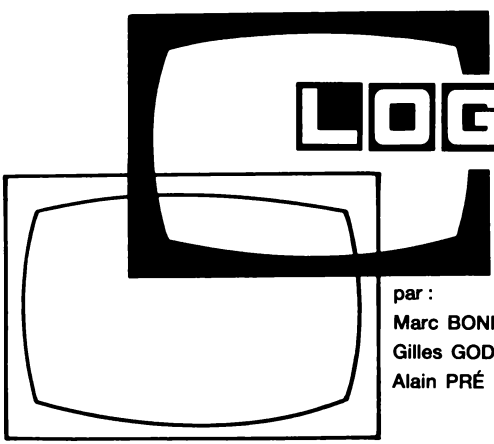


LOGOMONDE

sur ordinateurs THOMSON *

par :
Marc BONNETON
Gilles GODIMIER
Alain PRÉ

* THOMSON est une marque déposée Les ordinateurs THOMSON sont T07 T07 70 M05

SOMMAIRE

TORTUE DESSINE MOI UN...

Spécialiste du graphisme, la tortue LOGO est capable de dessiner toutes les figures géométriques imaginables. Il est particulièrement facile de lui demander des polygones réguliers.
Les procédures CARRÉ et MEILLEUR.CARRÉ dessinent toutes deux un carré de 50 pas ; la deuxième est préférable.
Remarque :

- dans chacune de ces procédures, la tortue termine son dessin en revenant au point de départ, tournée dans la même direction. Elle a donc effectué une rotation de 360° (4 fois 90°). Le théorème de TCT (Tour Complet Tortue) est vérifié.

à vous de jouer...

Dessinez un triangle équilatéral, un hexagone... N'oubliez pas le TCT !
Essayez de dessiner un cercle. C'est un polygone avec beaucoup de côtés qui vérifie lui aussi le TCT.

programmer : TORTUE DESSINE MOI UN [redacted]

```
POUR CARRE
AVANCE 50 TD 90
AVANCE 50 TD 90
AVANCE 50 TD 90
AVANCE 50 TD 90
FIN
```

• (TCT : 90+90+90+90=360)

```
POUR MEILLEUR.CARRE
REPETE 4 [AVANCE 50 TD 90]
FIN
```

• (TCT : 4 × 90 = 360)

```
POUR RECTANGLE
REPETE 2 [AVANCE 80 TD 90 AVANCE 40 TD 90]
FIN
```

• (TCT : 2 × (90+90)=360)

```
POUR TRIANGLE
REPETE 3 [AVANCE 50 TD 120]
FIN
```

• (TCT : 3 × 120 = 360)

```
POUR PENTAGONE
REPETE 5 [AVANCE 50 TD 72]
FIN

POUR HEXAGONE
REPETE 6 [AVANCE 50 TD 60]
FIN

POUR CERCLE
REPETE 360 [AVANCE 1 TD 1]
FIN

POUR PETIT.CERCLE
REPETE 180 [AVANCE 1 TD 2]
FIN
```

LA TORTUE AU PIED DU MUR.

Comment faire réaliser un mur à notre tortue en lui apprenant les seuls savoir-faire élémentaires qui la rendront capable de monter ensuite n'importe quel type de construction.

comment faire ?

Un mur est composé de moëllons, aussi, apprenons à la tortue à faire un moellon.

La procédure MOËLLON :
- dessine un moëllon de 30 par 10 par exemple,
- replace la tortue à droite de ce moëllon, prête à poser le suivant.

La procédure DEMI-MOËLLON :
- dessine un demi-moëllon, ce qui sera bien utile pour commencer ou terminer certains rangs (1 sur 2),
- replace aussi la tortue à droite de ce demi-moëllon.

La procédure BORD.GAUCHE permettra à la tortue de se replacer à la gauche du mur, quand elle aura terminé un rang, car notre tortue pose ses moëllons sur la droite.

La procédure DEUX.RANGS :
- dessine une rangée de six moëllons,
- positionne la tortue à gauche du mur,
- dessine une rangée de 5 moëllons en commençant et finissant par un demi-moëllon.

Enfin, la procédure MUR :
- place la tortue à la position —100 —4 par exemple,
- fait répéter quatre fois l'exécution de deux rangs,
- fait cacher la tortue.

programmer : LA TORTUE AU PIED DU MUR

```
POUR MUR
LC FPOS [— 100 — 40] BC
REPETE 4 [DEUX.RANGS]
CT
FIN

POUR DEUX.RANGS
REPETE 6 [MOELLON]
BORD.GAUCHE
DEMI.MOELLON
REPETE 5 [MOELLON]
DEMI.MOELLON
BORD.GAUCHE
FIN

POUR BORD.GAUCHE
LC TG 90 AV 180
TD 90 AV 10 BC
FIN

POUR MOELLON
REPETE 2 [AV 10 TD 90 AV 30 TD 90]
TD 90 AV 30 TG 90
FIN

POUR DEMI.MOELLON
REPETE 2 [AV 10 TD 90 AV 15 TD 90]
TD 90 AV 15 TG 90
FIN
```

- FPOS fixe la position de la tortue au point de coordonnées — 100, — 40. LC (lève-crayon) permet à la tortue de se rendre à ce point sans laisser de trace
  Ne pas oublier le BC (baissecrayon) qui suit

- Un moëllon est un rectangle

POLYGONES

Utilisation de paramètres

La procédure OCTO dessine un octogone dont chaque côté mesure 30 pas tortue. Pour obtenir une figure plus grande ou plus petite, nous sommes contraints à redéfinir une autre procédure.
Par contre, en utilisant une variable que nous appellerons « CÔTÉ », construisons une procédure plus générale (OCTOGONE) qui nous permette de dessiner un octogone et d'en choisir la taille.
En tapant OCTOGONE 10, OCTOGONE 23, OCTOGONE 100, vous dessinez autant de figures de taille différente.
Créez la procédure qui dessine tout polygone régulier : il suffit de lui donner le nombre et la longueur des côtés.

comment faire ?

La tortue a besoin de trois variables :

- :Nombre (nombre de côtés)
- :Longueur (longueur d'un côté)
- :Angle (angle de rotation)

Mais grâce au TCT, l'angle de rotation de la tortue est calculé à partir des deux premiers paramètres :

- :Longueur = 360 / :Nombre

La procédure POLYGONE n'a donc que deux paramètres. Elle vous permet de tracer toutes les figures régulières.
Exemples :

- POLYGONE 80 3 dessine un triangle.
- POLYGONE 50 6 dessine un hexagone.
- POLYGONE 2 180 dessine un cercle.

programmer : POLYGONES

```
POUR OCTO
REPETE 8 [AV 30 TD 45]
FIN

POUR OCTOGONE :COTE
REPETE 8 [AV :COTE TD 45]
FIN

POUR POLYGONES :LONGUEUR :NOMBRE
REPETE :NOMBRE [AVANCE :LONGUEUR TD 360 / :NOMBRE]
FIN
```

ALEA JACTA EST

(jeu de dés).

Jouer aux dés avec un ordinateur ? ? ?
Rassurez-vous, il n'est pas question de le faire rouler sur la table comme un dé à jouer, mais de demander à LOGO de simuler puis de nous indiquer le résultat de son jet (sans tricher).

comment faire ?

Il faut demander à LOGO d'afficher un nombre choisi au hasard entre 1 et 6 inclus.
La primitive HASARD nous donne un nombre imprévisible entre 0 inclus et un maximum qui, lui, est exclus.
Exemple :

- EC HASARD 6 peut afficher 0, 1, 2, 3, 4 ou 5 mais pas 6.

Pour avoir le 6, on pourrait faire HASARD 7 mais dans ce cas, on ne règle pas le problème du 0 qui peut sortir et qui n'existe pourtant pas sur un dé.
Par contre, si on ajoute 1 au résultat de HASARD 6 (0 1 2 3 4 ou 5), on obtient toutes les sorties possibles d'un dé (1 2 3 4 5 ou 6).
La procédure DE simule donc bien le jet d'un dé.
Quant à DES, elle simule un jet de deux dés, c'est-à-dire qu'elle affiche un nombre aléatoire compris entre 2 et 12.
Cependant nous n'avons pas utilisé 2 + HASARD 11 car cette formule ne traduit pas fidèlement un jet de deux dés : en effet, un 7 peut être obtenu de six façons différentes (1 + 6, 2 + 5, 3 + 4, 4 + 3, 5 + 2, 6 + 1) alors qu'un 12 ne peut l'être que d'une seule (6 + 6). La formule 2 + HASARD 11 ne reflète pas cette différence de probabilité.

à vous de jouer...

Ecrivez un programme qui tire six nombres au hasard compris entre 1 et 49 (programme facile à réaliser mais qui peut rapporter gros !).

programmer : ALEA JACTA EST (jeu de dés)

```
POUR DE
ECRIS 1 + HASARD 6
FIN
```

Attention à la priorité des opérations

- 1 + HASARD 6 peut retourner 1,2,3,4,5, ou 6 alors que
- HASARD 1 + 6 peut retourner 0,1,2,3,4,5 ou 6. la primitive préfixée SOMME évite ce genre de problèmes

```
POUR DE
ECRIS SOMME 1 HASARD 6
FIN
```

- SOMME a b est équivalent à a + b

```
POUR DES
ECRIS SOMME 1 + HASARD 6 1 + HASARD 6
FIN
```

- On ajoute les résultats de 2 dés lancés séparément
- (SOMME a b c d) permet d'ajouter plus de deux nombres. Cette notation est équivalente à :
  SOMME a SOMME b SOMME c d

RETOUR A L'ENVOYEUR

Le point sur la primitive rends

Ecrire une procédure pour calculer le carré d'un nombre. Trop facile ! répondrez-vous. Mais pensez bien que nous ne savons pas encore comment sera utilisé le résultat. Il pourra être :

- Simplement affiché.
- Mis dans une variable.
- Utilisé dans un calcul.
- Pris comme paramètre d'une autre procédure.

Il faut donc que ce que nous allons écrire soit général et n'interdise aucune de ces utilisations.

comment faire ?

Vous l'avez deviné, il faut utiliser la primitive RENDS :

```
POUR CARRE :N
RENDS :N × :N
FIN
```

Le résultat (:N × :N) ne sera ni affiché, ni stocké... Il sera simplement « renvoyé ». Si une autre procédure est là pour le récupérer, tout va bien. Par exemple :

```
ECRIS CARRE 3
9
```

Mais, s'il n'y a personne, LOGO ne sait pas quoi faire de ce résultat. Il nous le dit d'ailleurs :

```
CARRE 3
QUE FAIRE DE 9
```

L'avantage de RENDS, c'est que notre fonction (c'est le nom d'une procédure qui comporte un RENDS) peut être utilisée très librement :

```
DONNE "RESULTAT CARRE 4          (stocke 16 dans "RÉSULTAT)

ECRIS 3 + CARRE 2                (affiche 3 + 4)
7

ECRIS CARRE CARRE 2              (affiche le carré... du carré de 2)
16
```

Pensez à utiliser les fonctions. LOGO est fait pour ça.
exemples :

```
POUR PI                          POUR DEUXIEME :LST
RENDS 3.1416                     RENDS PREM SP :LST
FIN                              FIN
```

Remarque : RENDS met fin à la procédure tout comme le ferait STOP.

à vous de jouer...

Les fonctions peuvent travailler sur des listes, des mots, des chiffres ou des grandeurs booléennes. Ecrire les fonctions suivantes :

CUBE :N	retourne le cube de :N
DIFF :N1 :N2	retourne la valeur :N1 - :N2 (certains LOGO possèdent déjà cette fonction)
SUPEG :N1 :N2	retourne VRAI si :N1 ⩾ :N2
INFEG :N1 :N2	retourne VRAI si :N1 ⩽ :N2
OUEX :P1 :P2	retourne VRAI si seulement l'une des deux propositions (:P1 ou :P2) est vraie
PAIR :N	retourne VRAI si :N est pair
IMPAIR :N	sans commentaires
MUTL :N1 :N2	retourne VRAI si :N2 est un multiple entier de :N1

etc, etc...

LE DEVIN

Sans vouloir prédire la prochaine panne de l'ordinateur, il faudra simplement deviner le nombre, compris entre 0 et 1 000, auquel LOGO aura « pensé ». A chaque essai, il nous répondra « Trop grand » ou « Trop petit » ou « Quelle perspicacité » selon le cas.

comment faire ?

Choisir un nombre au hasard entre 0 et 1000 ne devrait plus vous poser de problèmes. Ceci sera la première action du programme. Il faudra ensuite saisir et analyser la réponse du joueur, afficher le message adapté jusqu'à ce que le nombre soit découvert.
Nous allons appeler le nombre secret "N SECRET, et le nombre proposé par le joueur "RÉPONSE.

L'instruction « SI » permet de choisir l'action adaptée. On l'utilise de la façon suivante : SI condition, alors [ACTION 1] sinon [ACTION 2].

La procédure d'entrée DEVIN :
- Choisit un nombre au hasard entre 0 et 999 qu'elle range dans la variable "N.SECRET.
- Explique sommairement les règles.
- Lance le jeu en appelant la procédure CHOIX.JOUEUR.

Cette dernière compare la réponse du joueur au nombre secret et affiche le message correspondant (« trop grand » ou « trop petit »). Elle répète cette action (avec chaque fois une nouvelle réponse) jusqu'à ce qu'il y ait égalité entre :RÉPONSE et :N.SECRET. Dans ce cas, elle affiche « quelle perspicacité » et s'arrête.

à vous de jouer...

On peut transformer le programme pour qu'il compte les essais et les affiche au moment de la victoire.
Pour compliquer le jeu, on peut aussi limiter le nombre d'essais à 11 tentatives. Arrivé à ce stade, LOGO donnera la réponse.

programmer : LE DEVIN

```
POUR DEVIN
DONNE "N. SECRET HASARD 1000
EXPLICATIONS
CHOIX. JOUEUR PREM LL
FIN

POUR CHOIX.JOUEUR :RÉPONSE
SI :RÉPONSE = :N.SECRET (EC (QUELLE PERSPICACITÉ) STOP)
SI :RÉPONSE < :N.SECRET (EC (TROP PETIT)) (EC (TROP GRAND))
CHOIXJOUEUR PREM LL
FIN

POUR EXPLICATIONS
VT EC (ESSAYEZ DE DEVINER A QUEL NOMBRE JE PENSE)
EC"EC (IL EST COMPRIS ENTRE 0 ET 999 INCLUS)
EC"EC (PROPOSEZ VOS ESSAIS)
FIN
```

- LL retourne une liste, par exemple (14) PREM LL en est le premier élément : 14 Dans ce programme, nous voulons travailler sur le nombre 14 et non pas sur la liste (14)

LOGO MAGE TRANSCENDENTAL

En pratiquant le DEVIN, arrivez-vous à chaque fois à deviner le nombre en 11 tentatives au plus ? C'est à souhaiter car maintenant, nous allons apprendre à LOGO à réaliser cette performance : c'est vous qui allez penser à un nombre plus petit que 1000 et ce sera à l'ordinateur de le découvrir. Aussi, n'hésitez pas à recharger DEVIN en mémoire, le temps de vous entraîner.

comment faire ?

