NOTIONS DE PROGRAMMATION AVANCEE.

Les informations qui suivent dans ce paragraphe sont données de façon formelle pour les enseignants. Elles ne doivent en aucun cas être évoquées en présence des élèves des cycles 2 et 3 (Il ne s'agit donc pas encore une fois de faire de l'informatique pour elle-même) sauf de la manière la plus intuitive et la plus pratique possible dans le cadre de la mise en œuvre des compétences scolaires. L'enseignant(e) trouvera ci-dessous des connaissances qui faciliteront son travail d'aide pour les élèves qui, malgré les dispositifs de localisation des erreurs que Logoplus lui propose ainsi que les messages d'erreur reçus, n'arriveraient pas à faire travailler leur texte.

1) Les variables.

1 a) Les noms de variable.

Les noms de variable doivent être utilisés soit après la primitive **DONNE** , soit après la primitive **PARTAGE**. Dans les deux cas, ces noms doivent être précédés du symbole ". Lorsqu'ils sont utilisés par la suite, ils doivent être précédés par le symbole : suivi ou non d'un espace (facultatif).

Remarque: les noms de variable ne doivent pas comporter au dedans de signes opératoires ou de signes particuliers (: ").

1 b) Les types de variable en LOGO.

Les types indiquent la nature précise de ce que contient le nom d'une chose (variable).Lors de la mise en œuvre d'un programme, on peut être amené à manipuler des nombres, des lettres, des listes de mots (phrases) : ce sont les types que peuvent prendre une variable.

La procédure suivante illustre ce que peuvent être les différents types de variable :

```
POUR exemple
DONNE "nom 5
// ici "nom représente un nombre (type numérique) //
```

```
// Les points de suspension représentent la suite du déroulement de la
procédure. //
DONNE "nom [ a e i o u ]
  // ici "nom représente une liste de caractères (type liste) //
 // Les points de suspension représentent la suite du déroulement de la
procédure. //
 DONNE "nom JAUNE
 // ici "nom représente une couleur (type couleur) //
// Les points de suspension représentent la suite du déroulement de la
procédure, qui a peu d'importance dans ce paragraphe. //
FIN
```

Au cours du déroulement de la procédure exemple, la chose "nom a changé successivement trois fois de type (numérique, liste, couleur). Même si cela est déconseillé, il est possible de le faire en LOGO, sous réserve de connaître précisément le type de chaque chose (variable) dans toutes les parties du programme lors de son déroulement. Comme il n'est pas obligé d'indiquer au préalable le type d'une variable en LOGO, cela crée parfois des messages d'avertissement lors de la compilation du texte-programme. Il faudra simplement veiller au contenu de chaque variable utilisée lors de son déroulement.

Il est possible de définir le type d'une variable grâce à la primitive DEFINIS (voir le catalogue), ce qui n'empêche nullement de modifier le type de cette variable au cours du déroulement du programme.

En LOGO, il y a cinq types de variable différents:

Type 1 : les mots.

Précédés et suivis de guillemets. Ex: DONNE "prénom "François

Type 2 : les listes.

Encadrées par les symboles de liste '[' et ']'.

Ex : DONNE "voyelles [a e i o u]

Type 3 : les nombres.

Ils interviennent aussi bien dans les primitives graphiques (AVANCE, RECULE …) que dans les calculs purement algébriques. A l'intérieur des expressions algébriques, les quantités ainsi que les résultats numériques doivent se situer

entre +/- 9,22E18. Tout nombre situé à l'extérieur de ces intervalles occasionnera un message d'erreur "Chose trop grande. ".

Ex: DONNE "valeur 5,6E12

Type 4 : les booléens.

Peuvent être vrais ou faux.

Exemples :

```
DONNE "valeur1 VRAI
DONNE "valeur2 FAUX
DONNE "valeur3 (5=(3+2))
DONNE "valeur4 7=6
```

Remarque : "valeur3 aura pour contenu VRAI, car 5 est bien égal à 3+2.

"valeur4 aura pour contenu FAUX.

Rappel : dans l'instruction DONNE, le nom d'une chose doit être précédé du symbole " (erreur de syntaxe).

Type 5 : les couleurs.

Ex : DONNE "UneCouleur VERTE

Remarque: le type couleur peut prendre l'une des dix-sept valeurs suivantes : VERTE, ROUGE, BLEUE, JAUNE, BLANCHE, NOIRE, FUCHSIA, SARCELLE, TILLEUL, ROSE, ORANGE, MARRON, EAU, GRISE, OLIVE, ARGENT, VIOLETTE.

1 c) Portée d'une variable.

