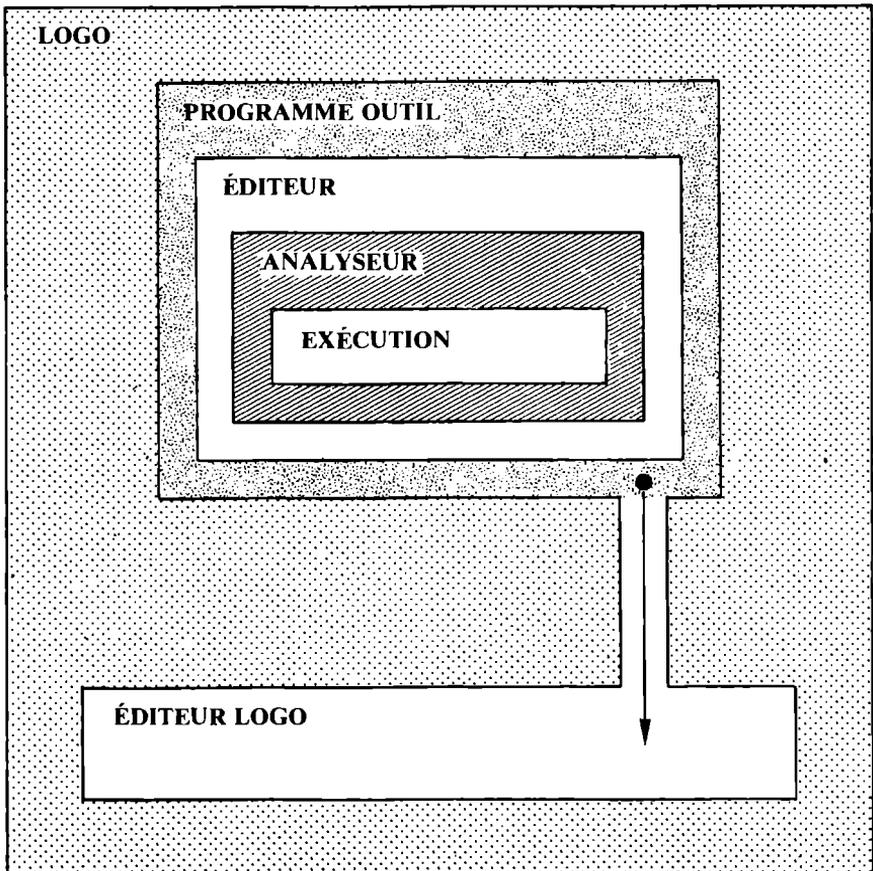


Comme vous en avez maintenant l'habitude, lancez l'exécution de ce programme par :
?OUTIL **ENTREE**

Les différents niveaux du programme

Vous êtes devant l'écran de l'éditeur du programme. La touche **RAZ** vous permet d'arrêter l'outil et de revenir à LOGO ; la touche **ENTREE** permet de passer au mode analyse (l'analyseur syntaxique) ; puis de ce mode-là, nous pouvons passer au mode exécution.

En voici le schéma :

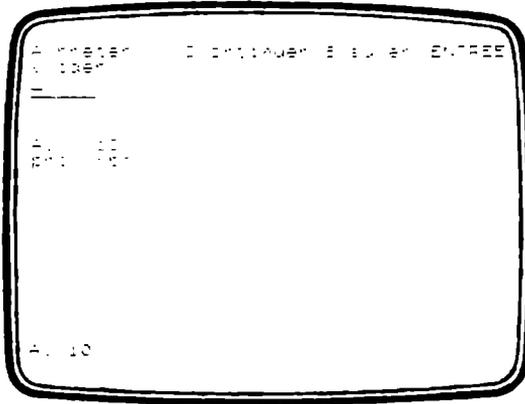


Un exemple

Écrivons donc dans l'éditeur :

AV 10 **ENTREE**

Nous en sommes maintenant au stade de l'analyse syntaxique de l'expression AV 10, c'est-à-dire que le programme cherche la nature des mots de l'expression tapée et des arguments écrits.



En l'occurrence, AV est une primitive : *pri*, et 10 un nombre : *nbr*. Il y a aussi les diminutifs pour CHOSE : *cho*, Procédure : *pro*, et les objets listes : *li*, et mot : *mot*.

Au-dessus, sont tracées des lignes qui rejoignent les primitives ou procédures à leurs arguments.

Le menu

A>rêter permet de revenir au niveau LOGO.

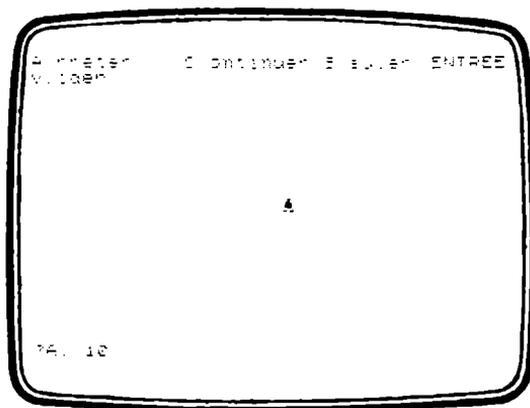
C>ontinuer permet de revenir à l'éditeur avec l'expression précédente et on peut alors ajouter d'autres expressions.

S>aver permet d'écrire l'expression dans l'éditeur LOGO.

V>ider permet de revenir au niveau Editeur mais sans expressions écrites.

Enfin, **ENTREE** fait passer au niveau de l'exécution.

Tapez sur **ENTREE**.



L'exécution est effectuée ; à ce niveau, l'écran est divisé en trois zones : le même menu qu'au niveau de l'analyse, une partie graphique et (en bas, en bleu sur fond jaune) le texte de l'expression et son exécution éventuelle après des commandes EC ou TAPE.

Les commandes du menu agissent de la même manière, et permettent de revenir au niveau éditeur.

L'intérêt essentiel pour vous est de créer maintenant vos propres expressions et de les faire analyser.