Quand LOGO nous proposera sa solution, nous appuierons sur les touches « < » si l'ordinateur a joué trop bas, « > » dans le cas contraire ou « = ». La stratégie gagnante qu'il faudra apprendre à l'ordinateur consiste à proposer le nombre situé juste au milieu de l'intervalle. Celui-ci se réduira à chaque réponse du joueur.
Exemple :

- Vous choisissez 432.
- Premier coup de l'ordinateur : 500 (milieu de 0 et 1000)
- Votre réponse : >
- Troisième coup de l'ordinateur : 375 (milieu de 250 et 500) etc...

C'est la méthode dichotomique.
Nous allons utiliser "MIN pour stocker la borne inférieure de l'intervalle et "MAX pour la borne supérieure.

La procédure TROUVER :MIN :MAX doit :

- proposer le milieu de l'intervalle :MIN :MAX.
- Lire la réponse du joueur (les touches « < », « > » ou « = »).
- En cas d'égalité, crier victoire.
- Sinon :

Soit la réponse est « < », le nouvel intervalle est MILIEU :MAX
Soit la réponse est « > », le nouvel intervalle est :MIN MILIEU on refait la même chose avec ce nouvel intervalle.

à vous de jouer...

Avec ce programme, deux cas peuvent se présenter :
1 LOGO trouve la solution en 11 coups au plus (cas normal).
2 Au bout du douzième coup, LOGO n'a toujours pas trouvé. Plusieurs raisons :

- Votre programme n'est pas faux mais n'est pas encore juste ! (voyez notre version).
- Vous avez choisi un nombre en dehors des bornes.
- Vous vous êtes trompé (involontairement ?) dans vos réponses.

Modifiez le programme pour que LOGO détecte ce genre de problèmes (dépassement du onzième coup).
Si la borne supérieure est 2000, combien de coups minimum faudra-t-il à LOGO pour trouver la réponse à coup sûr ?

programmer : LOGO MAGE TRANSCENDENTAL

```
POUR MAGE
TROUVER 0 1000
FIN

POUR TROUVER :MIN :MAX
EC PH (JE JOUE) MILIEU
DONNE "RÉPONSE LISCAR
SI :RÉPONSE = "= (EC (J'AI GAGNÉ) STOP)
SI :RÉPONSE = "> (TROUVER :MIN MILIEU) (TROUVER MILIEU
:MAX)

POUR MILIEU
RENDS ENT ( :MIN + :MAX ) / 2
FIN
```

SOUSTRAIRE POUR DIVISER

Le point sur la récursivité.

Effectuer une division à l'aide d'une calculette nous donne le quotient mais pas le reste.
Comment combler cette lacune en utilisant la méthode des soustractions successives ?

comment faire ?

On expose la méthode à l'aide d'un algorithme qui se présente sous cette forme :

```
DIVISER (dividende, diviseur)
• début
        reste = dividende
        quotient = 0
        Tant que reste > diviseur faire
                reste = reste — diviseur
                quotient = quotient + 1
        finfaire
        écrire quotient, reste
• fin.
```

Si votre LOGO ne possède pas la primitive QUOTIENT, vous pourrez vous créer une fonction LOGO qui, en en utilisant ce principe, retoume le résultat d'une division. Ainsi, vous pourrez utiliser ce résultat à des fins diverses (affichage, stockage dans une variable, opérations arithmétiques) comme on peut le faire avec les primitives SOMME et PRODUIT.

Les fonctions DIVISION et RESTE utilisent l'algorithme ci-dessus mais sous une forme récursive. Essayons d'examiner ce qui se passe sur un exemple : EC DIVISION 5 2.

à vous de jouer...

Modifiez le programme pour qu'il permette la division de nombres négatifs. Pensez à créer ou utiliser la fonction VALEUR ABSOLUE.

programmer : SOUSTRAIRE POUR DIVISER

```
POUR DIVISION :DIVIDENDE :DIVISEUR
SI :DIVIDENDE < :DIVISEUR (RENDS 0)
RENDS 1 + DIVISION :DIVIDENDE — :DIVISEUR :DIVISEUR
FIN

POUR SOLDE :DIVIDENDE :DIVISEUR
SI :DIVIDENDE < :DIVISEUR (RENDS :DIVIDENDE)
RENDS SOLDE :DIVIDENDE — :DIVISEUR :DIVISEUR
FIN

EC DIVISION 5 2
2

          DIVISION 5 2
          SI 5 < 2 (     )
          RENDS 1 + DIVISION 3 2

                    DIVISION 3 2
                    SI 3 < 2 (     )
                    RENDS 1 + DIVISION 1 2

                              DIVISION 1 2
                              SI 1 < 2 (RENDS 0)
```

8

DES ARBRES ! SI J'EN CONNAIS...

Une application simple des listes.

On se propose de découvrir différents noms d'arbres que connaît LOGO. A chaque proposition du joueur, LOGO répond s'il connaît ou non l'arbre cité. Le joueur a la possibilité de quitter le jeu quand il le désire.

comment faire ?

On stocke dans la variable "NOMS la liste des arbres connus de LOGO.
On attend la proposition du joueur.
Si cette proposition fait partie de la liste des mots connus, on affiche « je connais » sinon « je ne connais pas ».
Enfin, on demande au joueur s'il désire continuer à jouer.
C'est la procédure ARBRES qui exécute ces différentes actions.

à vous de jouer...

Reprendre le même jeu en ajoutant la possibilité de compter le nombre de coups joués et le nombre de noms découverts.

Réalisez le programme qui connaît plusieurs types d'objets (ARBRES, FLEURS, MEUBLES, VÉHICULES, etc.) Après la proposition du joueur, LOGO nous dit s'il connaît ce nom et à quelle catégorie il appartient.

programmer : DES ARBRES ! ! SI J'EN CONNAIS

```
POUR ARBRES
VT
DONNE "NOMS [PIN SAPIN PLATANE CERISIER POMMIER POIRIER
BOULEAU HÊTRE]
EC [DONNEZ MOI UN NOM D'ARBRE ET JE VOUS DIRAI]
TAPE PH [SI JE LE CONNAIS]"
SI MEMBRE? PREM LL:NOMS [EC [JE CONNAIS]] [EC [JE NE
CONNAIS PAS]]
TAPE [VOULEZ VOUS REJOUER?]
SI LISCAR = "O [ARBRES]
FIN
```

APPRENTISSAGE... (mes lacunes m'enrichissent)

A partir du jeu précédent, modifions le programme pour qu'en plus, LOGO apprenne les mots qu'il découvre. A la fin du jeu, il affiche le nombre d'arbres appris et en donne la liste.

comment faire ?

Cette fois-ci, LOGO ne connaît aucun nom au départ du jeu. On initialise donc la variable ARBRES avec liste vide. Comme précédemment, LOGO demande à l'intervenant un nom d'arbres ; si celui-ci est connu, il le dit sinon, il l'ajoute à sa liste de mots connus.

La procédure d'entrée APPRENTISSAGE :
- Initialise ARBRES
- Appelle la procédure APPRENDS.
- Affiche le résultat (nombre et noms des arbres appris).

La procédure APPRENDS :
- Interroge l'intervenant et range sa proposition dans "NOUVEL.ARBRE.
- Teste si la proposition est connue.
- Si c'est le cas, elle affiche « je connais déjà ».
- Sinon elle appelle MEMORISER.
- Enfin, elle demande au joueur s'il veut rejouer.

La procédure MEMORISER :
- Elle place :NOUVEL.ARBRE en fin de la liste :ARBRES.

programmer : APPRENTISSAGE (OU MES LACUNES M'ENRICHISSENT)

```
POUR APPRENTISSAGE
DONNE "ARBRES ( )
VT EC (DONNEZ MOI DES NOMS D'ARBRES) EC "
APPRENDS PREM LL
VT EC PH PH (J'AI APPRIS) COMPTE :ARBRES (NOUVEAUX ARBRES)
EC " EC PH (EN VOICI LA LISTE) :ARBRES
FIN

POUR APPRENDS :NOUVEL. ARBRE
SI MEMBRE? :NOUVEL. ARBRE :ARBRES (EC (JE CONNAIS DEJA))
(MEMORISER)
EC (VOULEZ VOUS M'EN DONNER UN AUTRE (O/N)?)
SI LISCAR = "O ((JE VOUS ÉCOUTE) APPRENDS PREM LL)
FIN

POUR MÉMORISER
DONNE "ARBRES MD :NOUVEL. ARBRE :ARBRES
FIN
```

NE PERDEZ PAS LA BOULE

Une salle de jeu à domicile avec, dans le rôle du croupier, LOGO. En effet, nous vous proposons de créer le jeu dit de « la boule » qui s'apparente à la roulette. Le programme demande votre mise, lance la boule, annonce les résultats :
Numéro sorti : 1 à 9
Pair ou impair
Rouge (2 4 7 9) ou noir (1 3 6 8)
Passe (plus grand que 5) ou manque (plus petit que 5)
Si vous avez gagné, il ajoute le gain à votre capital et, dans le cas contraire, il ramàsse votre mise.
Le programme s'arrête lorsque vous n'avez plus d'argent.

comment faire ?

La procédure JOUER :
- Efface l'écran.
- Définit ce qu'est un « simple » dans le langage du jeu.
- Demande le capital de départ.
- Commence le jeu avec ce capital.

La procédure BOULE :
- Affiche votre capital.
- Vérifie qu'il soit suffisant pour continuer le jeu.
- Demande votre pari et votre mise.
- Lance la boule (BOULER).
- Recommence avec votre capital augmenté ou diminué suivant votre chance.

La procédure BOULER détermine tous les paramètres du jet et les stockes dans la variable "RÉSULTAT.

NOUVEAU.CAPITAL calcule votre nouveau capital :
- Si votre pari n'est pas dans :RÉSULTAT, il retire la mise.
- Dans le cas contraire, il ajoute le gain : 7 fois la mise si vous aviez parié un numéro sinon, une fois la mise pour un « simple ».

Nous avons aussi utilisé deux procédures un peu « passe-partout » que vous retrouverez souvent dans d'autres programmes.
PAIR qui teste si un nombre est pair et retourne VRAI dans ce cas.
AJOUTE qui ajoute un nouvel élément à la liste :RÉSULTAT.

à vous de jouer...

- Réalisez le programme qui vous permet de jouer à plusieurs.
- Réalisez le jeu de la roulette en exploitant ses multiples possibilités de mises.

programmer : NE PERDEZ PAS LA BOULE

```
POUR JOUER
VT
TAPE PH [INDIQUEZ VOTRE CAPITAL DE DÉPART]"
BOULE PREM LL
FIN

POUR BOULE :CAPITAL
EC " EC PH [VOTRE CAPITAL : ] :CAPITAL          • EC " saute une ligne
SI :CAPITAL < 1 [STOP]
TAPE PH [VOTRE MISE ?] " DONNE "MISE PREM LL
TAPE PH [VOTRE PARI ?] " DONNE "PARI PREM LL
BOULER
BOULE NOUVEAU.CAPITAL
FIN

POUR BOULER
DONNE "RÉSULTAT [ ]
DONNE "NUMÉRO 1 + HASARD 9
AJOUTE :NUMÉRO
SI PAIR :NUMÉRO [AJOUTE "PAIR] [AJOUTE "IMPAIR]
SI MEMBRE? :NUMERO [1 3 6 8] [ AJOUTE "NOIR] [AJOUTE "ROUGE]
SI :NUMÉRO < 5 [AJOUTE "MANQUE] [AJOUTE "PASSE]
FIN

POUR NOUVEAU.CAPITAL
SI NON MEMBRE? :PARI :RÉSULTAT [RENDS :CAPITAL — :MISE]
SI NOMBRE? PARI [RENDS :CAPITAL + :MISE × 7] [RENDS :CAPITAL
+ :MISE]
FIN

POUR PAIR :N
RENDS 0 = RESTE :N 2
FIN

POUR AJOUTE :MOT
DONNE "RÉSULTAT MD :MOT :RÉSULTAT
FIN
```

PGCD, PPCM

Confions à LOGO le calcul du Plus Grand Commun Diviseur et du Plus Petit Commun Multiple de deux entiers.

comment faire ?

Pour le PGCD, on utilise le théorème suivant :
si un nombre D divise A et divise B, alors il divise aussi leur différence.
On ramène ainsi le calcul du PGCD de A et B au calcul du PGCD de leur différence et du plus petit des deux.
Exemple :

- PGCD 12 3

se ramène à

- PGCD 9 3

qui se ramène à

- PGCD 6 3

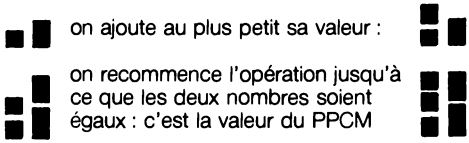
ou encore

- PGCD 3 3 qui est bien sûr 3.

La procédure PGCD :

- Elle renvoie le PGCD de la différence des deux nombres et du plus petit d'entre eux, ceci jusqu'à calculer le PGCD de deux valeurs identiques.

Pour le PPCM, voici une représentation graphique du calcul du PPCM de 2 et 3 :



La procédure PPCM appelle INCREMENT qui se charge d'ajouter au plus petit des deux nombres, sa valeur de départ jusqu'à les égaliser.

programmer : PGCD PPCM

```
POUR PGCD :N1 :N2
SI :N1 = :N2 [RENDS :N1]
SI :N1 > :N2 [RENDS PGCD :N1 — :N2 :N2] [RENDS PGCD :N2 —
:N1 :N1]
FIN

POUR PPCM :N1 :N2
RENDS INCREMENT :N1 :N2
FIN

POUR INCREMENT :A :B
SI :A = :B [RENDS :A]
SI :A < :B [RENDS INCREMENT :A + :N1 :B] [RENDS INCREMENT :A
:B + :N2]
FIN
```

VRAI OU FAUX

Le point sur les fonctions booléennes

Un prédicat est une question à laquelle LOGO répond par VRAI ou FAUX. Ils nous renseignent sur :

- L'état de la tortue,
- Le monde extérieur,
- Les éléments (objets LOGO) du langage,
- Le résultat d'une comparaison.

graphique :

VISIBLE? : nous dit si la tortue est visible (VRAI) ou non (FAUX)

```
POUR CHANGE ETAT
SI VISIBLE? [CT] [MT]
FIN
```

le monde extérieur :

TOUCHE? : nous dit si une touche a été pressée ou non.

```
POUR INTERRUPTION
SI TOUCHE? [RENDS LISCARA] [RENDS []]
FIN
```

INTERRUPTION permet de changer le cours d'un programme lorsque l'utilisateur appuie sur une touche.

