PROCEDURES LOGO

DISCIPLINE	Initiation à la programmation.
SUJET	Ensemble de cinq logiciels de programmation LOGO : GENE, ECLUSE, ASCEN 2, PLISTE, TEXTE-DEFINIS.
NIVEAU	Tout public.
AUTEUR	A. MYX.
EDITEUR	MEN/CNDP.
LANGUE D'ECRITURE	LOGO.
OBJECTIFS	Mettre en œuvre des applications et des procédures en vue d'une initiation à la programmation en LOGO.

PROCEDURES LOGO - GENE

FICHE PEDAGOGIQUE

Pour que l'enfant soit en relation véritable avec l'informatique et qu'il se l'approprie dans l'autonomie, il est nécessaire qu'il ait accès à des activités de programmation. La difficulté est alors de créer, de développer des ensembles pédagogiques devant prendre en compte :
- La nécessité d'intégrer l'action de programmation à une activité plus globale dans le cadre de l'éveil.
- La nécessité d'obtenir un résultat pour lequel l'ordinateur joue un rôle non trivial ;
- La nécessité d'offrir aux maîtres plusieurs niveaux d'accès pour un même ensemble pédagogique.

Une première recherche s'est donc orientée vers la production d'ensembles pédagogiques permettant aux maîtres de guider les élèves du cycle moyen dans la réalisation de simulation d'objets techniques sur écran graphique. A terme, un tel complexe pédagogique devra comprendre :
- La description et l'analyse de l'objet technique, thème de la simulation.
- Différentes propositions de démarches en fonction du degré de complexité retenu lors de l'étude de l'objet et de la construction d'algorithmes rendant compte de la simulation.
- Une documentation informatique permettant de disposer de toutes les briques (procédures LOGO, sous-programmes BASIC...) nécessaires, en fonction du niveau d'entrée des élèves, pour mener à terme le programme de simulation.

FICHE OPERATOIRE

• DOG (définition d'objet graphique)
DOG "nom [liste de définition]
permet de définir un objet en lui donnant un nom (mot logo). La liste de définition doit impérativement être constituée, dans l'ordre :

1) d'un nom de forme :
   RECTPL (rectangle plein)
   RECTV (rectangle vide)
   SEG (segment défini par un point, sa longueur, son cap)
   BIPT (segment défini par ses deux extrémités)
2) d'une couleur (un nombre compris entre — 8 et + 7).
3) de la colonne (du point en bas à gauche pour les rectangles).
4) de la ligne (du point en bas à gauche pour les rectangles).
5) de la longueur (du côté horizontal pour les rectangles, du segment pour les segments).
6) de la hauteur pour les rectangles ou du cap de la tortue pour les segments (0 pour les verticales, 90 pour les horizontales).

Exemples :
a) DOG "MOBILE [RECTPL 3 0 0 26 42] définit un objet de nom "MOBILE", de forme "rectangle plein", de couleur jaune, dont le point en bas à gauche est à l'origine (0,0), de longueur 26 et de hauteur 42.
b) DOG "TROT [SEG 7 - 30 20 40 36] définit un objet de nom "TROT, de forme segment, de couleur blanche, dont le point de départ est en -30 (colonne), 20 (ligne), de longueur 40, formant un angle de 36 degrés avec la verticale.

N.B. : Les objets sont mémorisés sous forme d'une liste correspondant à la définition plus la couleur de fond :
?EC CHOSE "MOBILE
RECTPL 3 0 0 26 46 -5
?EC CHOSE "TROT
SEG 7 -30 20 40 36 -5

• FCFOND (fixe la couleur de fond)
  FCFOND nombre
permet de changer la couleur de fond des objets définis dans la suite.

• DESSINE (dessine l'objet "nom dans l'état défini).
  DESSINE "nom
Exemple :
? DESSINE "MOBILE
? DESSINE "TROT

• EFFACE "nom : affiche "nom dans la couleur de fond.

• DEP (déplacement)
DEP "sens "nom nombre de pas
"sens peut être "HAUT, "BAS, "GAUCHE ou "DROITE (l'initiale suffit).
"nom doit être un nom d'objet graphique déjà défini et dessiné.

Exemple :
? DEP "D" MOBILE 20
déplace l'objet "MOBILE de vingt pas vers la droite.
N.B. : la liste de définition de l'objet que l'on déplace est modifiée à chaque pas, de manière à correspondre en permanence à la situation réelle de l'objet.