Quelques conseils

Commencez toujours vos expressions par une commande ; si un trait de rappel au-dessus de l'expression analysée tombe dans le vide, c'est qu'il manque un argument à la procédure ou à la primitive qu'il rejoint.

Vous êtes normalement habitué à cette notation ; c'est la même qui a servi (dans la première partie) à vous montrer et à vous décrire les expressions LOGO.

Les couleurs des traits de rappel sont importantes. Elles changent si l'expression prend de la profondeur, c'est-à-dire si vous faites appel à plusieurs primitives ou procédures OPERATIONS imbriquées.

Exemple

AV SOMME 10 DIFF 5 4

Vous pouvez imbriquer jusqu'à six fois les opérations les unes dans

les autres. Vous avez un peu de place mémoire pour créer quelques procédures que vous pouvez utiliser dans les expressions que vous allez taper.

Si vous exécutez une expression à laquelle il manque des arguments ou qui n'a pas de commandes correctement écrites, vous allez tomber sur une erreur LOGO. Dans ce cas, vous essayez de comprendre les messages LOGO qui vont maintenant vous suivre et vous poursuivre continuellement et vous relancez le programme par ?OUTIL.

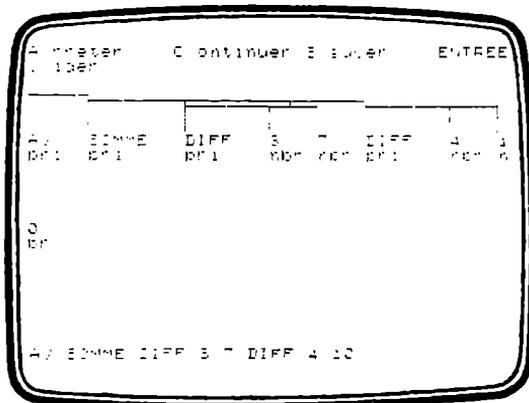
Pour commencer, écrivez des expressions que vous comprenez bien, ou des expressions que vous trouverez dans les pages précédentes et que vous n'avez pas bien comprises.

Des exemples pour vous entraîner

(Vous pouvez faire preuve d'initiatives, cela n'explose pas !)

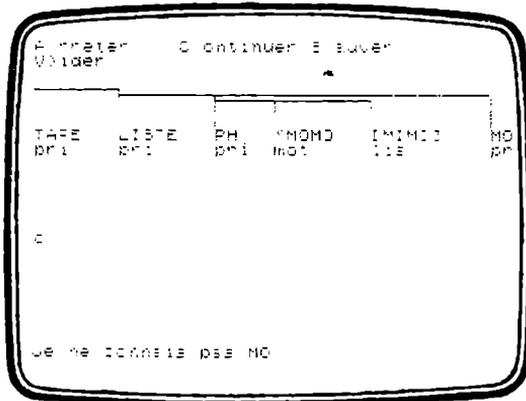
1. AV SOMME DIFF 3 7 DIFF 4 10
2. EC MOT "TARTE SP "DURE
3. DONNE "VALEUR "TATA EC PH "MA :VALEUR
4. Programmez une ou plusieurs des procédures du chapitre POUR, par exemple IMPLOSE EXPLOSE MPMOT et tapez TAPE MOT MOT "BEA "T MPMOT "L "ES
5. EC PROD DIV 1 2 RC 2
6. EC SIN PROD DIV 100 RC 2 10
7. AV SOMME DIFF 3 7 DIFF 4 10

donne :



équivalent à AV - 5, aussi équivalent à RE 5.

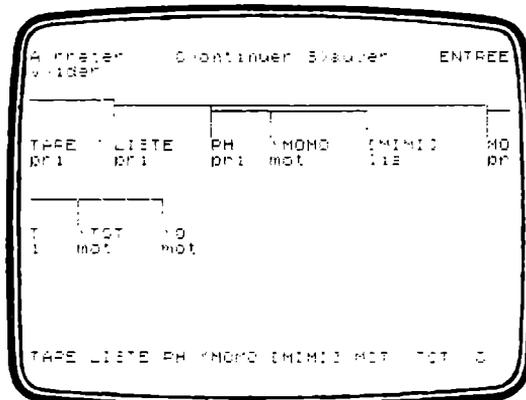
8. Voilà un exemple où l'expression est TAPE LISTE PH "MOMO [MIMI] MO"TOT "O.



En effet, MO n'étant pas une primitive, le programme cherche si une procédure MO existe ; comme il n'y en a pas, le programme exprime par « Je ne connais pas MO » qu'il n'a pas trouvé cette procédure.

C>ontinuez.

Pour changer MO en MOT,
cela donne :



```

Arrêter : 1) Continuer 2) Sauver
V)ider

EC PH *AZ PH (TOTO TATA)
PR1 PR1 mot PR1 115

EC PH *AZ PH (TOTO TATA)

```

Dans cette expression, il manque un argument au deuxième PH, un trait tombe sur rien.

Le message d'erreur LOGO serait :