BOUTON? rends VRAI si le bouton de la manette est pressé.
Exemple d'utilisation :

```
POUR MORSE
SI BOUTON? [JOUE "LA]
MORSE
FIN
```

CONTACT ? permet de savoir si le crayon optique est appuyé. Cette fonction est très utile car, si elle n'est pas VRAI, les informations renvoyées par POSOPT sont inutilisables.

```
POUR TELECRAN
SI CONTACT? [FPOS POSOPT]
TELECRAN
FIN
```

les objets logo :

```
LISTE? : OBJ
EC LISTE? 12
FAUX

EC LISTE? "MACHIN
FAUX
```

```
EC LISTE? (1 2 3 4 5 )
VRAI

EC LISTE? PREM (1 2 3 4)
FAUX

LISTE? PREM ((A B) C D )
QUE FAIRE DE VRAI

EC LISTE? SP (A B)
VRAI

MOT? : OBJ
EC MOT ? 12
VRAI

EC MOT? "MACHIN
VRAI

EC MOT? (1 2 3 4 5)
FAUX

EC MOT? PREM (1 2 3 4)
VRAI

EC MOT? PREM ((A B) C D)
FAUX

EC MOT? SP (A B)
FAUX

NOMBRE? : OBJ
EC NOMBRE? 12
VRAI

EC NOMBRE? "MACHIN
FAUX

EC NOMBRE? (1 2 3 4 5)
FAUX

EC NOMBRE? PREM (1 2 3 4)
VRAI

EC NOMBRE? PREM ((1 2) 3 4)
FAUX

EC NOMBRE? SP (1 2)
FAUX
```

NOM? :OBJ Retourne VRAI si le paramètre :OBJ est une variable donc s'il a un contenu.

```
DONNE "ARBRE "PIN
EC NOM? "ARBRE
VRAI

EC NOM? : ARBRE
FAUX
```

PROC? retourne VRAI si son paramètre est une procédure définie par l'utilisateur.

PRIM? complétant la fonction précédente, PRIM? rend VRAI si son paramètre est une primitive du langage LOGO.

```
POUR IMPS
AFFICHE CONTENU
FIN
```

- Imprime toutes les procédures créées par l'utilisateur.

```
POUR AFFICHE :LST
SI VIDEP :LST (STOP)
SI PROC? PREMIER :LST (IM PREMIER :LST)
AFFICHE SP :LST
FIN
```

VIDE? :OBJ Retourne VRAI si le paramètre :OBJ est liste vide ou mot vide.

```
EC VIDE? :ARBRE
FAUX

DONNE "ARBRE
EC VIDE? :ARBRE
VRAI

DONNE "ARBRE"
EC VIDE? :ARBRE
VRAI
```

Comparaison entre éléments du langage :

EGAL? :OBJ1 :OBJ2

Il s'applique aussi bien aux nombres, aux mots et aux listes. Dans la plupart des cas, il peut être utilisé sous sa forme infixe (=), mais dans les expressions complexes, il permet de lever l'ambiguité concernant la priorité des opérateurs.

Exemples :

```
EC MOT "A :B = MOT :C :D
AFAUX
```

Bien entendu, cette expression est incorrecte, car elle est interprétée comme suit :

```
    MOT "A ( :B = MOT :C :D)
soit MOT "A          FAUX
d'où AFAUX
```

L'expression correcte est :

```
EGAL? MOT "A :B MOT :C :D
```

qui est interprétée : y a-t-il égalité entre (MOT "A :B) et (MOT :C :D) ?

PLP? et PLG? (« plus petit que » et « plus grand que ») ne s'appliquent que sur des nombres.

Exemples :

```
EC PLP? 3 2
FAUX
EC PLG? 5 0
VRAI
```

Comme EGAL?, ils peuvent être remplacés par leur équivalent infixé : "<" et ">". Les mauvaises interprétations sont là aussi possibles.

MEMBRE? :OBJ :LST
Nous indique si l'objet :OBJ appartient à la liste :LST. L'objet pouvant être un mot, un nombre ou une liste.

Combinaisons de prédicats :

NON pred Rend le contraire du prédicat pred.

```
POUR PLEIN :OBJ
RENDS NON VIDE? :OBJ
FIN
```

```
POUR DIFFÉRENT :OBJ1 :OBJ2
RENDS NON ÉGAL? :OBJ1 :OBJ2
FIN
```

ET PRED1 PRED2 Ne rend VRAI que si PRED1 et PRED2 sont vrais tous les deux.

```
POUR COMPRIS :N :INF :SUP
RENDS ET (LG? :N :INF) (PLP? :N :SUP)
FIN
```

Cette fonction retourne VRAI lorsque : N est compris entre les bornes :INF et :SUP

OU PRED1 PRED2 rend VRAI si au moins l'un des deux prédicats PRED1 ou PRED2 est vrai

```
POUR OUEX :PRED1 :PRED2
RENDS ET (OU :PRED1 :PRED2) (NON ET :PRED1 :PRED2)
FIN
```

• (ou exclusif)

TRADUCTEUR DE CHIFFRES ROMAINS EN CHIFFRES ARABES

On se propose de réaliser un programme particulièrement destiné aux archéologues : en effet, il traduit automatiquement un nombre romain en nombre arabe.
Par exemple, la traduction de XIX est 19.

comment faire ?

Il faut associer à chaque chiffre romain son équivalent arabe (exemples : V a pour équivalent 5, M a pour équivalent 1000). Cette tâche est confiée à la procédure INIT.
Pour effectuer la traduction, il faut parcourir un à un, en partant de la gauche, tous les chiffres de ce nombre et les ajouter :
CXXVI = 100 + 10 + 10 + 5 + 1 = 126

Ce procédé n'est valable que si tous les chiffres sont rencontrés dans l'ordre décroissant. Mais, si un chiffre plus petit précède un plus grand, il faut retrancher ce plus petit du total (ils sont fous ces romains !)
CXXIV = 100 + 10 + 10 — 1 + 5 = 124.

L'exploration se faisant de gauche à droite, il faut comparer chaque chiffre avec son voisin de droite afin de détecter une éventuelle inversion. La procédure-fonction INVERSE se charge de ce travail. Elle retourne VRAI s'il y a inversion.

La procédure-fonction TRADUCT explore un à un tous les chiffres en ajoutant ou retranchant leur valeur suivant le cas. Quand elle explore le dernier chiffre (ou s'il n'y en a qu'un) elle retourne directement sa valeur.
Pour savoir si un mot n'a qu'un seul caractère, on écrit :
SI" = SP :MOT
TRADUCT étant une fonction, vous pouvez l'utiliser dans une expression arithmétique :
EC (TRADUCT "XXIV) * (TRADUCT "CXIX)
2856

à vous de jouer...

Réalisez le programme qui traduit en sens inverse (de l'arabe en romain).
Recréez l'ensemble des fonctions de calcul (SOMME PRODUIT DIFFÉRENCE QUOTIENT RESTE) qui travaillent directement sur les caractères romains sans passer par la traduction en arabe.

programmer : TRADUCTION DE CHIFFRES ROMAINS EN CHIFFRES ARABES

```
POUR INIT
DONNE "I 1
DONNE "V 5
DONNE "X 10
DONNE "L 50
DONNE "D 500
DONNE "C 100
DONNE "M 1000
FIN

POUR TRADUCT :N
SI " = SP :N (RENDS CHOSE :N)
SI INVERSE (RENDS DIFF TRADUCT SP :N CHOSE PREM :N)
RENDS SOMME CHOSE PREM :N TRADUCT SP :N
FIN
```

- PREM : N est un chiffre romain
- CHOSE PREM : N est son équivalent arabe

```
POUR INVERSE
SI 1 = COMPTE :N (RENDS FAUX)
RENDS (CHOSE PREM :N) < (CHOSE PREM SP :N)
FIN
```

- donc retourne VRAI ou FAUX

```
POUR LONG :N
SI VIDE? :N (RENDS 0)
RENDS 1 + LONG SP :N
FIN
```

LOTS D'UTILITAIRES

Maintenant que vous programmez en LOGO depuis quelque temps, vous vous êtes sans doute aperçu que certaines procédures ou fonctions se retrouvaient dans la plupart des programmes. Jusqu'à présent, vous les avez à chaque fois redéfinies.
Pour vous éviter ce travail, vous pouvez facilement vous constituer une bibliothèque. Chacun de ses « volumes » contiendra un certain nombre de procédures regroupées par thème (un lot).

Ces volumes pourraient être :

- EXTGRAPH — extensions graphiques
- TROISD — procédures pour dessins en perspective
- EXTNUM — fonctions numériques supplémentaires
- EXTLST — fonctions sur les listes
- ABREV — abréviations des primitives courantes de LOGO

Nous aurons l'occasion d'en découvrir d'autres par la suite.

Pour constituer cette bibliothèque :
- Effacer entièrement la mémoire centrale.
- Taper toutes les procédures concernant un même thème.
- Les tester rapidement.
- Les sauver sur la disquette sous le nom du volume.

Pour utiliser l'un des volumes de la bibliothèque, vous avez deux possibilités :
- Recharger le volume,
- Charger ou taper votre programme (utilisateur des procédures du lot),
- Sauver l'ensemble dans un même fichier.

ou alors :
- Ecrire votre programme,
- En début de celui-ci placer une instruction pour charger le lot d'extensions nécessaires.
- Sauver uniquement votre programme.

Vous trouverez, dans la suite du livre, plusieurs lots d'utilitaires.
A vous de vous créer votre propre bibliothèque, qui contiendra les procédures que vous utilisez fréquemment, classées comme bon vous semblera.

programmer : UTILITAIRES

Extensions graphiques (EXTGRAPH)

```
POUR POLYGONE :N :COTE
REPETE :N [AV :COTE TD 360 / :N]
FIN
```

• Trace un polygone à :N cotés chacun de longueur :COTE

```
POUR SAUTILLE :N
REPETE :N / 4 [BC AV 2 LC AV 2]
BC AV RESTE :N 4
FIN
```

• Trace un pointillé sur une longueur :N

```
POUR EFFACE :N
DONNE "COULEUR CC
FCC CF
AV :N BC
FCC :COULEUR
FIN
```

• Efface tout ce qui coupe la route de la tortue sur une longeur :N

Extensions pour graphismes en trois dimensions

```
POUR ENF :L
DONNE "ANGLE CAP
FCAP 45
AV :L × 0 75
FCAP :ANGLE
FIN
```

• Fait «enfoncer» la tortue dans l'écran

```
POUR REM :L
DONNE "ANGLE CAP
FCAP 225
AV :L × 0 75
FCAP :ANGLE
FIN
```

• Fait «remonter» la tortue

Exemples d'utilisation :

```
POUR TUNNEL :N
REPETE :N [BC CARRE LC ENF 10]
FIN

POUR CUBE :COTE
REPETE 4 [AV :COTE ENF :COTE REM :COTE TD 90]
ENF :COTE
REPETE 4 [AV :COTE TD 90]
FIN
```

Extensions numériques (EXTNUM)

```
POUR PAIR :N
RENDS 0 = RESTE :N2
FIN
```

- Teste la parité d'un nombre retourne VRAI ou FAUX.

```
POUR PI
RENDS 31416
FIN
```

```
POUR SUPEG :N1 :N2
RENDS NON :N1 < :N2
FIN
```

- Comparaison au sens large de deux nombres

```
POUR INFEG :N1 :N2
RENDS NON :N1 > :N2
FIN
```

```
POUR MULT :N1 :N2
RENDS 0 = RESTE :N1 :N2
FIN
```

- Testent si un nombre est multiple ou diviseur d'un autre

```
POUR DIVIS :N1 :N2
RENDS 0 = RESTE :N2 :N1
FIN
```

```
POUR EXP :N :E
SI :E = 0 [RENDS 1]
SI :E < 0 [RENDS ( EXP :N :E + 1) / :N] [RENDS :N × EXP :N :E — 1]
FIN
```

- Retourne :N à l'exposant :E

```
POUR MIN :LST
RENDS MINI :LST PREM :LST
FIN
```

- Retourne le plus petit élément de :LST
  Utilise la fonction MINI

```
POUR MINI :LST :N
SI :LST = [ ] [RENDS :N]
SI :N < PREM :LST [RENDS MINI SP :LST :N] [RENDS MINI SP :LST
PREM :LST]
FIN
```

```
POUR MAX :LST
RENDS MAXI :LST PREM :LST
FIN
```

- Symétrique de la fonction précédente

```
POUR MAXI :LST :N
SI :LST = [ ] [RENDS :N]
SI :N > PREM :LST [RENDS MAXI SP :LST :N] [RENDS MAXI SP :LST
PREM :LST]
FIN
```

Extensions pour le traitement de listes (EXTLST)

```
POUR POSIT :MOT :LST
SI :LST = ( ) (RENDS 0)
```

- Retourne la position d'un mot dans une liste

```
POUR SUP :MOT :LST
SI :LST = ( ) (RENDS ( ))
SI EGAL? :MOT PREM :LST (RENDS SP :LST)
SI :MOT = PREM :LST (RENDS 1) (RENDS 1 + POSIT :MOT SP :LST)
FIN
```

- Supprime une fois :MOT dans :LST

```
POUR SUPP :MOT :LST
SI :LST = ( ) (RENDS ( ))
SI EGAL? :MOT PREM :LST (RENDS SUPP :MOT SP :LST)
RENDS MP PREM :LST SUPP :MOT SP :LST
FIN
```

- Supprime tous les :MOT de :LST

```
POUR AJOUTED :MOT :LST
SI MEMBRE? :MOT :LST (RENDS :LST)
RENDS MD :MOT :LST
FIN

POUR AJOUTEP :MOT :LST
SI MEMBRE? :MOT :LST (RENDS :LST)
RENDS MP :MOT :LST
FIN
```

- AJOUTED (AJOUTEP) retourne :LST avec :MOT rajoute en dernier (premier) mais seulement si :MOT n'appartient pas déjà à :LST

```
POUR POUR.TOUT :LST :INSTR
SI EGAL? :LST ( ) (RENDS ( ))
RENDS MP EXEC PH :INSTR PREM :LST POUR.TOUT SP :LST :INSTR
FIN
```

- Applique :INSTR (liste d'instructions à tous les éléments de :LST et retourne la liste des résultats

Abréviations

```
POUR CAT
CATALOGUE
FIN
```

```
POUR PR :LST
RENDS PREM :LST
FIN
```

```
POUR PR :LST
RENDS DER :LST
FIN
```

B4 NOMBRES PREMIERS

Rapidité d'un algorithme.

Ecrivez une fonction qui retourne VRAI quand un nombre est premier et FAUX dans le cas contraire.
Pour les non-matheux, un nombre premier est un nombre qui ne se divise que par lui-même et par 1.
Exemples : 1, 2, 3, 5, 7, 11 sont premiers,
4 se divise par 1, 2 et lui-même : il n'est pas premier.

comment faire ?

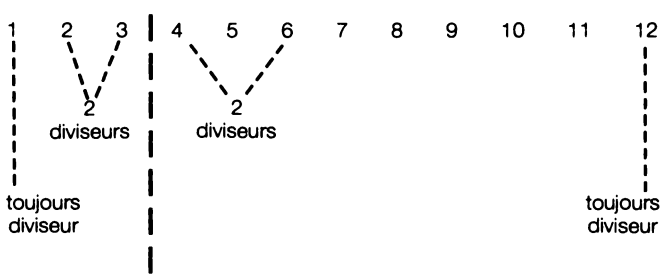
La première solution qui vient à l'esprit est la suivante : essayez de diviser le nombre testé par tous ceux qui lui sont inférieurs, sauf 1 bien sûr. Si on lui trouve au moins un diviseur, le nombre testé n'est pas premier.
Programmé en LOGO, cela donne les fonctions PREMIER1 et TEST.DIV1.
Cette première solution est tout à fait valable, mais elle fait du travail pour rien.
En effet, après avoir vérifié que le nombre testé est indivisible par deux, il n'est plus nécessaire de vérifier sa divisibilité par les multiples de 2.