Exemple :
? EC CHOSE "MOBILE
RECTPL 3 20 0 26 42
Le numéro de la colonne du point en bas à gauche est maintenant 20.

• SAUTE "sens "nom nombre-de-pas : idem mais déplacement en une fois.

• ROT (rotation autour du point de définition, valable uniquement pour les segments définis par SEG)
ROT "nom "deg "pas
"nom : nom d'un objet graphique (segment).
"deg : nombre de degrés dont on veut faire tourner le segment.
"pas : nombre de degrés de chaque rotation élémentaire.

Exemple :
ROT "TROT 360 6
fait tourner le segment "TROT (par pas de six degrés) de 360 degrés au total, soit soixante rotations de six degrés (dans le sens des aiguilles d'une montre...!).

N.B : Pour faire tourner un segment dans le sens inverse des aiguilles d'une montre, il suffit d'utiliser un nombre de degrés négatif.

• DUPLIQUER (duplique un objet sous un autre nom).
DUPLIQUER "nom1 "nom2
définit un nouvel objet de nom "nom2, qui a la même liste de définition que l'objet "nom1 déjà défini.

Exemple :
? DUPLIQUER "TROT "MIN
? ECRIS CHOSE "MIN
SEG 7 -30 20 40 36 -5

PROCEDURES UTILITAIRES

• FORME COUL COL LIG LONG HAUT COULFD
FORME "nom
COUL "nom
COL "nom
LIG "nom
LONG "nom
HAUT "nom
COULFD "nom
rendent la valeur de la propriété correspondante ; permettent en particulier de construire des tests concernant les caractéristiques de l'objet "nom.

Exemple :
SI EGAL ? HAUT "TROT O [ROT "MIN 6 6]
fait tourner le segment "MIN de six degrés en une seule fois si le cap du segment "TROT est égal à 0.

- CFORME (change la forme de l'objet "nom).
  CFORME "nom "forme
- CCOUL (change la couleur de l'objet "nom).
  CCOUL "nom coul
- CCOL (change le numéro de la colonne de l'objet "nom).
  CCOL "nom n

fixe à n le numéro de colonne (n positif ou négatif).

- CLIG (change le numéro de la ligne de l'objet "nom).
  CLIG "nom n

fixe à n le numéro de ligne (n positif ou négatif) de "nom.

- CLONG (change la longueur de l'objet "nom.
  CLONG "nom longueur
- CHAUT (change la hauteur ou le cap de l'objet "nom).
  CHAUT "nom hauteur
- CCOULFOND (change la couleur de fond)
  CCOULFOND "nom nouvelle-couleur-de-fond
- AIDE "nom proc : donne un résumé très succinct des informations concernant ces macro-primitives.
- SOS : affiche toutes les aides sur toutes les macro-procédures.

PROCEDURES LOGO - ECLUSE

FICHE PEDAGOGIQUE

• Le type d'objet technique étudié.
Pour que la simulation sur l'écran soit à la fois possible et intéressante, les conditions minimales doivent être réunies :
- la nécessité pour l'objet, thème de la simulation, d'être représentable de manière graphique par un schéma figurant la réalité de l'objet et son fonctionnement ;
- la nécessité que les fonctions mises en jeu puissent être décrites sous forme d'algorithmes séquentiels, c'est-à-dire de processus dont les étapes bien définies se succèdent dans le temps. Un tel algorithme doit être traduit sous forme d'une succession d'images graphiques entre lesquelles les transformations ne mettent en jeu qu'un nombre restreint d'éléments de l'image.
Afin que la démarche garde tout son sens, la simulation doit être étudiée et réalisée par les enfants. Ce qui impose que les algorithmes mis en œuvre puissent être compris et si possible élaborés par eux.
• L'exemple de l'écluse.
Après une étude de l'écluse resituée dans son contexte et en liaison avec la géographie (rivières, canaux...) et/ou la physique (vases communiquants), son fonctionnement est examiné. Le premier travail consiste à trouver une schématisation (représenter la réalité) et non une modélisation.
Le temps, la succession des actions vont créer le dessin animé souhaité. Pour l'écluse, ce sont par exemple :
- ouverture/fermeture des portes amont et aval ;
- ouverture/fermeture des vannes amont et aval ;
- trajets horizontaux du bateau dans les différents biefs ;
- montée/descente de l'eau dans le bassin.
• D'autres objets techniques...
La démarche qui prend comme point de départ un "objet" spécifié ici par l'écluse, pourrait tout aussi bien être menée avec d'autres objets techniques : ascenseur, grue, machine à laver, etc...
Notons que, si l'utilisation permettait seulement d'aborder, de mieux connaître un objet technique, ce serait suffisant pour justifier son utilisation.
Mais il ne faut pas rejeter les possibilités d'application à d'autres domaines, moyennant des transformations plus ou moins importantes.
On peut créer par exemple :
- des simulations de tissage ; algorithmes de tissage pour visualiser certains effets voulus.
- des simulations de pavage du plan : création de motifs, rôle des transformations géométriques mises en jeu.
- des représentations de modèles de circulation de l'information dans une calculette.
- etc.