PAS ASSEZ DE DONNÉES POUR PH

C>ontinuez, pour corriger.

```

Arrêter : 1) Continuer 2) Sauver ENTREE
V)ider

EC PH *AZ PH (TOTO TATA) TIT
PR1 PR1 mot PR1 115 mot

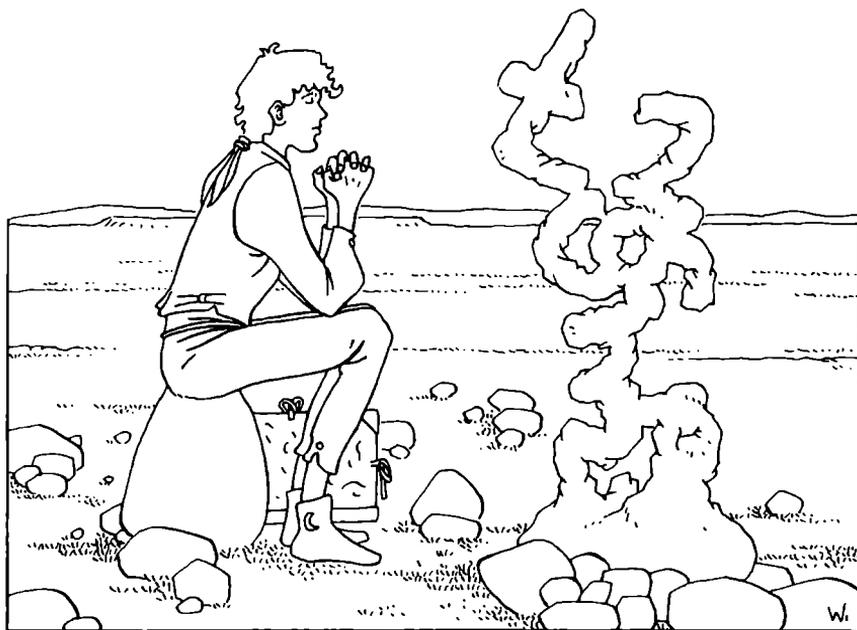
EC PH *AZ PH (TOTO TATA) TIT

```

Voilà l'expression correcte.

CHAPITRE 5

L'OUTIL DE VISUALISATION "VISUA

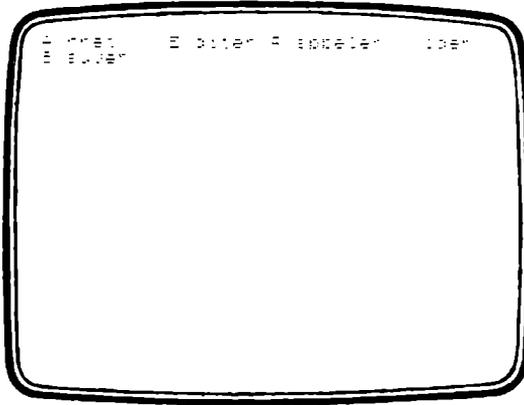


Vous chargez le programme par :

?RAMENE "VISUA

Vous lancez par :

?OUTIL



Le menu

Voici le menu qui vous est proposé :

A>rret!!! pour revenir sur LOGO.

E>diter pour passer à l'éditeur du programme.

R>appeler pour reprendre une expression qui aurait été déjà sauvée.

S>auver dans l'éditeur de LOGO.

V>ider pour vider l'éditeur du programme, et réécrire une nouvelle expression.

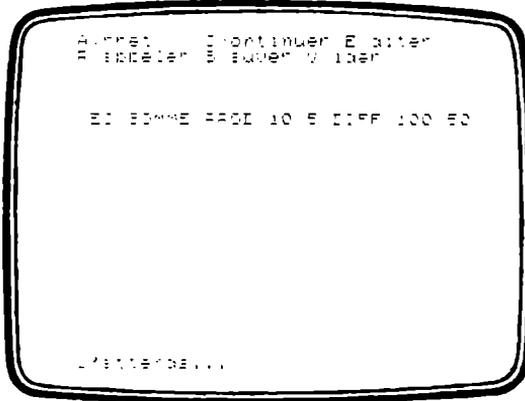
Après **E**, vous pourrez taper l'expression de votre choix, mais avec une restriction : vous ne pouvez pas utiliser vos propres procédures dans cette expression (il faut vous contenter des primitives).

La touche **ENTREE** valide votre expression (elle entre dans la mémoire de l'ordinateur) et fait passer le programme à la visualisation de l'évaluation.

Un exemple

Écrivez :

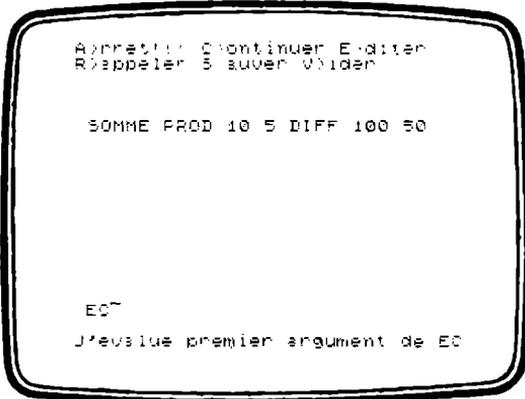
EC SOMME PROD 10 5 DIFF 100 50 **ENTREE**



J'attends... une des commandes, bien sûr.

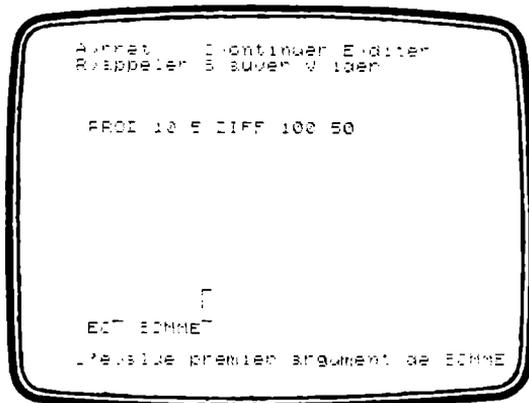
Le menu a changé :

C> continuer est justement la commande que nous allons utiliser.



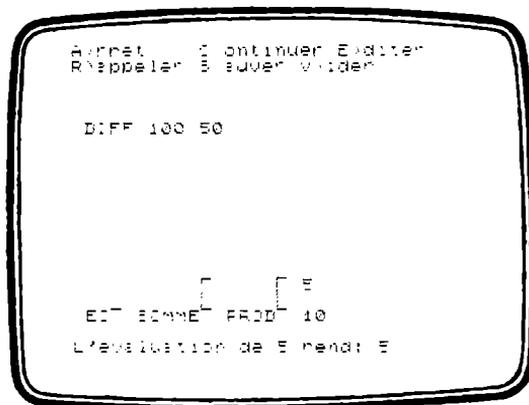
Le processus d'évaluation commence.