PREMIER2 et TEST.DIV2 illustrent cette deuxième solution. Elle est bien sûr plus rapide que la première. Vous pouvez le vérifier vous-même sur votre ordinateur. La différence est d'autant plus grande que le nombre testé est grand.
Essayez par exemple :

- PREMIER1 101
  (temps d'exécution)
  VRAI
- PREMIER2 101
  (temps d'exécution)
  VRAI

Il est encore possible de diminuer le temps d'exécution en remarquant que les diviseurs d'un nombre sont répartis en quantités égales de chaque côté de sa racine carrée.

Exemple :



On peut donc arrêter notre test de divisibilité à la racine carrée du nombre testé : si l'on n'a pas trouvé de diviseurs jusque là, il n'y en aura pas plus loin non plus. Le gain de temps est très important.

à vous de jouer...

Perfectionnez ces programmes pour qu'ils marchent aussi pour les nombres négatifs.
Trouvez une quatrième façon, encore plus rapide en généralisant la première remarque que nous avons faite :
Quand la divisibilité par n a été vérifiée, il n'est plus nécessaire de faire le test pour tous les multiples de n.

programmer : NOMBRES PREMIERS

```
POUR PREMIER1 :N
RENDS TEST.DIV1 2
FIN

POUR TEST.DIV1 :Q
SI NON :Q < :N (RENDS VRAI)
SI 0 = RESTE :N :Q (RENDS FAUX) (RENDS TEST.DIV1 :Q + 1)
FIN

POUR PREMIER2 :N
SI 0 = RESTE :N 2 (RENDS FAUX) (RENDS TEST.DIV2 3)
FIN

POUR TEST.DIV2 :Q
SI NON :Q < :N (RENDS VRAI)
SI 0 = RESTE :N :Q (RENDS FAUX) (RENDS TEST.DIV2 :Q + 2)
FIN

POUR PREMIER3 :N
SI 0 = RESTE :N 2 (RENDS FAUX) (RENDS TEST.DIV3 3)
FIN

POUR TEST.DIV3 :Q
SI :Q > RC :N (RENDS VRAI)
SI 0 = RESTE :N :Q (RENDS FAUX) (RENDS TEST.DIV3 :Q + 2)
FIN
```

CHAUD OU FROID

Jeu avec la tortue

La tortue LOGO va nous montrer qu'elle a un tempérament très joueur et qu'il n'est pas facile de découvrir ses cachettes.
Elle cache un trésor dans l'écran. A nous de le découvrir. Quand on s'en rapproche, elle répond « tu brûles » sinon, « tu refroidis ».

comment faire ?

La première chose est de choisir une cachette. On utilisera HASARD pour que le joueur ne puisse absolument pas déterminer sa position. Ensuite, la tortue exécute les ordres du joueur. Suivant qu'elle s'est rapprochée ou éloignée du trésor, elle affiche « tu brûles » ou « tu refroidis ». Il aura donc fallu conserver l'ancienne position pour pouvoir comparer avec la position actuelle.
Enfin, quand le joueur sera très proche du trésor (moins de 10 pas), on peut lui annoncer qu'il a gagné.

Pour mémoriser l'ancienne distance de la tortue par rapport au trésor, nous avons utilisé la variable ANC.DIST.
La procédure principale CHAUD.OU.FROID :

- vide l'écran,
- explique le jeu,
- choisit la cachette,
- mémorise la distance entre la tortue et le trésor,
- lance le jeu.

Le jeu est géré par la procédure JEU qui comporte deux tests. Le premier détermine si le joueur a gagné et le deuxième, s'il s'est rapproché ou éloigné du trésor.
La fonction DIST retourne un nombre correspondant à la distance entre la tortue et le trésor, calculée suivant la formule :

$$\text{DISTANCE} = \text{RACINE CARRÉE}\left((\text{Xtortue} - \text{Xtrésor})^2 + (\text{Ytortue} - \text{Ytrésor})^2\right)$$

Il est nécessaire de définir une autre fonction pour calculer le carré d'un nombre.

à vous de jouer...

Remarque : A la suite d'une rotation de la tortue, le message « tu refroidis » apparaît alors qu'elle n'a pas changé de place. Il serait intéressant de modifier le programme pour qu'elle ne dise rien ou alors, qu'elle émette un message approprié.
Ce jeu fait seulement appel à la déduction. Il est possible d'y ajouter le facteur « mémoire » en imposant à la tortue de ne pas laisser la trace de ses déplacements. Ecrivez une procédure qui demande au joueur s'il désire ou non voir la trace de la tortue.

programmer : CHAUD OU FROID

```
POUR CHAUD.OU.FROID
VT VE
EXPLICATIONS
DONNE "CACHETTE PH ( 100 - HASARD 200) ( 50 - HASARD 100)
DONNE "ANC.DIST DIST
JEU
FIN

POUR EXPLICATIONS
EC [UN TRESOR EST CACHE DANS L'ECRAN]
EC [DONNEZ MOI LES INSTRUCTIONS QUI ME PERMETTRONT DE
M'EN APPROCHER]
EC [EX : AV 40 TD 90 RE 5]
FIN

POUR JEU
EXEC LL
SI DIST < 10 [EC "GAGNE STOP]
SI :ANC.DIST < DIST [EC [TU BRULES]] [EC [TU REFROIDIS]]
DONNE "ANC.DIST DIST
JEU
FIN

POUR DIST
RENDS ENT RC [CARRE PREM :CACHETTE - PREM POS) + (CARRE
DER :CACHETTE - DER POS)
FIN

POUR CARRE :N
RENDS :N × :N
FIN
```

LES LISTES DE PROPRIÉTÉS

Lot d'utilitaires.

Souvent, vous aurez besoin « d'attacher » à un mot LOGO, une ou plusieurs propriétés. Par exemple, « COULEUR », « AMIS », « APPARTIENT » peuvent être des propriétés.

COULEUR CIEL ————> BLEU
COULEUR CRAYON ————> ROUGE
APPARTIENT CRAYON ————> PAUL
AMIS PAUL ————> [PIERRE JACQUES JEAN]

Voici les outils qui vous permettront d'utiliser les listes de propriétés :
PUT définit une propriété associée à un mot
Ex : PUT "COULEUR "CIEL "BLEU
GET retourne ce que vaut la propriété d'un objet donné.
Ex : EC GET "COULEUR "CIEL
BLEU
PROP permet de tester si une propriété a déjà été utilisée.
Ex : EC PROP "COULEUR "CIEL
VRAI
EC PROP "AMIS "JACQUES
FAUX
DEL annule une propriété.
Ex : DEL "COULEUR "CIEL
EC PROP "COULEUR "CIEL
FAUX
Il existe d'autres façons de gérer les listes de propriétés, mais celle-ci présente l'avantage d'être simple et rapide à l'exécution.

programmer : LES LISTES DE PROPRIETES

```
POUR PUT :PROP :OBJ :VAL
DONNE MOT :PROP :OBJ :VAL
FIN

POUR GET :PROP :OBJ
RENDS CHOSE MOT :PROP :OBJ
FIN

POUR PROP :PROP :OBJ
RENDS NOM ? MOT :PROP :OBJ
FIN

POUR DEL :PROP :OBJ
EFN MOT :PROP :OBJ
FIN
```

TRADUCTEUR DE LANGUES

Utilisation des listes de propriétés.

Réalisez un programme qui soit capable d'effectuer une traduction bilingue (par exemple anglais/français). Il s'agira bien sûr d'une traduction mot à mot : il serait prétentieux de lui demander une version française de HAMLET ! Cependant, ce petit programme peut s'avérer très utile pour traduire des programmes : les changer de machine ou récupérer du LOGO anglais.

comment faire ?

Il faut simplement que ce programme matérialise une liaison entre le mot anglais et le mot français. Cette liaison doit pouvoir être parcourue dans les deux sens. Dans notre premier programme, nous avons utilisé une double affectation : chaque mot de vocabulaire que nous apprenons à LOGO devient donc une variable qui contient sa traduction.
Cela . . che presque dans tous les cas. Un problème apparaît quand un mot anglais à un synonyme en français (ou vice versa).
Exemple d'utilisation :

- EC TRADUCT "YES
  OUI
- EC TRADUCT "HOUSE
  INCONNU
- APPRENDS "HOUSE "MAISON
- EC TRADUCT "MAISON
- MAISON

Dans la deuxième version, les listes de propriété (voir le lot d'utilitaires correspondant) ont été mises à contribution. A chaque mot, peut être associée la propriété "FRANÇAIS ou "ANGLAIS.
Ce deuxième programme est un peu plus compliqué mais n'a pas l'inconvénient du premier.

à vous de jouer...

Ecrivez un traducteur multilangues.
Ajoutez quelques procédures pour pouvoir traduire un programme complet depuis le LOGO anglais. Vous pouvez par exemple lire l'original directement sur disquette ou cassette et imprimer la traduction.

programmer : TRADUCTEUR DE LANGUES

```
POUR APPRENDS :MOT1 :MOT2
DONNE :MOT1 :MOT2
DONNE :MOT2 :MOT1
FIN

POUR TRADUCTION :MOT
SI NOM? :MOT (RENDS CHOSE :MOT) (RENDS "INCONNU)
FIN

POUR APPRENDS :MOT.F :MOT.A
PUT "ANG :MOT.F :MOT.A
PUT "FRANC :MOT.A :MOT.F
FIN

POUR FRANCAIS :MOT
SI PROP "FRANC :MOT (RENDS GET "FRANC :MOT) (RENDS "INC
ONNU)
FIN

POUR ANGLAIS :MOT
SI PROP "ANG :MOT (RENDS GET "ANG :MOT) (RENDS "INCONNU)
FIN
```

MASTERMIND

LOGO choisit quatre chiffres au hasard (qui peuvent se répéter) et vous devez les découvrir. Après chacun de vos essais, le programme vous dit le nombre de chiffres trouvés et bien placés (BP) et le nombre de chiffres trouvés mais non placés (NP).

Exemple, l'ordinateur a choisi en secret : 1 5 2 2
vous proposez 1 2 9 2
la réponse que LOGO vous donne est BP: 2 NP: 1

Réfléchissez bien avant de jouer car LOGO compte les coups.

comment faire ?

La procédure d'entrée (celle que vous tapez au clavier pour lancer le jeu) est MASTERMIND. Elle choisit tout d'abord au hasard, un nombre de quatre chiffres, puis, vous donne votre chance pour le premier tour de jeu (JEU 1).

Les fonctions BP et NP calculent le nombre de chiffres « Bien Placés » et « Non Placés » par rapport au code à découvrir.

à vous de jouer...

Quand vous serez imbattable à ce jeu, compliquez le : modifiez le programme pour que le code à découvrir puisse comporter aussi des lettres.
Vous pouvez faire en sorte que le nombre secret soit composé d'un nombre variable de chiffres (de 4 à 6 par exemple). Ceci rend le jeu très compliqué.

programmer : MASTERMIND

```
POUR MASTERMIND
DONNE "N.SECRET ALEATOIRE 4
VT EC (QUE JOUEZ VOUS ?) EC "
JEU 1 LL
FIN

POUR ALEATOIRE :N                          • Rends une liste composée de :N
SI EGAL? :N 0 (RENDS ( ))                    chiffres tirés au hasard
RENDS MP HASARD 10 ALEATOIRE :N - 1
FIN

POUR JEU :COUP :ESSAI
TAPE PH (BIEN PLACES:) BP :ESSAI :N.SECRET REPETE 4 (TAPE CAR 32)
EC PH (NON PLACES:) NP :ESSAI :N.SECRET
SI NON EGAL? BP :ESSAI :N.SECRET 4 (JEU :COUP + 1 ") (RESUL
TATS)
FIN

POUR BP :N1 :N2
SI OU VIDE? :N1 VIDE ? :N2 (RENDS 0)
SI EGAL? PREM :N1 PREM :N2 (RENDS 1 + BP SP :N1 SP :N2)
RENDS BP SP :N1 SP :N2
FIN

POUR NP :N1 :N2
SI VIDE? :N1 (RENDS - BP :ESSAI :N.SECRET)
SI NON MEMBRE ? PREM :N1 :N2 (RENDS NP SP :N1 :N2)
RENDS 1 + NP SP :N1 SUPP PREM :N1 :N2
FIN

POUR SUPP :MOT :LST
SI VIDE? :LST (RENDS ( ))
SI EGAL? PREM :LST :MOT (RENDS SP :LST)
RENDS MP PREM :LST SUPP :MOT SP :LST
FIN

POUR RESULTATS
EC " EC "
EC PH (VOUS AVEZ TROUVE EN) PH :COUP "COUPS
FIN
```

MACHINE A TOUT FAIRE

Paramétrage des entrées/sorties.

Réalisez un programme qui remplace :
- Une machine à écrire (tout ce qui est tapé au clavier est envoyé sur l'imprimante).
- Un pense-bête (un petit texte tapé au clavier est envoyé sur disquette ou cassette).
- Ce texte pourra être soit rappelé à l'écran, soit imprimé.

comment faire ?

A la mise sous tension, LOGO lit sur le clavier et écrit sur l'écran mais il est possible de changer cela. On peut lui demander, de lire ou écrire sur la disquette, d'écrire sur l'imprimante, etc.

Notre problème se résoud donc très simplement :
- Créer une procédure qui lise puis écrive des listes (TRANSFERT).
- Avant d'appeler cette procédure, modifier les entrées/sorties.
- MACHINE A ÉCRIRE : entrée au clavier
  sortie sur imprimante.
- PENSE BÊTE : entrée au clavier
  sortie sur disquette
- LECTURE DU PENSE BÊTE : entrée disquette
  sortie écran
- IMPRESSION DU PENSE BÊTE : entrée disquette
  sortie imprimante.

Pour rendre l'utilisation du programme plus agréable, on peut ajouter une procédure « chapeau » (MACHINE) qui demande le choix de l'utilisateur et appelle la procédure adéquate.

L'utilisateur indique son choix en tapant seulement une lettre (sans <RETURN>). Celle-ci correspond à une procédure qui est exécutée aussitôt.
Attention : LOGO ne peut exécuter que des listes : c'est pourquoi le caractère lu est assemblé avec le mot vide (") pour former une liste.

à vous de jouer...