FICHE OPERATOIRE

• Premier examen du logiciel.

Le logiciel ECLUSE est réalisé à partir de GENE.
Rappelons encore que le but est ici de faire réaliser par l'enfant un "éclusage" correct correspondant à un algorithme préalablement élaboré.

Lancer la procédure # ECLUSAGE.
Logo exécute alors un éclusage amont → aval.
Le logiciel (ensemble de macro-primitives présentées ici) ne consiste pas à lancer cette procédure d'appel !
Toutes les procédures dont le nom est précédé d'un # correspondent à l'une des nombreuses possibilités pour valider graphiquement un algorithme d'éclusage.

Travail attendu des élèves dans le cas de l'écluse.

a) Présentation du lot de macro-procédures.
A la demande des enfants, des procédures de dessin et d'animation sont données. Le choix du maître dans les différents lots se fait en fonction de leurs propositions.

b) Animation en mode immédiat.
La présentation du lot de macro-procédures est suivie, bien évidemment, de jeux graphiques libres pour en assurer une bonne maîtrise. Ces procédures permettant de dessiner sur l'écran la coupe de l'écluse et le bateau.

L'animation d'un éclusage consiste à écrire une suite d'ordres nécessaires pour obtenir la simulation graphique d'un passage de bateau.

c) Structuration de l'algorithme.
Un état initial dessiné, l'éclusage peut se décomposer en actions élémentaires. Exemple :

```
Pour # ECLUSAGE
# DEPART
# REMPLIR
# PASSAGE1
# VIDER
# PASSAGE2
FIN
```

Liste des macro-procédures
FOND
La procédure FOND définit tous les objets graphiques utiles et dessine les parties en eau (amont, bassin, aval).
Le bassin est en position basse.
Le fond est fixé à la couleur noire.

```
          AMONT      OUVERTE
VANNE [                     ]
          AVAL       FERMEE
```

BASSIN [A REMPLIR]
BASSIN [A VIDER]

Si le bateau est dans le bassin, ces deux procédures entraînent également la montée ou la descente de ce bateau.

BATEAU [(DESSINE / EFFACE) (HAUT / BAS) (AMONT / BASSIN / AVAL)]

Ainsi, la commande BATEAU [DESSINE HAUT BASSIN] dessine-t-elle un bateau dans le bassin dont l'eau est en position haute.

La liste [DESSINE HAUT BASSIN] est un argument possible de la commande bateau.

Les procédures # REMPLIR, etc., sont elles-mêmes construites à partir des macro-procédures citées plus haut.

Créer des prédicats.

```
?ECRIS FERME? [PORTE AMONT]
VRAI
?ECRIS FERMEE? [VANNE AMONT]
FAUX
?ECRIS OUVERTE? [PORTE AVAL]
FAUX
?ECRIS FERMEE? [VANNE AVAL]
VRAI
?ECRIS BATEAU? [HAUT AMONT]
VRAI
?ECRIS FERME? [BAS BASSIN]
FAUX
```

Les prédicats FERMEE? et OUVERTE? exigent un argument choisi parmi

[(PORTE / VANNE) (AMONT / AVAL)]

A l'aide de ces prédicats, nous pouvons "interroger" le micro-monde de l'écluse. Il reste à définir un prédicat BATEAU ? afin de déterminer/confirmer la présence du bateau en tel ou tel lieu.