C> continuez.



EC n'a qu'un argument et SOMME en a deux, ce qui explique les traits.

- C> continuez,*
- C> continuez,*
- C> continuez,*
- C> continuez.*



La « vague » d'évaluation continue sous nos yeux et c'est ce que fait LOGO beaucoup plus rapidement.

En continuant, nous allons tomber sur la forme suivante :

```

A: arrêt : continuer E: aller
R: répéter B: boucle W: vide

- 50
- DIFF 100
EST BONNE 50

```

Maintenant le processus revient sur lui-même, c'est le retour de la « vague » jusqu'au résultat :

```

A: arrêt : continuer E: aller
R: répéter B: boucle W: vide

100

*
-
- la validation des EC 100 n'a rien retourné

```

D'autres exemples

```

A: arrêt : continuer E: aller
R: répéter B: boucle W: vide

- 100
- DIFF 10
- la validation des 100 n'est pas

```

Mais quel résultat cela va-t-il donner ?

L'expression arrête le programme, car :

EC SOMME 12 DIV 11 0, bien sûr, la division par zéro est strictement interdite.

EXERCICES

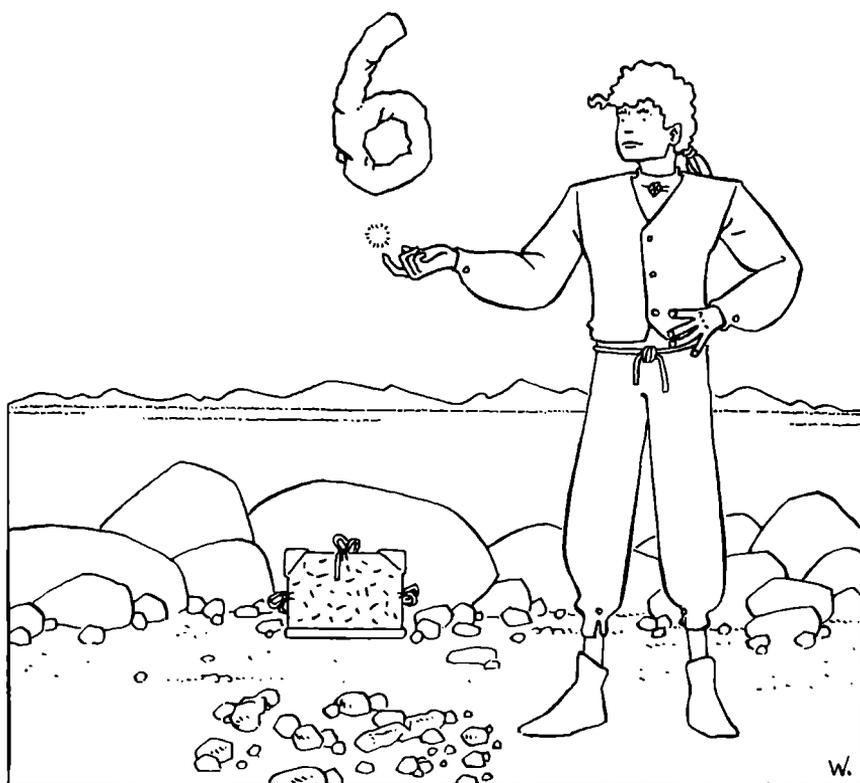
1. EC MOT "TOTO PH "TOTO "TITI

2. DONNE "VAR 5 EC PROD 5 :VAR

3. Reprenez les expressions écrites tout au long du chapitre 1 pour les visualiser.

CHAPITRE 6

OUTIL DE TRACE "TRACE"



Vous chargez le programme par :

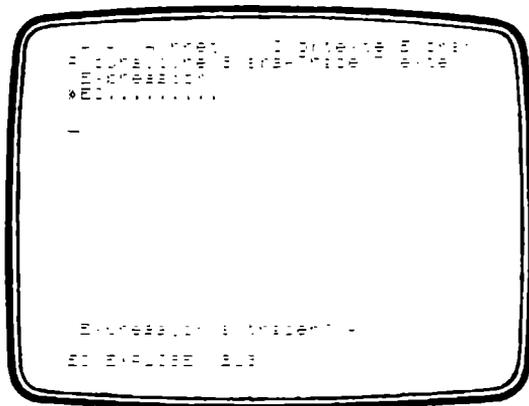
?RAMENE "TRACE

et bien sûr vous tapez :

?OUTIL

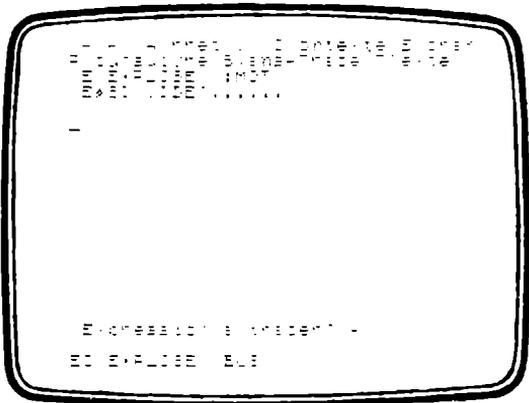
Exemple (pas à pas)

On nous propose maintenant de suivre «à la trace» l'exécution d'une procédure préalablement programmée. Nous allons suivre ici la trace de la procédure EXPLOSE.



La trace commence après la frappe de **ENTREE** ; l'étoile indique l'endroit où la trace se dessine.