Ecrivez un programme qui vous permette de conserver simultanément plusieurs textes sur la disquette (il vous faudra bien sûr plusieurs fichiers). Vous pouvez par exemple, vous créer un agenda classé par jour (7 fichiers : LUNDI, MARDI... DIMANCHE) et par rubriques (téléphoner à..., écrire à..., rendez-vous avec...)

programmer : MACHINE A TOUT FAIRE

```
POUR MACHINE
VT
EC [MACHINE A ECRIRE (E)]
EC [PENSE BETE (P)]
EC [LIRE FICHIER (L)]
EC [IMPRIMER LE FICHIER (I)]
EC "
EXEC LL
MACHINE
FIN

POUR TRANSFERT :LIGNE :ECHO
SI :ECHO [EC :LIGNE]
SI :LIGNE = [#] [STOP]
TRANSFERT LL :ECHO
FIN

POUR MESSAGE
VT EC [TAPER "#" POUR ARRETER] EC " EC "
FIN

POUR E
MESSAGE
SORTIE 4
TRANSFERT LL FAUX
SORTIE 2
FIN

POUR P
MESSAGE
SORTIE 8
TRANSFERT LL FAUX
SORTIE 2
SAUVED "TEXTE
FIN

POUR L
VT
CHARGE "TEXTE
ENTREE 8
TRANSFERT LL VRAI
ENTREE 2
FIN

POUR I
VT
RAMENE "TEXTE
SORTIE 4 ENTREE 8
TRANSFERT LL VRAI
ENTREE 2 SORTIE 2
FIN
```

• "ECHO est un variable booléenne elle contient soit VRAI soit FAUX.

RÉPERTOIRE TÉLÉPHONIQUE

Remplacez avantageusement votre carnet d'adresses par un micro-ordinateur.
Ce programme vous offrira les possibilités de :

- Consulter : retrouver un numéro de téléphone à partir d'un nom.
- Ajouter : pour vos nouvelles relations.
- Effacer : en cas de changement.
- Afficher tous les noms avec leur numéro correspondant.
- Mémoriser tout le fichier sur la disquette.
- Récupérer le fichier depuis la disquette.

comment faire ?

Le principe que nous avons utilisé est simple : chaque nom de personne est considéré comme une variable dans laquelle on range le numéro de téléphone.
Ainsi : EC : DUPONT
882.23.21
On constitue d'autre part un répertoire : la liste de tous les noms inscrits.

La procédure de lancement est RÉPERTOIRE : elle initialise la liste des noms et vous invite au MENU.
Une fois n'est pas coutume, nous vous proposons une représentation imagée de fonctionnement de LOGO:
Imaginez une corde à linge sur laquelle seraient épinglés tous les noms : c'est notre liste :RÉPERTOIRE.
A chaque nom, est attachée une ficelle au bout de laquelle est inscrit le numéro de téléphone correspondant.
Retrouver un numéro revient à aller voir ce qu'il y a au bout de la ficelle attachée au nom choisi.
Supprimer un nom, c'est le décrocher de la corde. Le lien entre le nom et le numéro (la ficelle) existe toujours mais, puisque le nom ne fait plus partie de :RÉPERTOIRE, le programme ne peut plus le retrouver.

à vous de jouer...

Créez la procédure TRIER qui classe la liste :RÉPERTOIRE par ordre alphabétique.
Modifiez le programme pour qu'il vous permette de retrouver un numéro de téléphone même si vous ne connaissez pas l'orthographe exacte du nom. A vous de définir les critères de ressemblance. Par exemple, plus des trois quart de leurs lettres communes...

programmer : REPERTOIRE TELEPHONIQUE

```
POUR REPERTOIRE
DONNE "REPERTOIRE ( )
MENU
FIN

POUR MENU
VT EC (VOULEZ VOUS...) EC " EC "
EC (AJOUTER (UN NOM))
EC (CONSULTER)
EC (EFFACER (UN NOM))
EC (AFFICHER (TOUS LES NOMS))
EC "
EC (MEMORISER (SUR DISQUETTE))
EC (CHARGER (DEPUIS LA DISQUETTE))
EC " EC "
EXEC LL
MENU
FIN

POUR AJOUTER
VT TAPE PH (NOM DE LA PERSONNE) "
DONNE "NOM PREM LL
DONNE "REPERTOIRE MD :NOM :REPERTOIRE
TAPE PH (SON TELEPHONE ?) "
DONNE :NOM LL
FIN

POUR CONSULTER
VT TAPE PH (QUEL NOM ?) "
DONNE "NOM PREM LL
SI INCONNU (EC "INCONNU ATTENDRE STOP)
EC " EC PH (SON TELEPHONE :) CHOSE :NOM
ATTENDRE
FIN

POUR EFFACER
VT TAPE PH (QUEL NOM ?) "
DONNE "NOM PREM LL
SI INCONNU (EC "INCONNU ATTENDRE STOP)
SUPPRIME :NOM
FIN
```

```
POUR AFFICHER
VT
SI :REPERTOIRE = ( ) (EC FICHIER VIDE) ATTENDRE STOP)
IMPRIME :REPERTOIRE
ATTENDRE
FIN

POUR MEMORISER
SAUVE "FICHIER
FIN

POUR CHARGER
RAMENE "FICHIER
FIN

POUR IMPRIME :LST
SI :LST = ( ) (STOP)
EC PH PH PH PREM :LST "" CHOSE PREM :LST
IMPRIME SP :LST
FIN

POUR LIT.NUMERO :LST
SI :LST = ( ) (STOP)
DONNE PREM :LST DER :LL
LIT.NUMERO SP :LST
FIN

POUR SUPPRIME :NOM
DONNE "REPERTOIRE SUPP :NOM :REPERTOIRE
FIN

POUR SUPP :MOT :LST
SI :LST = ( ) (RENDS ( ))
SI :MOT = PREM :LST (RENDS SUPP :MOT SP :LST)
RENDS PH PREM :LST SUPP :MOT SP :LST
FIN

POUR ATTENDRE
DONNE "A LISCAR
FIN

POUR INCONNU
RENDS NON MEMBRE? :NOM :REPERTOIRE
FIN
```

TOURS DE HANOÏ

Ce jeu bien connu est composé de trois tiges verticales et d'un certain nombre de disques, de diamètres tous différents. Au début, tous les disques sont empilés sur l'une des tiges, le plus grand en bas. Le but du jeu est de déplacer la totalité des disques sur une des deux autres tiges en respectant les règles suivantes :

- Ne déplacer qu'un seul disque à la fois.
- Ne jamais déposer un disque plus grand sur un plus petit.
- Réaliser le transfert de la pile en un minimum de coups.

Nous allons réaliser le programme qui vous permettra de jouer avec le nombre de disques que vous désirez (9 au maximum). LOGO dessine l'état des piles et effectue les déplacements que vous lui indiquez. Il refusera bien entendu les déplacements impossibles.

comment faire ?

Les trois piles sont repérées par les lettres A, B et C. A étant la pile de départ et C la pile d'arrivée. Pour déplacer un disque, on indiquera tout d'abord, la pile où il se trouve, puis, la pile où on désire le poser.
Le jeu s'arrête quand tous les disques ont été amenés sur la pile C.
Chaque disque est repéré par sa taille (10 pour le premier, 20 pour le deuxième...) Pour LOGO, chaque pile est une liste de disques. Le premier élément de cette liste correspond au disque inférieur de la pile.

La procédure d'entrée JOUER.A.HANOÏ :

- Lit le nombre de disques,
- Prépare l'écran et initialise les piles,
- Fait appel à la procédure EFFECTUER.DÉPLACEMENTS.

EFFECTUER.DÉPLACEMENTS :

- Range dans la variable DEP, le nom de la pile où se trouve le disque à déplacer et dans ARR, le nom de la pile d'arrivée.
- Vérifie que le déplacement soit possible et, s'il l'est, appelle DÉPLACER qui se charge de le réaliser.

ÔTER.DE :

- Positionne la tortue au sommet de la pile de départ.
- Efface le disque supérieur (dessine dans la couleur du fond),
- Ote ce disque dans la liste de cette pile.

POSER.SUR :

- Ajoute le disque déplacé à la liste de la pile d'arrivée.
- Le dessine après avoir positionné la tortue.

à vous de jouer...

Modifiez le programme de telle sorte que le joueur ait le choix de la pile de départ et de la pile d'arrivée.
Lorsque le joueur dépasse le nombre de coups optimal, ajouter l'impression d'un message. Le nombre de coups optimal se calcule par :

(2 puissance :NB.DISC) — 1

programmer : HANOI

```
POUR JOUER.A.HANOI
VT EC (DEPLACER DE A VERS C) EC "
TAPE PH (COMBIEN DE DISQUES ?) " DONNE "NB.DISC PREM LL
PREPARE.ECRAN
INITIALISE.PILES
EFFECTUER.DEPLACEMENTS
FIN

POUR PREPARE.ECRAN
VE CT TD 90
FIN

POUR INITIALISE.PILES
DONNE "A ( )
EMPILER :NB.DISC "A
DONNE "B ( )
DONNE "C ( )
FIN

POUR EMPILER :N :PILE
SI :N = 0 (STOP)
POSER.SUR :PILE 10 × :N
EMPILER :N - 1 :PILE
FIN

POUR EFFECTUER.DEPLACEMENTS
SI :NB.DISC = COMPTE :C (EC "BRAVO STOP)
EC "TAPE PH (PILE DE DEPART ?) "DONNE "DEP LISCAR
EC :DEP
TAPE PH (PILE D'ARRIVEE ?) "DONNE "ARR LISCAR EC :ARR
SI AUTORISE (DEPLACER DER CHOSE :DEP) (EC "IMPOSSIBLE)
EFFECTUER.DEPLACEMENTS
FIN

POUR AUTORISE
SI ( ) = CHOSE :ARR (RENDS VRAI)
SI ( ) = CHOSE :DEP (RENDS FAUX)
RENDS (DER CHOSE :ARR) > (DER CHOSE :DEP)
FIN

POUR DEPLACER :DISQUE
OTER.DE :DEP :DISQUE
POSER.SUR :ARR :DISQUE
FIN
```

```
POUR OTER.DE :PILE :DISQUE
POSITIONNE :PILE
FNC 0 BC RE :DISQUE / 2 AV :DISQUE
DONNE :PILE SD CHOSE :PILE
FIN

POUR POSER.SUR :PILE :DISQUE
DONNE :PILE MD :DISQUE CHOSE :PILE
POSITIONNE :PILE
FNC 1 BC RE :DISQUE / 2 AV :DISQUE
FIN

POUR POSITIONNE :PILE
LC
DONNE "Y 6 × COMPTE CHOSE :PILE
SI :PILE = "A (FPOS PH - 80 :Y)
SI :PILE = "B (FPOS PH 0 :Y)
SI :PILE = "C (FPOS PH 80 :Y)
FIN
```

TOURS DE HANOÏ (SUITE)

Nous vous proposons maintenant de faire jouer LOGO. Nous n'allons bien sûr pas lui détailler chaque mouvement, mais lui enseigner une méthode de résolution valable quel que soit le nombre de disques. Evidemment, il réalisera la performance en un minimum de coups.

comment faire ?

Amener N disques de la pile de départ (A) à la pile d'arrivée (C) revient à :
- Déplacer les N-1 premiers disques de A vers B (pile intermédiaire),
- Déplacer le disque qui reste sur A vers C,
- Déplacer les N-1 disques de B sur C.

Bien entendu, déplacer N-1 disques de A vers B ne peut pas se faire en une seule opération. Nous la décomposons de la même façon que précédemment en prenant A comme pile de départ, B comme pile d'arrivée C devenant la pile intermédiaire. Cette décomposition se poursuit jusqu'à n'avoir qu'un seul disque à déplacer.

Regardons ce qui se passe avec trois disques :
- Déplacer 3 de A vers C :

c'est
* Déplacer 2 de A vers B :

qui revient à :
- Déplacer 1 de A vers C
- Déplacer 1 de A vers B
- Déplacer 1 de C vers B

puis
* Déplacer 1 de A vers C

enfin
* Déplacer 2 de B vers C :

qui revient à
- Déplacer 1 de B vers A
- Déplacer 1 de B vers C
- Déplacer 1 de A vers C

La procédure clé, HANOÏ, est calquée sur ce schéma.

La fonction INT détermine la pile intermédiaire connaissant les piles de départ et d'arrivée.

Les autres procédures utilisées sont les mêmes que celles de la première version à une différence près : JOUER.A.HANOÏ fait appel à HANOÏ (procédure « pensante ») au lieu de EFFECTUER.DÉPLACEMENTS (qui demandait les mouvements au joueur).

à vous de jouer...

Ecrivez le programme qui vous demande le mouvement à effectuer et vérifie qu'il correspond au meilleur mouvement.

programmer : TOURS DE HANOI

```
POUR JOUER.A.HANOI
VT EC (DEPLACER DE A VERS C) EC "
TAPE PH (COMBIEN DE DISQUES ?) " DONNE "NB.DISC PREM LL
PREPARE.ECRAN
INITIALISE.PILES
HANOI :A "A "C
FIN
```

- PREPARE, INITIALISE, EMPILER, DEPLACER, OTER DE, POSER SUR, POSITIONNE sont décrits dans TOURS DE HANOÏ (pages 49 et 50)

```
POUR HANOI :DISQUES :DEP :ARR
SI 1 = COMPTE :DISQUES (DEPLACER PREM :DISQUES STOP)
HANOI SP :DISQUES :DEP INT
DEPLACER PREM :DISQUES
HANOI SP :DISQUES INT :ARR
FIN

POUR INT
SI :DEP = "A (SI :ARR = "B (RENDS "C) (RENDS "B))
SI :DEP = "B (SI :ARR = "A (RENDS "C) (RENDS "A))
SI :DEP = "C (SI :ARR = "A (RENDS "B) (RENDS "A))
FIN

POUR ATTENDRE
DONNE "A LISCAR
FIN
```

FRACTAL

Graphisme récursif

Sous ce nom bizarre se cache souvent de merveilleux dessins. Les courbes fractales sont obtenues à partir d'un motif de base simple (angle, rectangle, T...) dont chacun des éléments (côté du rectangle, branche du T) est remplacé par le motif complet mais plus petit. Ce qui donne naissance à de nouveaux segments de taille plus réduite qui sont à leur tour remplacés, etc, etc. Avec certains motifs de base, la courbe finit par remplir complètement l'écran.
Celui que nous avons choisi (pour ses qualités esthétiques) est composé de quatre segments. Il est dessiné par la procédure MOTIF.DE.BASE.

comment faire ?

Il faut partir de MOTIF.DE.BASE mais y ajouter une condition : on ne fait bouger la tortue que si l'on a atteint les motifs les plus fins (longueur des segments inférieure à 2). Dans le cas contraire, on dessine un motif de base plus petit (dans notre cas, trois fois plus petit).
Essayez d'écrire vous même le programme. Si vous suivez scrupuleusement les quelques lignes ci-dessus, vous y arriverez très bien.

L'allure de FRACTAL (procédure d'entrée) montre que l'humour peut aussi se glisser dans l'écriture des programmes LOGO. Ce départ de marathon s'explique ainsi :

- A.VOS.MARQUES : efface l'écran et la tortue.
- PRET : positionne la tortue sur la gauche de l'écran.
- PARTEZ : lance l'exécution du dessin, c'est-à-dire appelle la procédure MOTIF.DE.BASE. Au départ, la longueur des segments est de 81.

Enfin, voici notre MOTIF.DE.BASE du début. Vous le reconnaissez certainement : on a simplement ajouté une condition au déplacement de la tortue. Cette condition (longueur inférieure à 10) est résumée par la fonction FINI. Si elle est vraie, on trace réellement le segment : dans le cas contraire, on refait un motif de base 3 fois plus petit.

à vous de jouer...