A l'aide de ces prédicats, on va définir des macro-procédures gérant elles-mêmes les conditions d'utilisation de l'un des objets techniques constituant l'écluse. Prenons l'exemple analysé ci-dessus.
La vanne amont est ouverte, et nous pourrons admettre que c'est une condition nécessaire pour pouvoir ouvrir la porte amont. D'où, par exemple, la procédure :

POUR OUVRIR : LI
où LI est [(PORTE / VANNE) (AMONT / AVAL)]

```
POUR OUVRIR :LI
SI EGAL? PREM :LI "VANNE [VANNE LISTE DER :LI"OUVERTE]
SI FERMEE? LISTE "VANNE DER :LI [EC "IMPOSSIBLE STOP]
PORTE LISTE DER :LI "OUVERTE
FIN
```

De même, pour éviter les effets "pervers" tels que EAUMONTE (dans le bassin) avec FERMEE? [VANNE AMONT] valant VRAI, l'enfant peut introduire ce prédicat dans EAUMONTE.

Rappelons que le but est ici de créer un micro-monde et que l'enfant doit valider son algorithme graphiquement. c'est : donc à lui seul de "protéger" s'il le désire certaines macro-procédures.

PROCEDURES LOGO - ASCEN 2

FICHE PEDAGOGIQUE

ASCEN2 a été réalisé de la même manière que le logiciel ECLUSE : il s'agit donc de fournir des macro-primitives "ad hoc", nécessaires à la réalisation par les élèves d'un logiciel de simulation du fonctionnement d'un ascenseur. ASCEN2 reprend les mêmes macro-primitives, écrites cette fois à partir des procédures du générateur d'objets graphiques GENE. Cette différence de conception mise à part, les deux logiciels sont quasiment identiques.

ASCEN2 permet à l'utilisateur d'entrer facilement, si nécessaire, dans l'écriture des macro-primitives de niveau III ; il constitue un exemple d'application de GENE, ouvrant ainsi la voie à l'écriture de séries de logiciels de simulation d'objets sur écran graphique.

FICHE OPERATOIRE

- PALIER (dessin d'un palier).
  PALIER nombre

Nombre désigne le numéro de l'étage du palier que l'on veut dessiner (étages prévus de 1 à 4).

- TOUSPALIERS (dessine tous les paliers)

dessine tous les paliers de n1 à n2 (espacement: 20 lignes graphiques).

- PORTE (dessine une porte).
  PORTE n

n désigne le numéro de l'étage

- INIPORTES (dessine toutes les portes).
  INIPORTES
- MOBILE (dessine le sujet attendant l'ascenseur).
  MOBILE n

n désigne le numéro du palier sur lequel stationne le mobile.

- CABINE (dessine la cabine de l'ascenseur).
  CABINE n

n désigne le numéro de l'étage.

- CONTREPOIDS (dessine le contrepoids).
  CONTREPOIDS n

n désigne le numéro de l'étage où se trouve la cabine.

- AVMOB REMOB MONTEMOB DESMOB (permet de faire avancer, reculer, monter, descendre d'un pas le mobile). Le pas correspond à une ligne ou une colonne graphique.
  AVMOB
  REMOB
  MONTEMOB
  DESMOB
- MONTECAB DESCAB (permet de faire monter ou descendre la cabine d'un pas).
  MONTECAB
  DESCAB
- MONTECONTREPOIDS DESCONTREPOIDS (permet de faire monter ou descendre le contrepoids d'un pas).
  MONTECONTREPOIDS
  DESCONTREPOIDS
- CABPL? (test rendant VRAI si le mobile est dans la cage de l'ascenseur).
  CABPL?

- ENTREMOBILE (fait avancer le mobile dans l'ascenseur).
  ENTREMOBILE
- SORTMOBILE (fait reculer le mobile).
  SORTMOBILE n

n désigne le nombre de pas dont on veut faire reculer le mobile.

- MONTETOUT DESTOUT (fait monter ou descendre d'un pas la cabine ainsi que le mobile, s'il est dans la cage de l'ascenseur, et fait subir le mouvement inverse au contrepoids).
  MONTEBOUT
  DESTOUT
- MONTETAGE DESETAGE (fait monter ou descendre de quarante pas, soit un étage, la cabine ainsi que le mobile s'il est dans la cage de l'ascenseur, et fait subir le mouvement inverse au contrepoids).
  MONTETAGE
  DESETAGE
- INIT (affiche une image initiale possible)
  INIT n

n désigne le numéro de l'étage où attend le mobile. La cabine est au niveau 1, le contrepoids au niveau 4.