Nous allons P>oursuivre.




```

- - - A l'intérieur du contexte E1E2E3
Pour suivre sans-trace le chemin
E D D DELETOUT :LISTE .....
E S S>SI VIDE :LISTE .....

-

Expression à tracer : -
EC DELETOUT "AZ (AZ ER (AZ))

dans DELETOUT : chose renvoyée - ER

dans DELETOUT : chose renvoyée - EC

dans DELETOUT : chose renvoyée - CC

```

```

(- - - A l'intérieur du contexte E1E2E3
Pour suivre sans-trace le chemin
E D D DELETOUT :LISTE .....
E S S>SI VIDE :LISTE .....

EC DELETOUT "AZ (AZ ER (AZ))

dans DELETOUT : chose renvoyée - ER

dans DELETOUT : chose renvoyée - CC

dans DELETOUT : chose renvoyée - CC

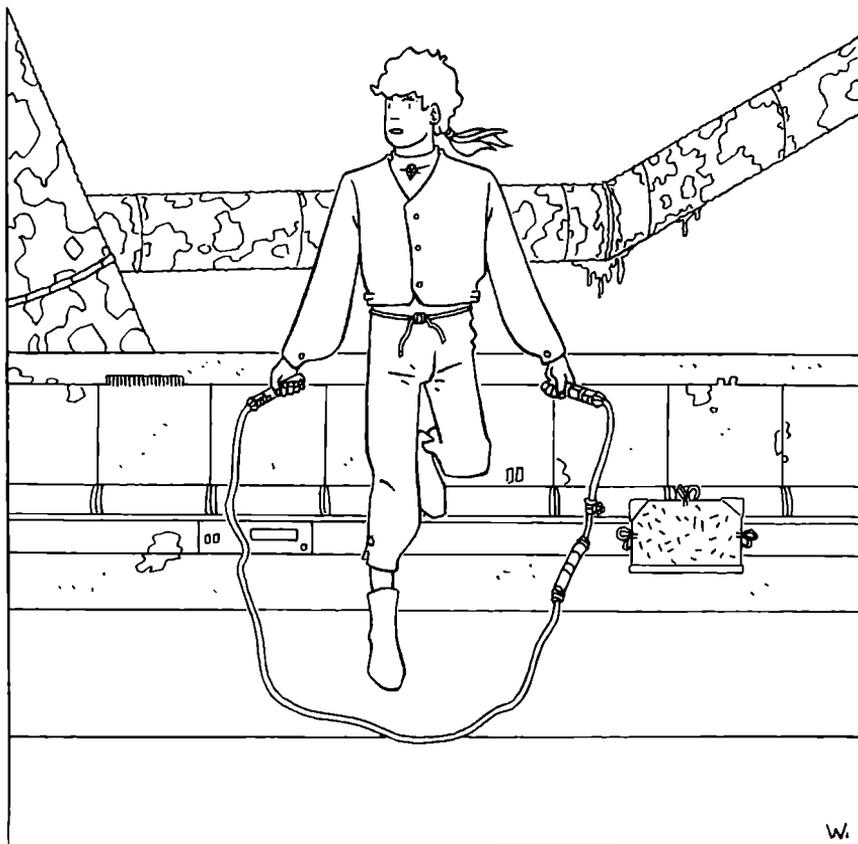
dans DELETOUT : chose renvoyée - CC

```

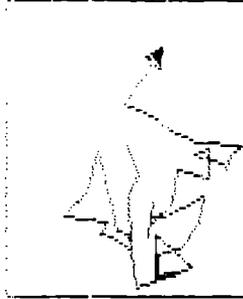
Il y a huit appels récursifs dans cette expression.

CHAPITRE 7

PROCÉDURES



Un petit programme amusant La promenade de la tortue



```
POUR PROMENADE  
PROME [-50 50] [50 -50]  
FIN
```

Nous allons faire errer la tortue dans une aire définie par les coordonnées de deux points, le Point en Haut à Gauche :PHG = [- 50 50], le Point en Bas à Droite :PBD = [50 - 50].

Ces points sont des arguments variables : il suffit de changer leurs valeurs pour modifier l'aire de promenade.

```
POUR PROME :PHG :PBD  
FOFT 4 FCT 7 ME 6  
CARRA :PHG :PBD  
PRO 20 760 :PHG :PBD  
FIN
```

La procédure PROME initialise les couleurs, dessine (procédure CARRA) l'aire où la tortue évoluera, puis lance la promenade de la tortue par PRO.

```
POUR CARRA :PHG :PBD  
LC FPOS :PHG BC  
FPOS LISTE PREM :PBD DER :PHG  
FPOS :PBD  
FPOS LISTE PREM :PHG DER :PBD  
FPOS :PHG  
LC ORIGINE BC  
FIN
```

La commande de la tortue FPOS (Fixe POSition) prend comme argument une liste de deux nombres et positionne la tortue au point de coordonnées correspondant. LC ORIGINE BC fait revenir la tortue au centre de l'écran.

```
POUR PRO :DD :AA :PHG :PBD
DONNE "POS-1 POS
SI CALPOS? [TG :A AV :D PRO :DD :AA :PHG
:PBD]
TG :A
PRO :DD :AA :PHG :PBD
FIN
```

PRO a pour arguments la distance maximum que peut parcourir la tortue à chaque « pas » et l'angle maximum fixé pour tourner. "POS-1 sert à mémoriser la position au début du calcul.

La procédure CALPOS? est un prédicat* qui calcule si le prochain trajet de la tortue reste dans l'aire de promenade.

Si elle rend "VRAI, alors la tortue tourne et avance, puis continue la promenade, si CALPOS? rend "FAUX, la tortue ne fait que tourner et continue sa promenade.

```
POUR CALPOS?
DONNE "D HASARD :DD
DONNE "A HASARD :AA
CALP :A :D
SI OU ( :POSX > PREM :PBD ) ( :POSX PR
EM :PHG ) [REND "FAUX]
SI OU ( :POSY > DER :PHG ) ( :POSY DER
:PBD ) [REND "FAUX]
REND "VRAI
FIN
```

"D et "A prennent les valeurs au hasard pour calculer le prochain trajet de la tortue.