Ecrivez d'autres programmes de dessin récursif en partant d'autres motifs de base (ceux-ci doivent cependant être toujours simples). Pour aider votre imagination, voici quelques idées :

- Motif de base : un carré. A chaque pas, on remplace tous les angles par des carrés trois fois plus petits.
- Motif de base : une droite. On ajoute à chacune de ses extrémités, un segment perpendiculaire, de longueur moitié, comme pour former un T.

programmer : FRACTAL

```
POUR FRACTAL
A.VOS.MARQUES
PRET
PARTEZ
FIN

POUR A.VOS.MARQUES
VE
CT
FIN

POUR PRET
LC FPOS (-128 0) BC
FCAP 90
FIN

POUR PARTEZ
MOTIF.DE.BASE 81
FIN

POUR MOTIF.DE.BASE :LONG
SI FINI (AV :LONG) (MOTIF.DE.BASE QUOT :LONG 3)
TG 60
SI FINI (AV :LONG) (MOTIF.DE.BASE QUOT :LONG 3)
TD 120
SI FINI (AV :LONG) (MOTIF.DE.BASE QUOT :LONG 3)
TG 60
SI FINI (AV :LONG) (MOTIF.DE.BASE QUOT :LONG 3)
FIN

POUR FINI
RENDS PLP? :LONG 2
FIN
```

MON LOGO

La plupart des primitives concernant les listes ne sont pas indispensables. En effet, elles peuvent être définies à partir de quelques primitives élémentaires qui sont :

Construction de listes :	MP
Séparation du premier élément :	PREM, SP
Test :	EGAL?
Et bien sûr, les instructions de structuration :	SI, RENDS

Naturellement, tous les LOGO possèdent déjà la plupart des primitives que nous avons redéfinies mais il est intéressant de réfléchir sur la façon dont elles fonctionnent.

programmer : MON LOGO

```
POUR ITEM :N :LST
SI :LST = ( ) (RENDS ")
SI :N = 1 (RENDS PREM :LST)
RENDS ITEM :N - 1 SP :LST
FIN

POUR DER :LST
SI :LST = ( ) (RENDS ( ))
SI PREM SP :LST = " (RENDS PREM :LST)
RENDS DER SP :LST
FIN

POUR SAUFDER :LST
SI SP :LST = ( ) (RENDS ( ))
RENDS PH PREM :LST SAUFDER SP :LST
FIN

POUR SD :LST
RENDS SAUFDER :LST
FIN

POUR MEMBRE? :MOT :LST
SI :LST = ( ) (RENDS FAUX)
SI PREM :LST = :MOT (RENDS VRAI)
RENDS MEMBRE? :MOT SP :LST
FIN

POUR COMPTE :LST
SI :LST = ( ) (RENDS 0)
RENDS 1 + COMPTE SP :LST
FIN
```

LES ENSEMBLES

Utilitaires pour travailler sur les ensembles.

LOGO se prête bien au traitement des ensembles car ceux-ci peuvent se représenter par des listes.

Voici les fonctions de base :

INTER :E1 :E2 retourne tous les éléments communs aux ensembles E2 et E2.

UNION :E1 :E2 retourne tous les éléments de E1 et tous les éléments de E2.

COMPL :E1 :E2 retourne les éléments de E1 qui n'appartiennent pas à E2.

DELTA :E1 :E2 retourne les éléments qui appartiennent à E1 ou à E2 mais pas aux deux à la fois.

INTERVALLE :N1 :N2 retourne sous forme énumérée, un ensemble donné sous forme d'intervalle (ne fonctionne que pour un intervalle numérique).

EGAL :E1 :E2 retourne VRAI si tous les éléments de E1 appartiennent aussi à E2 et vice versa.

DISJ :E1 :E2 retourne VRAI si aucun des éléments de E1 n'appartient à E2 et vice versa.

INCL :E1 :E2 retourne VRAI si tous les éléments de E1 appartiennent aussi à E2.

programmer : ENSEMBLES

```
POUR INTER :E1 :E2
SI NON MEMBRE? PREM :E1 :E2 (RENDS INTER SP :E1 :E2)
RENDS MP PREM :E1 INTER SP :E1 :E2
FIN

POUR UNION :E1 :E2
SI :E1 = ( ) (RENDS :E2)
SI MEMBRE? PREM :E1 :E2 (RENDS UNION SP :E1 :E2)
RENDS MP PREM :E1 UNION SP :E1 :E2
FIN

POUR COMPL :E1 :E2
SI :E1 = ( ) (RENDS ( ))
SI MEMBRE? PREM :E1 :E2 (RENDS COMPL SP :E1 :E2)
RENDS MP PREM :E1 COMPL SP :E1 :E2
FIN

POUR DELTA :E1 :E2
RENDS COMPL UNION :E1 :E2 INTER :E1 :E2
FIN

POUR INCL :E1 :E2
SI :E1 = ( ) (RENDS VRAI)
SI NON MEMBRE? PREM :E1 :E2 (RENDS FAUX)
RENDS INCL SP :E1 SUP PREM :E1 :E2
FIN

POUR EGAL :E1 :E2
RENDS ET INCL :E1 :E2 INCL :E2 :E1
FIN

POUR DISJ :E1 :E2
SI OU :E1 = ( ) :E2 = ( ) (RENDS VRAI)
SI MEMBRE? PREM :E1 :E2 (RENDS FAUX)
SI MEMBRE? PREM :E2 :E1 (RENDS FAUX)
RENDS DISJ SP :E1 SP :E2
FIN

POUR INTERVALLE :N1 :N2
SI :N1 > :N2 (RENDS ( ))
RENDS MP :N1 INTERVALLE :N1 + 1 :N2
FIN

POUR SUP :MOT :LST
SI :LST = ( ) (RENDS ( ))
SI EGAL? :MOT PREM :LST (RENDS SP :LST)
RENDS MP PREM :LST SUP :MOT SP :LST
FIN
```

58 LE MOT LE PLUS LONG

Application des ensembles.

Ce programme trouve tous les mots réels que l'on peut former à partir d'un nombre de lettres quelconque. Sa puissance dépend uniquement de la taille de son dictionnaire, donc du nombre de mots que vous aurez eu la patience de lui rentrer.
Comme vous ne souhaitez sûrement pas qu'un tel travail ne serve qu'une fois, prévoyez une procédure pour stocker le dictionnaire sur cassette ou disquette.

comment faire ?

Le principe est très simple : nous allons examiner un à un tous les mots du dictionnaire. Si l'ensemble des lettres du mot examiné est inclus dans l'ensemble des lettres données au programme, alors ce mot est bon.

La procédure CHERCHE se charge de ce travail, avec l'aide de TROUVE.

La procédure APPRENDS, elle, ajoute un mot au dictionnaire de LOGO.
Pour que ce programme toume, vous aurez besoin du lot d'utilitaires sur les ensembles. Rechargez le en mémoire ou tapez au clavier les fonctions INCL et SUP (voir chapitre concemant ce lot d'utilitaires).

à vous de jouer...

Modifiez le progamme pour qu'il n'écrive que le mot le plus long et non pas tous les mots possibles.

programmer : LE MOT LE PLUS LONG

```
POUR APPRENDS :MOT
SI NOM? "DICO (DONNE "DICO MP :MOT DICO) (DONNE "DICO
MP :MOT ( ))
FIN

POUR CHERCHE :LETTRES
TROUVE :LETTRES :DICO
FIN

POUR TROUVE :LETTRES :DICO
SI :DICO = ( ) (STOP)
SI INCL EPELE PREM :DICO :LETTRES (EC PREM :DICO)
TROUVE :LETTRES SP :DICO
FIN

POUR EPELE :MOT
SI :MOT = " (RENDS ( ))
RENDS PH PREM :MOT EPELE SP :MOT
FIN
```

BATAILLE NAVALE

La flotte de LOGO (10 navires) est dissimulée sur un quadrillage. Votre mission, si vous l'acceptez, consiste à détruire ses navires en un minimum de coups. Chaque bâtiment occupe une seule case repérée par ses coordonnées X et Y comprises entre 1 et 8.
Le joueur tape les coordonnées de la case qu'il vise. Si un navire est touché, LOGO le signale par un carré rouge sur la case. Sinon, il dessine un carré bleu.
Le jeu s'arrête lorsqu'il n'y a plus de navire.

comment faire ?

Les 10 navires de LOGO sont connus par les coordonnées des cases qu'ils occupent. La variable "FLOTTE va contenir l'ensemble des positions des navires (une position est une liste de deux nombres (coordonnées)).

La procédure d'entrée s'appelle BATAILLE.NAVALE :
- Elle dessine la grille.
- Elle crée la liste des positions des bateaux.
- Elle attend l'action du joueur.

Toutes ces opérations sont confiées à des procédures secondaires (DESSIN.GRILLE, CHOIX.BATEAUX, FAIRE.TIRER).

La procédure CHOIX.BATEAUX :
- Initialise :FLOTTE
- Tire au hasard 10 positions qui viennent s'ajouter à :FLOTTE.
- Elle retourne le contenu final de "FLOTTE (liste de positions). C'est donc une fonction.

Par exemple, "FLOTTE peut contenir :
[[1 2] [2 5] [4 8] [8 5] [4 6] [5 8] [7 5] [1 4] [1 1] [7 4]]
On voit alors que :FLOTTE n'est autre qu'une liste de listes.

La procédure FAIRE.TIRER
- Arrête le jeu si tous les bateaux sont coulés (:FLOTTE est vide).
- Attend les coordonnées du tir et les range dans "TIR.
- Teste si les coordonnées sont celles d'un bateau appartenant à :FLOTTE.
- Dans ce cas, elle appelle la procédure COULE, sinon la procédure DANS.L'EAU.
- Elle recommence.

Les procédures COULE et DANS.L'EAU sont semblables :
- Elles affichent le message adapté (« plouf » ou « coulé »).
- Elles localisent le tir (appel à LOCALISATION).
- Elles fixent la couleur (en rouge : coulé ; en bleu : dans l'eau) avant de dessiner le point de tir (DESSIN).
- Seule la procédure COULE supprime le navire touché (appel à SUPPRIME).

LOCALISATION

- Positionne la tortue au milieu de la case visée.

SUPPRIME, avec l'aide de SUPP, enlève le bateau touché de :FLOTTE.
Remarques : La fonction CHOIX.NAVIRES est équivalente à CHOIX.BATEAUX mais elle est plus élégante : elle utilise la récursivité, un des atouts de LOGO.

à vous de jouer...

Ajoutez la fonction TOUCHE qui permettra au joueur d'obtenir un renseignement précieux : s'il se trouve un navire dans l'une des huit cases entourant son tir.
Dans l'affirmative, elle permettra l'affichage du message « touché ».

programmer : BATAILLE NAVALE

```
POUR BATAILLE.NAVALE
DESSIN.GRILLE
DONNE "FLOTTE CHOIX.BATEAUX
FAIRE.TIRER
FIN

POUR DESSIN.GRILLE
VE VT
FCC 7
LC FPOS [ 4 6] BC
REPETE 9 [TRAIT]
TG 90 AV 10
REPETE 9 [TRAIT]
CT
FIN

POUR TRAIT
BC AV 80 RE 80 LC TD 90 AV 10 TG 90
FIN

POUR CHOIX.BATEAUX
DONNE "FLOTTE [ ]
REPETE 10 [DONNE "FLOTTE MD PH 1 + HASARD 8 1 + HASARD 8 :FLOTTE]
RENDS :FLOTTE
FIN
```

```
POUR FAIRE.TIRER
SI :FLOTTE = ( ) (STOP)
TAPE PH (VOTRE TIR (1A8 1A8) ?) "
DONNE "TIR LL
SI MEMBRE? :TIR :FLOTTE (COULE) (DANS.L'EAU)
FAIRE.TIRER
FIN

POUR COULE
EC "COULE
LOCALISATION :TIR
FCC 1 DESSIN
SUPPRIME :TIR
FIN

POUR DANS.L'EAU
EC "PLOUF
LOCALISATION :TIR
FCC 2 DESSIN
FIN

POUR LOCALISATION :POS
LC
FPOS PH 10 × PREM :POS 10 × DER :POS
FIN

POUR DESSIN
REPETE 4 (BC AV 2 TD 90)
FIN

POUR SUPPRIME :POS
DONNE "FLOTTE SUPP :POS :FLOTTE
FIN

POUR SUPP :M :LST
SI :LST = ( ) (RENDS ( ))
SI :M = PREM :LST (RENDS SUPP :M SP :LST) (RENDS MP PREM :LST
SUPP :M SP :LST)
FIN

POUR CHOIX.NAVIRES :N
SI :N = 0 (RENDS ( ))
RENDS MD PH 1 + HASARD 8 1 + HASARD 8 CHOIX.NAVIRES :N - 1
FIN
```

JEU D'AVENTURE

Utilisation des listes de propriétés.

Vos talents de programmeur LOGO ont conduit à la démence l'ordinateur central de votre planète d'origine. Le tribunal galactique vous a exilé sur une lointaine planète !
Votre réintégration ne sera possible que si vous ramenez certains documents. Ceux-ci sont disséminés dans toute la galaxie. Il vous faudra parfois les troquer.
Votre périple ne fait que commencer : vous allez devoir vagabonder longuement de pays en pays pour amasser les précieux documents ainsi que divers objets, indispensables pour franchir certaines étapes.

comment faire ?

Chaque pays a plusieurs « propriétés » que nous devons stocker :
- Les objets qui s'y trouvent.
- Les conditions d'accès (objets à présenter).
- Les planètes directement accessibles, (planètes voisines).

Le nom des variables utilisées est construit à partir (voir le lot d'utilitaires « liste de propriétés ») :
- du nom de la planète,
- du nom de la propriété (OBJETS, ACCÈS, VOISINS).

Par exemple, la planète GALACTICA est caractérisée par LES VARIABLES :
- MOT "GALACTICA "OBJETS soit GALACTICAOBJETS
- MOT "GALACTICA "ACCÈS soit GALATICAACCÈS
- MOT "GALACTICA "VOISINS soit GALACTICAVOISINS.

A chaque étape, vous avez le choix entre trois actions : ALLER, PRENDRE, POSER. La première doit être suivie d'un nom de pays, les deux autres de noms d'objets.

NOUVEAU.JEU répartit au hasard les objets dans les pays. Cela vous permet de jouer à chaque fois, sur un nouveau circuit.

PRÉPARE.JEU vous permet de modifier les pays, leurs conditions d'accès et les objets mis en jeu. Vous pouvez ainsi redéfinir complètement un nouveau cadre pour le jeu.

Ces cinq procédures seront appelées directement depuis le clavier.
N'oubliez pas de charger le lot d'utilitaires « listes de propriétés » avant d'essayer le programme ou alors, retapez manuellement les procédures PUT, GET, PROP, DEL.

Pour commencer à jouer, taper NOUVEAU.JEU. Après quelques secondes, vous verrez apparaître un résumé de votre situation : ce que vous possédez, ce que vous pouvez prendre là où vous vous trouvez et les pays qui vous sont accessibles.
Décidez par exemple de prendre une barque (PRENDRE "BARQUE) ou d'aller au royaume interdit (ALLER "ROYAUME.INTERDIT).
Votre but est bien sûr de rejoindre BASE avec les objets qui vous y sont demandés.
Bonne chance ! ! !

à vous de jouer...