Toutes les variables déclarées à l'intérieur d'une procédure ne sont valables qu'à l'intérieur de cette procédure, sauf si elles figurent dans la liste des variables partagées (voir le mot-clé PARTAGE dans le catalogue) auquel cas ces variables sont autorisées dans tous les procédures situés après ce niveau. La portée d'une variable fixe la zone du programme sur lequel elle est définie.

Exemple :

```
POUR niveau2
DONNE "quantité2" 3 DONNE "quantité3" 5 DONNE "quantité4" 7
```

```
 DONNE "quantité4" : quantité2 + : quantité3 + : quantité4
//
  Cette ligne est rendue possible grâce au mot-clé PARTAGE situé au niveau1. Un
usage de "quantité1" occasionnerait une erreur ici.
//
 FIN
 POUR niveau1
 DONNE "quantité1" 6
//
"quantité1 n'est valable que dans niveau1.  Une tentative pour utiliser "quantité1
dans
 niveau2 provoquera une erreur.
//
 PARTAGE ["quantité2" "quantité3" "quantité4" ]
//
"quantité2" "quantité3" "quantité4" sont autorisées dans toutes les procédures
en aval de niveau1.
//
niveau2
ECRISLIGNE "quantité1"
//
L'usage de "quantité1" est à nouveau possible dans  niveau1.
//
FIN
```

2) Les messages d'avertissement ou d'erreur.

Les messages d'avertissement ou d'erreur apparaissent dans une petite fenêtre prévue à cet effet.

Lorsque la fenêtre de l'éditeur est visible, un simple clic de la souris sur la ligne correspondant au message occasionne la mise en inversion vidéo du mot ou passage erroné dans le texte-programme.

L'élève (ou l'enseignant(e)) peut ainsi connaître presque immédiatement où se situe l'erreur commise et la corriger dans l'éditeur.

a) Les messages d'avertissement.

Les messages d'avertissement correspondent à des zones du texte qui peuvent se révéler être des erreurs lorsque LOGO tente de le faire travailler: une chose (variable) mal définie, un calcul qui peut se révéler impossible, etc… . L'élève est

prévenu : il pourra faire travailler son texte mais un mauvais déroulement du programme peut survenir à cette occasion, entraînant du même coup un résultat erroné ou pas de résultat du tout (arrêt du programme).

Les messages d'avertissement font presque toujours référence au type d'une chose et commencent toujours par le mot "Avertissement". Ils préviennent soit:

- que LOGO ne sait pas quoi faire avec une consigne.

- que le type d'une expression n'est pas approprié.

b) Les messages d'erreur.

Les messages d'erreur commencent toujours par l'expression "Ne comprends pas" . Cela signifie que le texte-programme rédigé par l'élève ne peut pas faire l'objet d'un travail exécutable par LOGO. L'élève devra donc corriger ses erreurs et faire apprendre son texte à LOGO une nouvelle fois.

b.1) Les erreurs déclenchées sous éditeur:

Les erreurs formelles (de syntaxe):

Elles commencent toujours par "Ne comprends pas" ou "LOGO ne comprend pas".

Un coup d'œil sur le catalogue des mots-clés ou sur le passage précis du texte où se situe l'erreur permet d'y remédier. Elles signalent soit:

- que LOGO ne reconnaît pas quelque chose comme un nombre.
- qu'une virgule est mal placée.
- qu'un exposant doit être entier.
- qu'il manque quelque chose.
- qu'il y a trop d'opérateurs dans une expression.
- qu'un nom de chose n'est pas permis.

Les erreurs logiques:

Un coup d'œil sur un texte-programme permet presque toujours de remédier à la plupart de ces erreurs. Si cela ne fait pas disparaître l'erreur, il sera alors nécessaire de revoir la construction du programme.

Elles commencent elles aussi par "Ne comprends pas" ou "LOGO ne comprend pas".

Elles signalent soit:

- qu'il n'y a pas assez d'opérateurs.
- qu' un signe opératoire est mal placé.
- qu'un signe de comparaison est mal placé.
- qu'il y a trop de parenthèses ou de crochets.
- qu'il y a une erreur de parenthèses.
- qu'il y a une expression illogique (ex: la quantité de valeurs données ne correspond pas
  à la quantité de signes opératoires.)
- qu'il manque quelque chose.
- qu'un nom est déjà utilisé par LOGO pour définir une procédure.
- qu'il y a un nom de chose non permis dans la déclaration d'une procédure.
- que le nombre de déclarations de procédures ne correspond pas au nombre de définitions.
- que le nombre de crochets ouvrants n'est pas égal au nombre de crochets fermants.
- qu'il y a trop d'arguments ou au contraire, pas assez.
- qu'une primitive doit être suivie par une ou plusieurs consignes à exécuter.

b.2) Les erreurs déclenchées lors de l'exécution d'un programme:

Les erreurs fréquentes:

suspension utilisateur. (L'utilisateur a cliqué sur le bouton Stop) .
primitive incomplète. (Ex : SOMME 6. LOGO ne sait pas quoi ajouter avec 6. Il aurait fallu écrire par exemple : SOMME 6 2 qui aurait donné 8.)