Puis la procédure CALP calcule le trajet prévu et donne à "POSX et "POSY les valeurs qu'aurait la position de la tortue si elle bougeait.

```
POUR CALP :A :D
DONNE "CA ( CAP - :A )
DONNE "POSX PREM :POS-1 + ( :D * SIN :CA
)
DONNE "POSY DER :POS-1 + ( :D * COS :CA
)
FIN
```

Le retour à CALPOS? prend les éléments calculés par CALP et scrute si le trajet de la tortue sort ou non de l'aire ; s'il sort, "FAUX est rendu au SI de la procédure PRO, sinon CALPOS? rend "VRAI et la promenade continue *ad vitam aeternam*.

Quelques procédures pour les mots et les listes

Nous savons maintenant qu'une procédure peut s'appeler elle-même. Le processus s'appelle la RÉCURSIVITÉ*, un mot un peu barbare qui tient de récurrence, c'est-à-dire qui se répète.

Voici une procédure-commande récursive amusante.

Lorsqu'on écrit EC (3 EC 4), on obtient ceci :

```
4
3
```

En effet, l'évaluation de l'intérieur des parenthèses est prioritaire. Ce principe peut être utilisé pour écrire un mot à l'envers sur l'écran.

```
POUR TAPEINV :MOT
SI VIDE? :MOT [STOP]
TAPE ( PREM :MOT TAPEINV SP :MOT )
FIN
```

Lorsqu'on lance cette procédure, par exemple avec ?TAPEINV "ATOL, on obtient LOTA?.

Le processus des appels successifs est ici assez simple : on doit faire un appel récursif en changeant au moins un des arguments de la procédure (sinon, elle se répéterait indéfiniment. Ici on enlève la tête de :MOT). Et bien sûr on arrête le processus d'appel lorsqu'on a vidé :MOT à coups de SP.

MOTINV

La procédure MOTINV est une procédure opération qui rendra le mot inversé ; comme dans TAPINV, ce mot pourra bien sûr être affiché, mais mieux, il existera en tant que tel dans la mémoire de l'ordinateur.


```

POUR DELETC :MOT :LISTEPLATE
SI VIDE? :LISTEPLATE [RENDS ]]]
SI EGAL? PREM :LISTEPLATE :MOT [RENDS DE
LETC :MOT SP :LISTEPLATE]
RENDS PH PREM :LISTEPLATE DELETC :MOT SP
:LISTEPLATE
FIN

```

On arrête le processus lorsque la procédure est vide certes, mais il faut reconstituer la liste par-derrière. Ce n'est pas une mince affaire.

Voyons cela :

PH s'en charge toutes les fois que le mot comparé est différent du mot recherché, et lorsqu'on trouve ce mot, on le saute...

Plus difficile maintenant !

```

POUR DELETCOUT :ISTE :LISTE
SI VIDE? :LISTE [RENDS :LISTE]
SI MOT? :LISTE [RENDS :LISTE]
SI EGAL? PREM :LISTE :ISTE [RENDS DELETC
UT :ISTE SP :LISTE]
RENDS MP DELETCOUT :ISTE PREM :LISTE DELE
TOUT :ISTE SP :LISTE
FIN

```

Cette procédure plonge dans une liste complexe jusqu'au mot et enlève toutes les occurrences de la valeur de :ISTE à tous les niveaux de cette liste.

Essayons à la main.

(Voir schéma page suivante).

Ce procédé est très courant en LOGO : on représente un problème par une liste complexe et on va à la « pêche » à l'intérieur.

EXERCICES

1. Écrire une procédure qui compte tous les caractères de tous les mots d'une liste plate.
2. Écrire une procédure qui compte tous les caractères de tous les mots d'une liste complexe.

DELETOUIT "API [POM API [POI [API]]POP]

Faux
Faux
Faux

RENDS MP DELETOUIT "API "POM DELETOUIT "API [API [POI [API]]POP]

Faux
Vrai RENDS
"POM"

Vrai RENDS DELETOUIT "API [POI [API]]POP]

Faux
Faux
Faux

RENDS MP DELETOUIT "API [POI [API]]DELETOUIT "API [POP]

Faux
Faux
Faux

RENDS MP DELETOUIT "API

"POP DELETOUIT
"API II]

Faux
Vrai RENDS

"POP"

Vrai RENDS II]

RENDS MP DELETOUIT "API [API] DELETOUIT "API II]

Vrai RENDS II]

III]

Faux
Faux
Vrai RENDS DELETOUIT "API II]

Vrai RENDS II]

Annexes

Réponses aux exercices

INITIA OPTION 1

1. Par exemple :

AV 50 TD 90

Réexécuter trois fois.

2. Par exemple :

AV 50 TD 90 TD 10 TD 10 TD 10

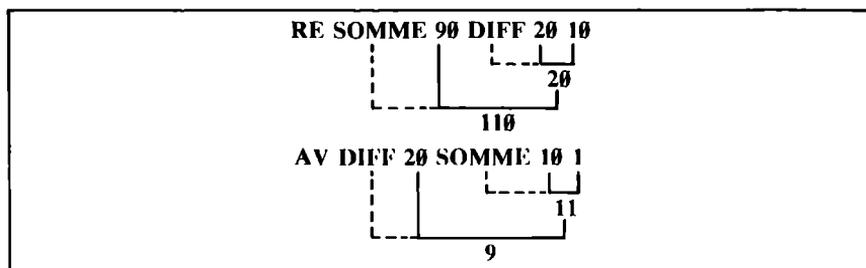
Réexécuter deux fois.

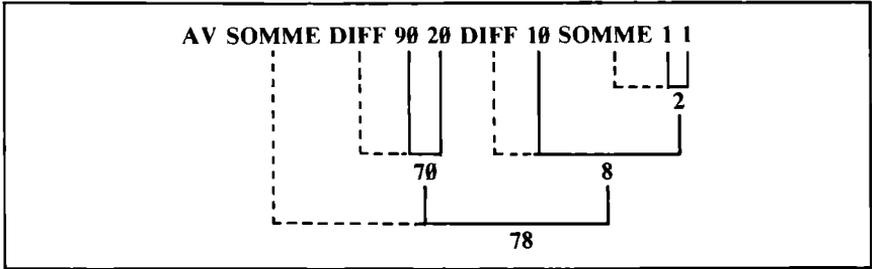
3. AV 10 TG 90 AV 10 LC AV 10 BC

Réexécuter trois fois.