Modifiez le programme pour que les droits d'entrée ne soit plus des objets mais des épreuves que vous devrez subir. Pour cela, dans la propriété ACCÈS de chaque pays, vous devrez mettre le nom d'une procédure de jeu (qui pourra être chargée depuis la disquette). Si vous gagnez, la procédure retournera VRAI et vous aurez accès au pays demandé.

programmer : JEU D'AVENTURE

```
POUR PREPARE.JEU
VT TAPE (QUELS SONT LES PAYS ?) DONNE "PAYS PH LL "BASE
TAPE (QUELS SONT LES OBJETS ?) DONNE "OBJETS LL
DÉFINIR "VOISINS :PAYS
DÉFINIR "ACCÈS :PAYS
FIN

POUR NOUVEAU.JEU
VIDER :PAYS
DISPERSER :OBJETS
DONNE "POSITION CHOISI :PAYS
DONNE "OBJ. POSSEDES ( )
DONNE "NB.DEP 0
AFFICHE.ETAT
FIN
```

- Enlève les objets du jeu précédent
- Remet les objets au hasard dans les pays
- Le joueur est dans un pays au hasard
- Il ne possède rien
- Il n'a encore effectué aucun déplacement.

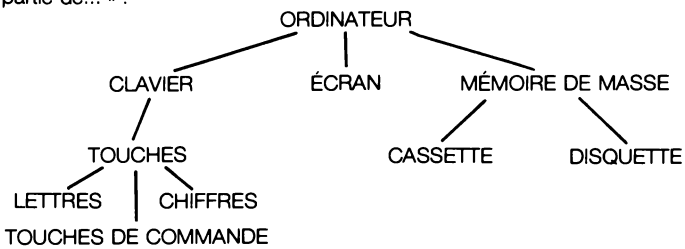
```
POUR ALLER :DEST
SI NON MEMBRE? :DEST :PAYS (EC "INCONNU STOP)
SI NON MEMBRE? :DEST GET "VOISINS :POSITION (EC (TROP LOIN)
STOP)
SI NON INCL GET "ACCES :DEST :OBJ.POSSEDES (EC (NE VENEZ PAS
LES MAINS VIDES) STOP)
DONNE "POSITION :DEST
DONNE "NB.DEP :NB.DEP + 1
SI :POSITION = "BASE (FIN.DU.JEU) (AFFICHE.ETAT)
FIN
```

```
POUR PRENDRE :OBJ
SI LISTE? :OBJ (EC (UNE CHOSE A LA FOIS S.V.P.) STOP)
SI NON MEMBRE? :OBJ GET "OBJETS :POSITION (EC (VOUS AVEZ LE
BRAS LONG) STOP)
SI 4 < COMPTE :OBJ.POSSEDES (EC (VOUS ETES DEJA TROP CHAR
GES) STOP)
DONNE "OBJ.POSSEDES MD :OBJ :OBJ.POSSEDES
PUT "OBJETS :POSITION SUPP :OBJ GET "OBJETS :POSITION
AFFICHE.ETAT
FIN

POUR POSER :OBJ
SI LISTE? :OBJ (EC (UNE CHOSE A LA FOIS S.V.P.) STOP)
SI NON MEMBRE? :OBJ :OBJ.POSSEDES (EC "HUMM ! ! STOP)
DONNE "OBJ. POSSEDES SUPP :OBJ :OBJ. POSSEDES
PUT "OBJETS :POSITION MD :OBJ GET "OBJETS :POSITION
AFFICHE.ETAT
FIN

POUR AFFICHE.ETAT
VT TAPE (VOUS POUVEZ PRENDRE)
SI VIDE? GET "OBJETS :POSITION (EC "RIEN) (EC GET "OBJETS :PO
SITION)
EC " TAPE (VOUS POSSEDEZ :)
SI VIDE? :OBJ.POSSEDES (EC "RIEN) (EC :OBJ.POSSEDES)
EC (VOUS POUVEZ ALLER EN :)
AFFICHE GET "VOISINS :POSITION
EC " EC "
FIN

POUR AFFICHE :PAYS
SI :PAYS = ( ) (STOP)
EC PREM :PAYS
EC PH (VOUS Y TROUVEREZ :) GET "OBJETS PREM :PAYS
EC PH (ON VOUS DEMANDERA :) GET "ACCES PREM :PAYS
AFFICHE SP :PAYS
FIN

POUR DEFINIR :PROP :PAYS
SI :PAYS = ( ) (STOP)
TAPE PH :PROP PH "DE PH DER :PAYS PH "?"
PUT :PROP DER :PAYS LL
DEFINIR :PROP SD :PAYS
FIN
```

```
POUR DISPERSER :OBJETS
SI :OBJETS = ( ) (STOP)
DONNE "PAYS.CHOISI CHOISI :PAYS
SI MEMBRE? PREM :OBJETS GET "ACCES :PAYS.CHOISI (DISPER
SER :OBJETS STOP)
DEPOSE PREM :OBJETS : PAYS. CHOISI
DISPERSER SP :OBJETS
FIN

POUR DEPOSE :OBJ :PAYS
PUT "OBJETS :PAYS MD :OBJ GET "OBJETS :PAYS

POUR CHOISI :LST
RENDS ITEM 1 + HASARD SP :COMPTE :LST :LST
FIN

POUR VIDER :PAYS
SI :PAYS = ( ) (STOP)
PUT "OBJETS PREM :PAYS ( )
VIDER SP :PAYS
FIN

POUR FIN.DU.JEU
VT EC PH PH (IL VOUS AURA FALLU) :NB.DEP "DEPLACEMENTS
FIN

POUR SUPP :MOT :LST
SI :LST = ( ) (RENDS ( ))
SI :MOT = PREM :LST (RENDS SUPP :MOT SP :LST)
RENDS MP PREM :LST SUPP :MOT SP :LST
FIN

POUR INCL :L1 :L2
SI :L1 = ( ) (RENDS VRAI)
SI MEMBRE? PREM :L1 :L2 (RENDS INCL SP :L1 :L2) (RENDS FAUX)
FIN
```

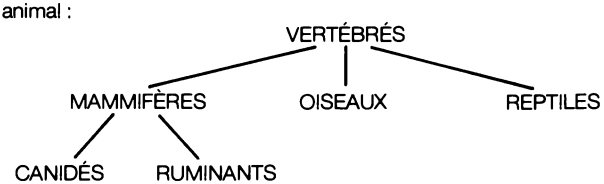
LOGOMONDE

Listes.
L'énoncé de ce problème est simple mais sa solution risque d'être un peu plus longue : écrire un programme qui soit capable de gérer un arbre de connaissances. Voici par exemple, un arbre qui pourrait s'appeler « fait partie de... » :



Chacun des mots inscrits sur cet arbre s'appelle un nœud, les traits qui les relient sont des branches. Vous remarquerez que, si en partant d'un nœud quelconque, on remonte l'arbre, la relation « fait partie de... » s'applique toujours.
Par exemple : TOUCHES fait partie de CLAVIER
CLAVIER fait partie de ORDINATEUR

Les applications de ce genre de programmes sont innombrables. Par exemple, on peut en faire un outil de gestion pour la classification du règne animal :



Naturellement, l'arbre complet est beaucoup plus grand mais ce n'est pas un problème car LOGO possède un énorme avantage : il n'est pas nécessaire de dimensionner. La même variable qui, il y a quelques instants, contenait un simple chiffre, peut maintenant recevoir une liste de plus de cent mots.
La structure et les instructions d'un tel programme doivent être totalement indépendantes des données manipulées. Autrement dit, on pourra gérer aussi bien la classification du règne animal que la hiérarchie des employés d'une société.

comment faire ?

Il y a bien sûr plusieurs solutions. Celle que nous avons retenue utilise le stockage, à la file indienne, de toutes les branches de l'arbre. Une branche étant elle même une liste de deux éléments : le fils et le père.
Tout ceci sera mémorisé dans une variable appelée UNIVERS.

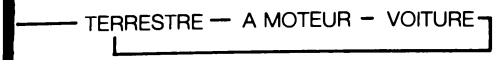
Des procédures (accessibles directement depuis le clavier) permettent d'ajouter ou de modifier des branches et de questionner l'ordinateur sur les connaissances qu'il a déjà. Ces procédures sont :
JE.DIS fils père : ajoute une branche.
QUI objet : écrit les fils de objet.
QUOI objet : écrit le père de objet.
EXPLIQUE objet : écrit toutes les branches entre objet et la racine de l'arbre.

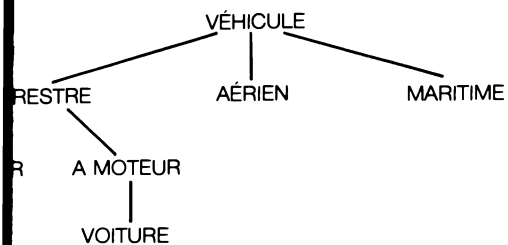
La fonction EST a été elle aussi définie. Elle n'exécute aucune action mais elle permet de poser les questions à l'ordinateur dans un langage proche du français :
QUI EST "POULE est équivalent à QUI "POULE
JE.DIS "CANARD EST "OISEAU est équivalent à JE.DIS "CANARD "OISEAU
C'est une fonction tout à fait neutre : elle retourne son paramètre sans aucune modification.

le programme

Il n'y a pas à proprement parler de procédure principale puisque, en fonction de ce que l'on veut faire (mémoriser de nouvelles branches ou consulter les anciennes), on appelle directement depuis le clavier, les procédures voulues.
L'une d'elle est cependant un peu à part :LOGOMONDE qui initialise le système. Il faut la lancer avant toute autre action. Son rôle est d'initialiser la variable UNIVERS (qui contiendra par la suite, toutes les branches), avec la racine de l'arbre :MONDE. La racine est représentée par une branche fictive qui relie MONDE à liste vide.
La fonction FILS :OBJ recherche la branche qui a comme dernier élément (la place du père) :OBJ et retourne le premier élément (donc le fils) de cette liste.
La fonction PÈRE :OBJ, est exactement symétrique. Quant à PARENTS :OBJ, elle retourne une liste de tous les parents de :OBJ en remontant jusqu'à la racine.
FILS, PÈRE et PARENTS sont les trois outils qui permettent d'explorer l'arbre.
JE.DIS et SUPPLIEN sont utilisés pour modifier l'arbre.
Examinons JE.DIS. La première ligne vérifie que le fils que l'on désire créer ne fait pas déjà partie des parents du père qu'on lui attribue. Si c'était le cas, on aurait un bouclage qui ne manquerait pas d'être source d'ennuis par la suite.

mple, si on voulait ajouter à l'arbre suivant, la branche de VOITURE, on créerait la boucle suivante :





e fait que cette situation est illogique, il devient impossible bre (cela n'en est plus un d'ailleurs) sans se « perdre » dans

ne de JE.DIS supprime uné éventuelle relation antérieure même fils. En effet, n'importe quel nœud de l'arbre ne doit père.
ir aussi bien à modifier une branche de l'arbre qu'à en créer

ccupe de la suppression d'une branche lorsque c'est

NCÊTRE sont chargés d'afficher toute la parenté d'un objet.

de jouer...

est un excellent exemple de ce que l'on peut faire avec les listes et la structure fortement récursive du langage se èrement bien à ce genre de travail.
nous avons adoptée ici (stockage « en vrac » des branches) nent la meilleure. Une autre consisterait à considérer chaque ne variable et à stocker à l'intérieur la liste de ses fils.
sayer de programmer cette deuxième version.

mmer : LOGOMONDE

ONDE

RS ((MONDE ()))

programmer : LOGOMONDE

```
POUR LOGOMONDE
VT
DONNE "UNIVERS ((MONDE ( )))
FIN

POUR FILS :OBJ :MONDE
SI EGAL? :MONDE ( ) (RENDS ( ))
SI NON EGAL? DER PREM :MONDE :OBJ (RENDS FILS :OBJ SP :MONDE)
RENDS PH PREM PREM :MONDE FILS :OBJ SP :MONDE
FIN

POUR PERE :OBJ :MONDE
SI EGAL? :MONDE ( ) (RENDS ( ))
SI EGAL? PREM PREM :MONDE :OBJ (RENDS DER PREM :MONDE)
RENDS PERE :OBJ SP :MONDE
FIN

POUR PARENTS :OBJ :MONDE
SI EGAL? PERE :OBJ :MONDE ( ) (RENDS ( ))
RENDS PH PERE :OBJ :MONDE PARENTS PERE :OBJ :MONDE
:MONDE
FIN

POUR JE.DIS :OBJ :PAR
SI MEMBRE? :OBJ :PARENTS :PAR :UNIVERS (EC "IMPOSSIBLE STOP)
DONNE "UNIVERS SUPPLIEN :OBJ :UNIVERS
DONNE "UNIVERS MP PH :OBJ :PAR :UNIVERS
FIN

POUR SUPPLIEN :OBJ :MONDE
SI EGAL? :MONDE ( ) (RENDS ( ))
SI EGAL? PREM PREM :MONDE :OBJ (RENDS SUPPLIEN :OBJ SP :MONDE)
RENDS MP PREM :MONDE SUPPLIEN :OBJ SP :MONDE
FIN

POUR QUI :OBJ
EC FILS :OBJ :UNIVERS
FIN

POUR QUOI :OBJ
EC PERE :OBJ :UNIVERS
FIN
```

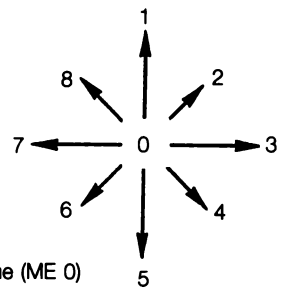
```
POUR EST :OBJ
RENDS :OBJ
FIN

POUR EXPLIQUE :OBJ
EC PH PH PH :OBJ "EST PERE :OBJ :UNIVERS ANCETRE PERE :OBJ
:UNIVERS :UNIVERS
FIN

POUR ANCETRE :OBJ :MONDE
SI EGAL? PERE :OBJ :MONDE ( ) (RENDS( ))
RENDS PH PH (QUI EST) PERE :OBJ :MONDE ANCETRE PERE :OBJ
:MONDE :MONDE
FIN
```

TELECRAN

Guidez la tortue du bout des doigts à l'aide de la manette de jeu... et transformez ainsi votre ordinateur en télécran.

comment faire ?

La fonction MANETTE nous renseigne sur la position des manettes :



Exemples :

EC MANETTE 0
3
signifie que la manette est à droite

SI 1 = MANETTE 0...
permet de tester facilement si la manette 0 est poussée vers le haut.

La procédure TELECRAN :
- Consacre l'écran tout entier au graphique (ME 0)
- Fait appel à la procédure TRACE.

La procédure TRACE :
- Si le levier n'est pas en position centrale (repérée par 0)
elle oriente la tortue en fonction de la position de la manette.
elle la fait avancer de 2 pas.
- Si le bouton est enfoncé (BOUTON?) :
elle lève le crayon
sinon, elle le baisse.

à vous de jouer...