N.B. : la variable de nom "CAB contient en permanence le numéro de l'étage où se trouve la cabine.

PROCEDURES LOGO-PLISTE

FICHE OPERATOIRE

Philippe IV le Bel (1285-1314)

Louis X le Hutin (1314-1316)	**Philippe V le Long (1316-1322)**	**Charles IV le Bel (1322-1328)**
	aime	Lucile Charles
Antoine	habite	Lyon
	joue	Football Piano
identificateur	propriété	valeur

Ce morceau de l'arbre généalogique des Capétiens ou encore ces renseignements relatifs à la vie intime d'Antoine pourraient être gérés en LOGO par la structure d'affectation

DONNE <identificateur> <valeur>. Une analyse plus fine de la situation nous montre que cette affectation est à "deux niveaux" :
<propriété> <valeur>

Elle ne peut être directement et facilement gérée par la primitive DONNE.

Le concept d'affectation se généralise donc en LOGO. Que nous prenions l'exemple des Capétiens ou l'arbre des goûts d'Antoine.

Si nous associons la liste [LOUIS X. PHILIPPE V. CHARLES IV.] à PHILIPPE IV c'est pour traduire la relation A.POUR.FILS.

De même la liste [LUCILE CHARLES] est associée à ANTOINE pour rendre compte du lien verbal AIME.

Pour rendre compte d'une telle situation, certains LOGO ont à leur disposition des primitives de P-LISTE.

Traduisons ces possibilités sur notre exemple :

- DPROP <nom> <propriété> <objet>
  MOT MOT MOT ou LISTE

Cette primitive va créer une liste (ici ANTOINE) faite des couples <propriété> <valeur>.

```
DPROP "ANTOINE "AIME [LUCILE CHARLE]
DPROP "ANTOINE "HABITE "LYON
DPROP "ANTOINE "JOUE [FOOTBALL PIANO]
```

• PLISTE <nom>
La primitive PLISTE retourne la liste des propriétés associées au nom générique.
ECRIS PLISTE "ANTOINE
[AIME [LUCILE CHARLES]] [HABITE LYON][[JOUE FOOTBALL PIANO]]
• RPROP <nom> <propriété>

Toutefois, il est possible de n'extraire que l'objet associé à une seule propriété.
ECRIS RPROP "ANTOINE "AIME LUCILE CHARLES
ECRIS RPROP "ANTOINE "HABITE LYON

• APROP <nom> <propriété>.
Si un indicateur de propriété de l'objet (mot ou liste) associé est inutile, APROP assure la fonction de supprimer le couple <propriété> <objet> correspondant.

Créer des P-LISTES
Une P-LISTE est donc une liste de listes. Ainsi, le contenu de ANTOINE est la liste :
[AIME [LUCILE CHARLES]] [HABITE LYON]...
P101P202

Pour créer une telle liste, il faut intégrer DEUX contraintes :
a. Créer la liste proprement dite (si ce n'est déjà fait) ;
b. Y adjoindre un couple <indic i - objet i> dans le cas contraire. Là encore, on pourrait distinguer le cas où indic i existe déjà ou non. S'il existe, on peut envisager de réaliser soit une substitution d'objet, soit une adjonction. On choisit généralement la substitution.

Les procédures DROP, APROP et RPROP sont bâties ici autour d'un prédicat noté PRES? : MO : INDIC : CO

PRES? retourne FAUX si l'indicateur de propriété :INDIC n'est pas présent dans la PLISTE :MO ou encore si la PLISTE :MO n'est pas créée.

PRES? retourne VRAI dans le cas contraire. De plus, cette procédure entraîne une permutation circulaire des couples <INDIC, OBJET> afin de placer le couple concerné en tête de PLISTE.

:CO désigne le nombre de sous-listes de la PLISTE. Ce compteur est à pour assurer le traitment récursif de PRES?

Gérer ces P-LISTES
RPROP est donc une liste et de ce fait, nous pouvons envisager de comparer des listes au "sens ensembliste" du terme.

Ainsi, nous observons que Lucile et Roger aiment tous deux le ski...

Pour traiter mécaniquement ce genre de problème, il reste à réaliser quelques procédures ensemblistes telles que :

POUR INTER :A :B

qui retourne la liste des objets communs aux listes A et B ;

POUR REUNION : A :B

qui retourne la liste des objets présents dans l'une ou l'autre des deux listes ;

```
  POUR INCLUS? :A :B
```
prédicat qui retourne VRAI si les objets présents dans A sont également présents dans B (et FAUX sinon).