INITIA OPTION 2

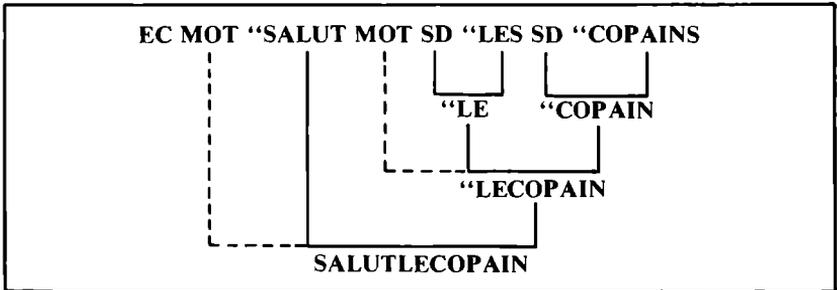
AV DIFF 90 10





INITIA OPTION 3

- EC "BONJOUR EC SD "LES EC SD "COPAINS
- TAPE PREM "SALUT TAPE PREM "LES TAPE PREM "COPAINS
- EC PREM SP SP "COPAINS
- EC SP SD "SALUT
- EC SP SP SP SP SD "BONJOUR

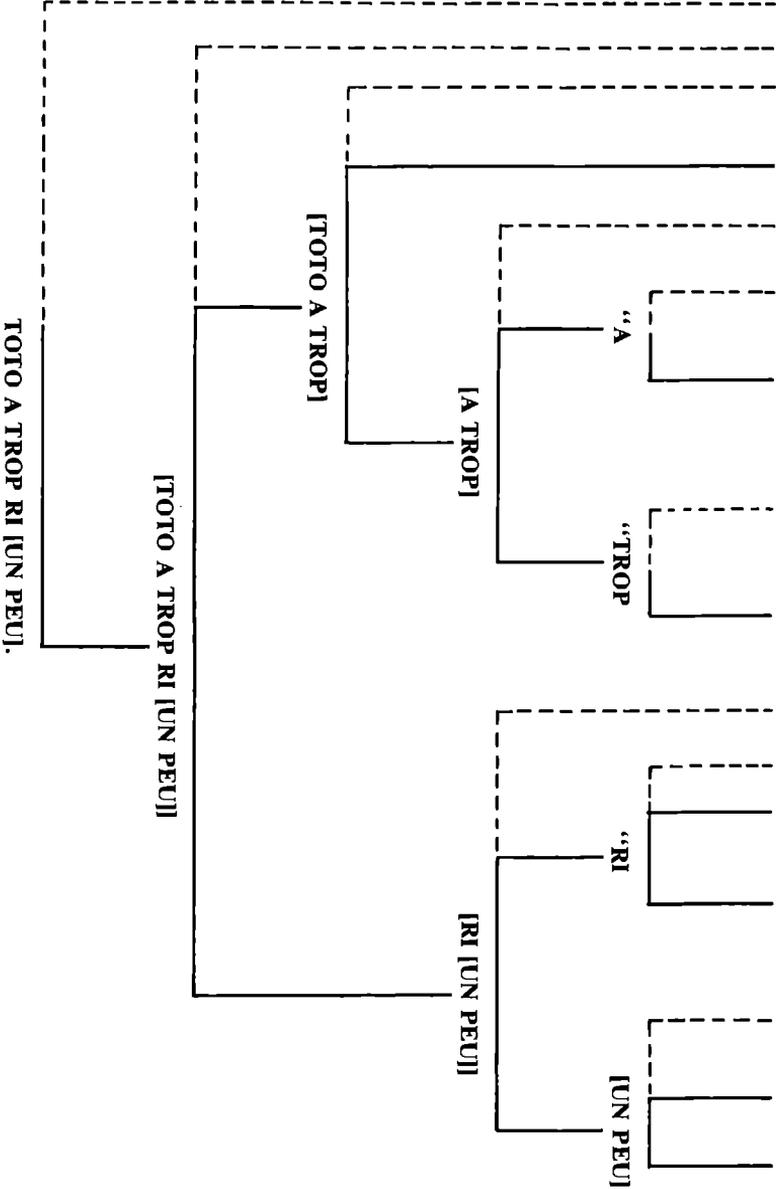


- EC MOT MOT "LES SD SD SD SD "BONJOUR MOT
- SD SD SD SD "BONJOUR PREM "SALUT
- EC MOT SD "LES MOT SP SP SP "BONJOUR MOT DER SD "COPAINS
- SP SD SD "SALUT
- FCFT 3 FCT 1 EC MOT HASARD 79 MOT
- SP SP SP "BONJOUR PREM "SALUT

INITIA OPTION 4

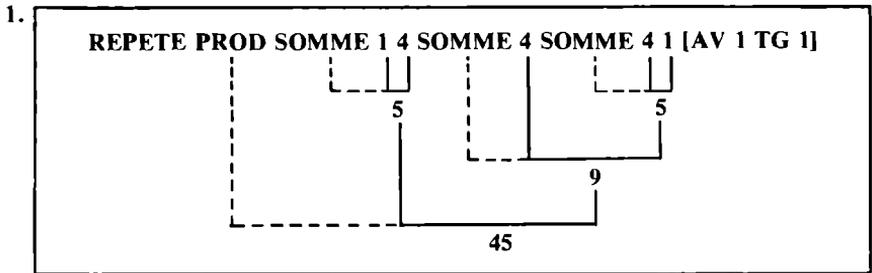
- EC PH PH PREM [A RI TROP] DER ITEM 2 [A RI TROP]
- PH DER "TOTO DER "PEU
- EC PH PH "TOTO ITEM 2 [A RI TROP]
- PH DER [A RI TROP] "PEU

— EC PH PH “TOTO PH PREM [A RI TROP] DER [A RI TROP] PH ITEM 2 [A RI TROP] LISTE “UN “PEU



TOTO A TROP RI [UN PEU].