Attention: Souvent, une erreur de syntaxe peut être occasionnée à l'intérieur de l'appel d'une procédure comportant plusieurs arguments dont l'un au moins est numérique et précédé d'un des signes opératoires + ou -.

Exemple n°1: ECRISLIGNE SOMME 4 -3 // La primitive SOMME nécessite deux arguments numériques. //

Dans cet exemple, LOGO va "apprendre" qu'il faut ôter 3 de 4 après SOMME, c'est à dire qu'il considérera le résultat 1 (4 – 3 = 1) comme le premier argument de cette primitive. Pour comprendre la syntaxe telle que LOGO la fera travailler, on peut réécrire la ligne précédente comme suivant :

ECRISLIGNE SOMME 1 … et plus d'autre argument, d'où l'erreur (primitive incomplète) qui se manifestera ensuite lorsque LOGO essaiera de faire travailler la primitive SOMME .

Solution: placer des parenthèses entre les arguments, ce qui donnera:

ECRISLIGNE SOMME 4 (-3) // L'erreur a disparu. //

Rappel: pour que LOGO affiche un résultat, il ne faut pas oublier de faire précéder l'opération utilisée par la primitive ECRIS ou ECRISLIGNE, sinon LOGO n'affichera rien du tout, ce qui peut parfois faire penser qu'il existe une erreur, mais erreur n'est pas oubli !

Exemple n°2 : TANTQUE (CARDINAL :liste < 6) [// actions à exécuter //]

Cette fois, une erreur sera occasionnée après la primitive CARDINAL , et plus exactement sur le signe de comparaison < , car LOGO va d'abord chercher à évaluer l'argument de cette primitive, c'est-à-dire :liste < 6 , où :liste est une liste d'items et 6, une valeur numérique. Comme ces deux choses ne sont pas comparables, un message d'erreur apparaîtra lorsque LOGO fera travailler votre texte, vous indiquant cette incohérence (erreur de logique).

Solution: Là encore, l'utilisation des parenthèses est souveraine, en indiquant que c'est (CARDINAL :liste) qui doit être inférieur à 6, ce qui se traduit par :

TANTQUE ((CARDINAL :liste) < 6) [// actions à exécuter //]
// L'erreur a disparu. //

chose trop grande. (Ex : AVANCE 600000)
Rappel: A l'intérieur des expressions algébriques, les quantités ainsi que les résultats numériques doivent se situer entre +/- 9,22E18.

choses non comparables. (Ex: 5 < [un deux trois])

chose non booléenne. (Ex : TESTE (8+4) ; 8+4 représente un nombre : il n'est ni vrai ni faux. Il aurait fallu écrire : TESTE ((8+4)=12)

chose non permise. (Ex : TESTE (8<"huit)

division par zéro. (Ex : 5/0 ou QUOTIENT 5 0)

chose non répertoriée. (Appel du contenu d'une chose (variable) inconnue dans la procédure)

suspension système programmée. (Le programme a rencontré la primitive (mot-clé) SUSPENDRE)

valeurs(s) graphique(s) non permises(s). (Ex : RECULE 500000)

nombre d'items incorrects.(Ex : ECRIS ITEM [un deux trois] 4)

système interrompu. (Une erreur est survenue au cours du déroulement du programme.)

système interrompu: pas de chose booléenne à évaluer.

Ex : SI [AVANCE 10 TOURNEDROITE 90]
Une expression booléenne doit suivre la primitive SI.

Il aurait fallu écrire:

SI (:n=5) [AVANCE 10 TOURNEDROITE 90]

nom de chose non permis. (Ex : DONNE "AVANCE 50)

nom de lutin non permis.

Remarque: les noms de lutin ne doivent pas comporter de signes opératoires ou de signes particuliers (: ").

Les erreurs d'exception.

Elles sont précédées du mot SYS (système) et apparaissent souvent lors d'appels récursifs sans critères d'arrêt :

SYS : Il n'y a plus assez de place en mémoire pour effectuer les calculs restants.

SYS : Il n'y a plus assez de place en mémoire pour appeler les procédures restantes.