Procédures "ensemblistes"

Attention... de bien observer qu'une liste n'est pas un ensemble au sens mathématique du terme.

```
  POUR REUNION :A :A
>SI VIDE? :A [RENDS :B]
>SI MEMBRE ? PREM :A :B [RENDS REUNION SP :A :B]
>RENDS MP PREM :A REUNION SP :A :B
>FIN
  POUR INTER :A :B
>SI VIDE? :A [RENDS :A]
>SI MEMBRE? PREM :A :B [RENDS MP PREM :A INTER SP :A :B]
>RENDS INTER SP :A :B
>FIN
  POUR INCLUS? :A :B
>RENDS EGAL? :A INTER :A :B
>FIN
```

PROCEDURES LOGO - TEXTE DEFINIS

FICHE OPERATOIRE

Deux commandes sont utilisables : DEFINIS et TEXTE. Pour bien comprendre les écritures de ces deux procédures, rappelons tout d'abord - sur un exemple - le code des périphériques et le rôle des commandes ENTREE et SORTIE.

Code des périphériques
0 Nulle 1 Console 2 Imprimante 3 Série 4 Editeur

```
POUR ESSAI
SORTIE 4                FIN
EC [DONNE "A 5]         ?
SORTIE 1                ?ESSAI
ENTREE 4                ?EC :B
DONNE "B LL             DONNE "A 5
ENTREE 1                ?EC :A
EXEC :B                 5
EC :A                   ?
```

Commentons les effets de cette procédure :
SORTIE 4 : Sortie EDITEUR ; la liste [DONNE "A 5] est écrite dans l'éditeur.
SORTIE 1 : Sortie ECRAN
ENTREE 4 : Dès cette commande, les entrées se font à partir de l'éditeur.
Donne "B LL ➔ LL opère sur la première liste lue dans la page éditeur. [DONNE "A 5] est donc placée dans B.
ENTREE 1 : Les entrées se font à partir de la console (clavier).

Les commandes DEFINIS

- DEFINIS nécessite deux entrées
  DEFINIS <nom> <liste>
  <nom> est le NOM de la procédure qui va ainsi être définie ;
  <liste> est une liste de listes :

la première sous-liste est la liste des arguments ; chaque sous-liste suivante représente une ligne de procédure.

- Listage des procédures réalisant DEFINIS :

```
POUR DEFINIS : TITR :LIST
SORTIE 4
TAPE "POUR TAPE CAR 32
TAPE :TITR TAPE CAR 32
EC PREM :LIST
DEFPROC SP :LIST
EC "FIN
ENTREE 4
```

```
  SORTIE 1
  FIN
  ?
  POUR DEFPROC :L
  SI VIDE ? :L [STOP]
  EC PREM :L
  DEFPROC SP :L
  FIN
```

Exemple :

```
? DEFINIS "SPI [[:CO :AN AU][AV :CO TD :AN][SPI :CO RESTE :AN
+ :AU  360 :AU]]
?
?IM "SPI
POUR SPI :CO :AN :AU
AV :CO TD :AN
SPI :CO RESTE :AN + :AU 360 :AU
FIN
?
?SPI 15 2 20
```

La commande TEXTE

Il est parfois intéressant d'utiliser la procédure inverse, c'est-à-dire transformer une procédure existante en une liste structurée comme précédemment.

- Listage des procédures réalisant TEXTE:

```
    POUR TEXTE :NOM
    SORTIE 4
    IM :NOM
    SORTIE 1
    ENTREE 4
    DONNE :NOM MP SP SP LL
    SUITE LL[]
    SORTIE 1
    FIN
    ?
    POUR SUITE :S
    SI EGAL? :S[FIN][STOP]
    DONNE :NOM MD :S CHOSE :NOM
    SUITE LL
    FIN
```

Exemple :

Réalisons d'abord une procédure pour ensuite la transformer par TEXTE...

```
?POUR CARRE :COTE
>VE
>REPETE 4[AV :COTE TG 90]
>FIN
VOUS VENEZ DE DEFINIR CARRE
?
?TEXTE "CARRE
?EC :CARRE
[:COTE][VE][REPETE 4[AV :COTE TG 90 ]]
```