INITIA OPTION 5



R > exécutez

$$45 \times 2 = 90^\circ$$

2. REPETE 360 [AV 1 TG 1] REPETE 360 [AV 0.5 TG1]

3. REPETE 4 [EC "SALUT] REPETE SOMME 1 SOMME 1 4
[EC "BYE] REPETE SOMME 360 360 [AV 1 TG 1]

INITIA OPTION 6

SI EGAL? ITEM HASARD 2 [EC "PERDU]

ITEM 1 [EC "GAGNE]

[EC "GAGNE] [EC "PERDU]

SI PLG? 2 1 []

POUR

POUR MDMOT :CAR :MOT

RENDS IMPLOSE MD :CAR EXPLOSE :MOT

FIN

POUR MP-MOT-OU-LISTE :AJOUT :OBJET

SI LISTE? :OBJET [RENDS MP :AJOUT :OBJET]

SI LISTE? :AJOUT [TAPE [MP-MOT-OU-LISTE N'AIME PAS] EC :AJOUT
LOGO]

RENDS MPMOT :AJOUT :OBJET

FIN

La primitive LOGO fait arrêter le programme et revient au niveau LOGO, c'est-à-dire au point d'interrogation. On peut créer des primitives qui donnent des messages d'erreur si elles sont mal employées.

VISUA

L'expression évaluera 100 en fin de parcours.

PROCÉDURES

```
POUR COMPTOUTLIP :LISTEPLATE
SI "IDE" SF :LISTEPLATE [RENDS COMPTMOT
PREM :LISTEPLATE]
RENDS SOMME COMPTMOT PREM :LISTEPLATE 0
OMTOUTLIP SF :LISTEPLATE
FIN
```

Le test d'arrêt se fait cette fois s'il reste un mot dans la liste et on REND son nombre de lettres.

```
POUR COMPTETOUT :LISTE
SI "IDE" :LISTE [RENDS 0]
SI "MOT" :LISTE [RENDS COMPTMOT :LISTE]
RENDS SOMME COMPTETOUT PREM :LISTE COMPT
ETOUT SF :LISTE
FIN
```

C'est le même processus que pour DELETOUT, mais, cette fois, on fait la SOMME, car on REND un nombre.

```
POUR EXPLOSE :MOT
SI "IDE" :MOT [RENDS 0]
RENDS MP PREM :MOT EXPLOSE SF :MOT
FIN
```

C'est une procédure connue.

```
POUR COMPTMOT :MOT
RENDS COMPT EXPLOSE :MOT
FIN
```

Cette procédure n'apporte pas de traitement particulier, mais elle rend, grâce à son nom, la lecture de COMPTETOUT plus aisée.

ARGUMENT

Nom des variables que l'on donne comme entrée dans une primitive ou une procédure LOGO.

COMMANDE

Primitive ou procédure LOGO qui s'exécute sans rendre d'objet. LOGO, EC, TAPE, AV sont les primitives-commandes.

DONNÉE

Valeur que l'on donne à un argument.

ÉDITEUR

Programme qui sert à saisir ce que l'utilisateur tape sur le clavier ; dans notre cas, chaque outil possède un éditeur qui lui est dédié, et LOGO possède lui-même son propre éditeur.

ENVIRONNEMENT

Ensemble des possibilités que donne un système informatique à un utilisateur. Pour LOGO, l'environnement est très important, car, en créant des procédures, l'utilisateur ne cesse de le modifier.

ÉVALUATION

Processus qui permet à LOGO de comprendre et d'exécuter ce qu'on lui demande.

EXPRESSION

Suite de primitives et de procédures avec toutes leurs données écrites en LOGO.

MENU

Écran qui propose (souvent au début d'un programme) un choix entre différentes possibilités. Ainsi, les programmes OUTIL proposent des menus de «commandes».

OBJET

LOGO reconnaît deux types d'objets : le MOT qui est une suite de caractères et la LISTE qui peut être une composition d'objets LOGO. Le nombre est un mot particulier qui a une valeur directement affectée. (12 est le mot "12, lequel a pour valeur le nombre entier égal à douze.)

OPÉRATION

Primitive ou procédure LOGO qui rend un résultat. Dans EC SOMME 10 10, l'opération SOMME rend 20 à la commande EC.

PRÉDICAT

Forme particulière d'expression LOGO de type « opération » qui rend les mots "VRAI ou "FAUX. Un prédicat est utilisé derrière l'expression conditionnelle SI. Il est recommandé, lorsqu'on programme une procédure prédicat, de la finir par ? (comme les primitives prédicat PLG?, PLP?, EGAL?, VIDE?...).

PRIMITIVE

Mot LOGO dans le contenu initial du langage. Certaines primitives sont primordiales comme SP, SD, MP, MD, LISTE, SOMME ; d'autres pourraient être programmées à partir de celles-ci comme ITEM, COMPTE.

PROCÉDURE

Élément de base d'un programme écrit en LOGO. Une procédure se maintient en mémoire et peut être sauvegardée. Les programmes LOGO sont un ensemble de procédures liées les unes aux autres. La procédure se crée par la commande POUR.

RENDS

Commande LOGO qui sert à définir des procédure-opérations. Lorsqu'une procédure contient un ou plusieurs RENDS, l'objet donné comme argument à RENDS devient le résultat de la procédure. Ce résultat sera transmis à la primitive ou la procédure appelante.

SAUVEGARDE

Action de mettre un programme en mémoire de masse : disquette, cassette... Nous vous conseillons de sauvegarder régulièrement vos propres programmes, toutes les demi-heures ou toutes les heures de travail ; au-delà, vous risquez de ne plus vous souvenir clairement de ce que vous avez fait.