Ecrivez un programme qui vous permette de changer à tout instant la couleur de votre dessin par simple pression sur la touche numérique correspondant à la couleur désirée.
Ajoutez à votre télécran la possibilité de dessiner en trois couleurs.

programmer : TELECRAN

```
POUR TELECRAN
ME 1
TRACE
FIN

POUR TRACE
SI 0 < MANETTE 0 [FCAP 45 — 45 × MANETTE 0 AV 2]
SI BOUTON? 0 [LC] [BC]
TRACE
FIN
```

MUSILOGO

Transformez votre manette de jeu en générateur sonore : chacune des huit positions possibles correspondra à une note de la gamme.

comment faire ?

La primitive JOUE est capable d'interpréter une suite de notes donnée sous la forme d'un mot. Exemple : " DOREFAPAFA

Les notes sont DO, RÉ, MI, FA, SO, LA, SI et PA pour un silence (pause).

La procédure d'entrée MUSILOGO :

- Initialise la variable " NOTES.
- Appelle la procédure JOUER.

JOUER retrouve la note demandée dans la liste :NOTES et la joue.

à vous de jouer ?

Faites participer la tortue afin qu'elle évolue en fonction de la hauteur des notes jouées. En partant de la gauche de l'écran, elle rejoindra le bord droit en montant et descendant en même temps que la mélodie. Ainsi, elle tracera une courbe, image de votre composition.

programmer : MUSILOGO

```
POUR MUSILOGO
DONNE " NOTES (PA DO RE MI FA SO LA SI DO)
JOUER 1 + MANETTE 0
FIN

POUR JOUER :NUMERO.NOTE
JOUE ITEM :NUMERO.NOTE :NOTES
JOUER 1 + MANETTE 0
FIN
```

CRAYON OPTIQUE
ARDOISE MAGIQUE

Tout le monde sait ce qu'est une ardoise magique. Grâce au crayon optique, LOGO permet de transformer l'ordinateur en ardoise magique. Vous pouvez alors écrire sur l'écran comme sur une feuille de papier. Cela peut être tout simplement un pense-bête électronique ou, pour les dessinateurs, un outil de création graphique.

comment faire ?

Dans le principe, c'est très simple : il faut allumer le point sur lequel est appuyé le crayon optique. Cette opération devra se répéter indéfiniment afin de pouvoir tracer un trait qui est une suite de points.
La position du crayon optique est donnée par POSOPT. Quant à la procédure qui permet d'allumer un point, vous la connaissez déjà, c'est FPOS.
Sachant cela, vous pouvez écrire le programme mais, attention, la fonction POSOPT ne rend un résultat correct que lorsque le crayon optique est appuyé. Il faut donc utiliser aussi la fonction CONTACT ?
La procédure d'entrée est ARDOISE.MAGIQUE. Son premier travail est de vider l'écran puis, de cacher la tortue. Ceci afin d'avoir une surface entièrement « blanche » pour dessiner.
Ensuite, elle fait appel à la procédure DESSIN qui s'occupe, comme son nom l'indique, du dessin. Son rôle est simplement d'allumer le point qui est indiqué par le crayon optique (POSOPT) mais seulement dans le cas où ce dernier est en CONTACT avec l'écran. Ensuite, il faut recommencer car le dessinateur peut avoir bougé son crayon et il y a, dans ce cas, un autre point à allumer. DESSIN répète cette action indéfiniment.

à vous de jouer...

Ajoutez une scrutation du clavier : quand le dessinateur appuie sur une touche numérique, la couleur du tracé change.
Trouvez d'autres fonctions qui pourraient être commandées par l'appui d'une touche : vider l'écran, faire apparaître la tortue, la cacher, la remettre au centre, etc.

programmer : ARDOISE MAGIQUE

```
POUR ARDOISE.MAGIQUE
VE CT
DESSIN
FIN
```

```
POUR DESSIN
SI CONTACT? [FPOS POSOPT BC]
[LC]
DESSIN
FIN
```

CHASSE TORTUE

Habituellement, les tortues ne sont pas réputées pour leur rapidité. Celle de LOGO échappe cependant à cette règle. Nous allons l'utiliser pour ce jeu. Pour gagner, il faudra poser le crayon optique sur la tortue mais bien sûr, celle-ci se déplace sur l'écran de façon imprévisible.

comment faire ?

Naturellement, la première chose à faire sera de préparer l'écran. Ensuite, il faudra mettre la tortue en mouvement. Mais, pour être tout à fait imprévisible, ce mouvement devra faire appel à hasard. Enfin, entre chaque déplacement de la tortue, il faudra regarder si le crayon optique est à la même position, auquel cas le joueur a réussi à viser la tortue et a donc gagné.

La procédure d'entrée est CHASSE.TORTUE. Ce sera donc ce mot que vous taperez au clavier pour lancer le programme. CHASSE.TORTUE efface l'écran puis lève le crayon. Ainsi la tortue ne laissera pas de trace derrière elle (ce qui se traduirait, à la longue, par un énorme gribouillis).

Enfin, on fait appel à la procédure ANIME.TORTUE :

La tortue est, dans un premier temps, positionnée au hasard sur l'écran. Ensuite, à plusieurs reprises pour faciliter le jeu, le programme compare la position du crayon optique et celle de la tortue. Si elles sont égales, le joueur a gagné, on arrête le jeu (STOP). Dans le cas contraire, on recommence la même suite d'actions.

Vous vous rendrez très vite compte qu'il est très difficile de gagner à ce jeu. Aussi, il est peut-être intéressant de le simplifier. Par exemple, on peut décréter que le joueur gagne dès qu'il place son crayon à moins de 10 unités de la tortue.

A vous de modifier le programme. Le mieux est de remplacer la suite EGAL ? POS POSOPT par PROCHE ? Cette fonction utilise DISTANCE (déjà vue dans CHAUD ou FROID).

à vous de jouer...

Faites en sorte que le jeu soit progressif : chaque fois que le joueur touche la tortue, sa vitesse augmente.

Pour éviter de commencer toujours par la vitesse la plus lente, demander le niveau de jeu souhaité.

Prévoyez un arrêt de jeu : soit une durée déterminée, soit lorsque le joueur n'a pas touché la cible dix fois de suite.

programmer : CHASSE TORTUE

```
POUR CHASSE.TORTUE
ME 1 LC
ANIME.TORTUE
VT TAPE "BRAVO
FIN

POUR ANIME.TORTUE
FPOS PH 100 - HASARD 200 60 - HASARD 120
REPETE 3 (SI EGAL? POS POSOPT (STOP))
ANIME.TORTUE
FIN

POUR PROCHE? :P1 :P2
SI OU VIDE? :P1 VIDE? :P2 (RENDS FAUX)
RENDS ET PLG? 10 ABS DIFF PREM :P1 PREM :P2 PLG? 10 ABS DIFF
DER :P1  DER :P2
FIN

POUR ABS :N
SI :N < 0 (RENDS - :N) (RENDS :N)
FIN
```

COMMENT INTERPRÉTER LES MESSAGES LOGO ?

Comme d'habitude, LOGO se distingue des autres langages informatiques. En effet, ceux-ci affichent généralement un code d'erreur laissant à l'utilisateur, le soin de se reporter au lexique. LOGO affiche ses messages en clair. Peu nombreux (moins de 20) ils sont cependant très efficaces.

que faire de...

Surplus de données, par exemple : LC 100

QUE FAIRE DE 100

Ou un résultat a été renvoyé par une fonction alors que rien n'était prévu pour le recevoir. Exemples : SOMME 3 2

QUE FAIRE DE 5

Il aurait fallu écrire :

ECRIS SOMME 3 2
ou
DONNE "VAR SOMME 3 2

- Le résultat de SOMME est récupéré par ECRIS qui l'affiche.
- Le résultat est récupéré par DONNE qui le stocke dans VAR.

pas assez de données pour...

Il manque des paramètres à la procédure citée. Exemples :

ECRIS SOMME 3
PAS ASSEZ DE
DONNÉES POUR SOMME

- SOMME demande normalement 2 arguments.

ECRIS PH PH "A "B
PAS ASSEZ DE
DONNÉES POUR PH

- Le deuxième PH prend "A et "B comme arguments et renvoie la liste <A B> au premier PH mais cela ne fait qu'un seul argument. Il en manque donc un.

XXX 24
PAS ASSEZ
DE DONNÉES POUR XXX

- La procédure XXX a été définie par vous avec plus d'un argument.

PAS ASSEZ DE DONNÉES POUR ( )

Cette forme un peu spéciale du message PAS ASSEZ DE DONNÉES POUR... peut signifier que :

- vous avez fermé une parenthèse de trop,
- il manque un argument à la dernière des procédures entre parenthèses,
- une parenthèse est mal placée. Ex : (EC " BONJOUR)

PAS ASSEZ DE DONNÉES POUR ( )

Comment faire...

- Vous avez oublié de définir la procédure citée.
- Vous l'avez effacée depuis.
- Vous avez oublié de faire précéder le nom d'une variable par ":". LOGO essaye alors de l'interpréter comme une procédure.
- Vous avez oublié un guillemet avant un mot LOGO (ce cas ressemble au précédent).
- Vous avez fait une faute de frappe (cas le plus fréquent) :

ECRIT "A au lieu de ECRIS "A
COMMENT FAIRE ÉCRIT

- Il se peut aussi que vous essayez de travailler sur un nombre trop grand

Ex : EC 1E120
1E120
EC 1E121
COMMENT FAIRE 1E121

- LOGO ne connaît pas les nombres au-dessus de 1E120 et en dessous de 1E-120

pas de chose donnée à...

Vous essayez d'extraire le contenu d'un mot qui n'est pas une variable parce qu'il ne contient rien.

- Vous avez oublié d'initialiser cette variable (pas de DONNE).
- Vous avez mal orthographié le nom de la variable.

... existe déjà

Vous essayez de redéfinir une procédure que vous avez déjà définie.
Si vous vouliez volontairement changer la définition de cette procédure, utilisez plutôt l'éditeur, sinon trouvez-lui un autre nom. Il se peut aussi que la définition précédente ne vous intéresse plus : effacez-la :
EF "NOM
Il se peut aussi que vous essayez de définir une fonction qui a le même nom qu'une primitive.

Ex : POUR FIN
FIN EXISTE DÉJÀ

plus de place

La place mémoire dont dispose votre ordinateur est pleine. Ceci arrive fréquemment lorsque le programme « boucle » sans fin, par exemple à cause d'un appel résursif qui ne s'arrête pas.
La primitive RECYCLE permet de libérer l'espace mémoire non utilisé par vos procédures.
Il se peut aussi que votre application demande réellement toute la mémoire même si le texte du programme n'est pas très long. Dans ce cas, il n'y a pas grand espoir mais vous pouvez quand même essayer de :

- Diminuer le nombre de noms de variables. Si votre programme est bien construit, la plupart de ses variables sont internes à une procédure (locales) : vous pouvez sans problème utiliser le même nom dans une autre procédure.
- Restructurer votre programme pour minimiser le nombre de procédures récursives ayant comme paramètres des listes trop longues.

... n'aime pas...

Le type du paramètre ne correspond pas à celui qu'attendait la procédure.
Exemples :

```
EC SOMME "A "B
SOMME N'AIME PAS "A COMME ENTRÉE

FCAP 410
FCAP N'AIME PAS 410
```

- SOMME attend des nombres. "A et "B sont des mots.
- FCAP attend un nombre inférieur ou égal à 360.

... non trouvé

Exemple :

```
RAMENE "MACHIN
MACHIN NON TROUVÉ
```

Le fichier MACHIN n'est pas sur la disquette ou sur la cassette.

nombre trop grand

Le résultat d'un calcul dépasse la capacité de la machine : les limites sont 1E120 et 1E-120.

UTILISATION DE LA CASSETTE

charger un programme

.EFT
- (nettoie la mémoire)

RAMENE "XXX
- (transfère le fichier XXX de la cassette à la mémoire de l'ordinateur).

essayer le programme (après le chargement)

Une procédure (EXPLICATION) a été rajoutée à tous les programmes. Elle affiche quelques commentaires sur l'utilisation du programme et donne le nom de la procédure d'entrée.
Vous pouvez taper directement ce dernier si vous le connaissez.
Les "lots d'utilitaires" ne sont pas des programmes autonomes : il s'agit d'ensembles de procédures qui doivent être intégrés dans un programme tout comme des primitives.
En résumé, pour faire tourner un progamme, tapez...

EXPLICATION (affiche des commentaires et des conseils).
ou le nom de la procédure d'entrée (lance le programme).

obtenir la liste de toutes les procédures d'un programme : IMTS

obtenir le listing d'une procédure : IM "nom de la procédure.

modifier une procédure : ED (nom de la procédure)

arrêter un programme

Certains programmes mettent très longtemps pour se terminer d'eux-mêmes (essayez HANOI avec 9 disques). Vous pouvez les interrompre :
Maintenir enfoncée la touche <CNT>
et appuyez sur <C>

si un message logo apparaît...

LOGO envoie un message par exemple, si vous avez mal répondu à une question du programme.

Exemples de messages :
PREMIER N'AIME PAS ( ) COMME ENTREE DANS...
PLUS DE PLACE
etc

Essayez de définir pourquoi LOGO a été obligé de s'arrêter (voir tableau ci-après) et relancez le programme (tapez le nom de la procédure d'entrée)

PLUS DE PLACE
— Vous avez demandé une opération avec un chiffre trop grand
Ex : EC PREMIER 1013
— Vous n'avez pas tapé .EFT avant de charger le programme.

PREMIER N'AIME PAS ( )
— Vous avez tapé la touche <ENTREE> en premier alors que vous auriez dû taper par exemple, une lettre suivie de <ENTREE>

Correspondance entre titres de projets et noms de fichier sur cassette

NOM DU PROJET	NOM DU FICHIER
Tortue, dessine moi un...	DESSINS
La tortue au pied du mur	MUR
Polygones	POLYGONE
Alea jacta est (jeu de dés)	ALEA
Le devin (jeu)	DEVIN
LOGO, mage transcendental	MAGE
Soustraire pour diviser	DIVISION
Des arbres, si j'en connais	ARBRES
Apprentissage (mes lacunes m'enrichissent)	APPRENDS
Ne perdez pas la boule	BOULE ✗
PGCD PPCM	PGCDPPCM
Traducteur de chiffres romains en chiffres arabes	ROMAINS
Lots d'utilitaires :	
Extensions graphiques	EXTGRAPH
Dessin en 3 dimensions (perspective)	EXT3D
Extensions numériques	EXTNUM
Traitement des listes	EXTLST
Nombres premiers	PREMIERS
Chaud ou froid (jeu avec la tortue)	CH-FROID
Lot d'utilitaires : les listes de propriétés	PROP
Traducteur anglais français	TRADUCT
Mastermind	MASTER
Machine à tout faire	MACHINE
Répertoire téléphonique	REPERTOI
Tours de Hanoi	
première partie : dessin des mouvements du joueur	HANOI 1
deuxième partie : résolution du problème par LOGO	HANOI 2
Fractal (dessin récursif)	FRACTAL
Lot d'utilitaires : les ensembles	ENSEMBLE
Le mot le plus long (application des ensembles)	MOT-LONG
Bataille navale	BATNAV
Jeu d'aventure	AVENTURE
Logomonde (gestion d'un arbre de connaissances)	LOGOMOND
Télécran (utilisation d'une manette de jeu)	TELECRAN