ASELEC

INFORMATIQUE PEDAGOGIQUE

# LSEG - EDL

## Langage Symbolique d'Enseignement Graphique

THOMSON TO7 TO7-70 MO5

ASELEC Avenue de Paris 78820 Juziers Tél : (3) 475.21.02

# TABLE DES MATIÈRES

# INTRODUCTION

Conçu dès 1970 par l'Ecole Supérieure d'Electricité, à la demande du Ministère de l'Education Nationale, le L.S.E. (Langage Symbolique d'Enseignement) a été, au niveau du logiciel de base, le principal support de l'introduction de l'informatique dans le système éducatif français.

Il a été d'abord implémenté sur les mini-ordinateurs T 1600 et Mitra 15, dans le cadre de l'expérience dite des « 58 lycées », puis sur les micro-ordinateurs, à base de Z 80, de l'opération « 10 000 micros ». On le trouve à nouveau, dans le marché dit « 4e tranche », sur des micros à base de 8088/8086 ou 6809.

Normalisé et enrichi par l'AFNOR, ce langage évolué à syntaxe française mérite qu'on s'y arrête :
— Mise en œuvre commode, avec une analyse syntaxique par ligne.
— Possibilité de définir et d'utiliser de véritables procédures, avec passage de paramètres, par valeur et par adresse, récursivité simple et croisée.
— Gestion autonome des fichiers, rendant le système d'exploitation transparent à l'utilisateur.
— Prise en compte d'objets, d'opérateurs et de fonctions graphiques.

Le L.S.E. que nous allons décrire est celui de la société E.D.L. implémenté sur T07, T07-70 et M05, sous le nom de LSEG-EDL. Il a été édité spécialement pour l'Éducation Nationale et l'UGAP par la société ACT Informatique.
— Il gère complètement les 16 couleurs du T07-70 et du M05 de telle manière qu'un programme écrit sur T07 puisse tourner sans aucune modification sur T07-70 et M05. Un programme écrit sur T07-70 ou M05 et utilisant les couleurs pastel fonctionnera sans modification sur T07 mais remplacera chacune des couleurs pastel par la couleur saturée de même nom.
— L'utilisation en mode Calculateur de Bureau (CALBUR) est très riche : elle permet en particulier l'appel des procédures (qu'elles soient écrites en L.S.E. ou binaires, internes ou externes).
— Un soin tout particulier a été apporté à la transmission des paramètres dans les procédures, y compris les cas particuliers (paramètres synonymes transmis par adresse, par exemple).
— La gestion de la mémoire, lors de l'utilisation de procédures externes, optimise le nombre de chargements des procédures qui sont conservées tant qu'il y a de la

place, même si elles sont inactives.
— L'arithmétique permet de faire des calculs sur des entiers sans aucun arrondi avec 8 chiffres. Les fonctions mathématiques donnent le plus souvent 9 chiffres significatifs exacts ce qui est un maximum, compte tenu de la représentation interne des nombres.
— Enfin, on trouve un certain nombre d'innovations intéressantes : les chaînes formatées, par exemple, ou encore la possibilité de lecture directe d'expressions numériques ou booléennes.

Les pages qui suivent n'ont pas pour ambition d'enseigner la programmation : elles s'adressent en fait (et qu'il nous soit permis de le regretter sans autre commentaire), à des utilisateurs effectifs ou potentiels qui connaissent déjà par ailleurs le L.S.E., et ou un autre langage sur les nano-machines Thomson :
— le Manuel Utilisateur est conçu pour faire acquérir rapidement les notions propres au L.S.E. et ne se veut nullement un document « autodidactique » ;
— le Manuel de Référence et les Annexes, plus classiques, s'attachent néanmoins à mettre en évidence la qualité et les spécificités de l'implémentation.





# MANUEL UTILISATEUR

# CHAPITRE I

## MISE EN ROUTE

# MISE EN ROUTE

## I.1 MISE EN ROUTE SUR TO7, TO7-70 ET MO5

L'unité centrale est munie de sa cartouche LSEG-EDL

Allumer dans l'ordre :
- le téléviseur
- le lecteur - enregistreur de cassettes
- l'unité (ou les unités) de disquettes
- l'unité centrale

Sur TO7 un MENU s'affiche à l'écran. Répondez par 1 ou 2 selon votre choix.
Vous avez choisi 2 : (règlez votre crayon et revenez au MENU)
Vous avez choisi 1 : le MENU disparaît de l'écran.
Le message DATE (JJ, MM, AA)?
s'affiche en haut de l'écran (ce message apparaît directement sur MO5, sans passer par le MENU)

***Exemple :***

Nous sommes le 6 février 1984
Vous tapez : 6 2 84 puis vous validez en appuyant sur la touche « ENTRÉE ».
Vous devez encore taper BO puis appuyer sur la touche « ENTRÉE » : Le système complète et affiche BONJOUR.
Vous venez de taper votre première commande. Son rôle est principalement d'initialiser la zone utilisateur, dans laquelle vous allez travailler.

## I.2 SYNTAXE DES COMMANDES L.S.E.

Il n'est pas possible d'utiliser le L.S.E. sans connaître son langage de commande.
Les commandes L.S.E. s'activent en tapant les deux premiers caractères suivis





d'une validation par la touche ENTREE (symbolisée dans la suite par le signe Ø). Il pourra ensuite vous être demandé des compléments d'information pour certaines des commandes.

***Exemple :***

EXØECUTER A PARTIR DE 1Ø
(les caractères soulignés sont ceux tapés par l'utilisateur)
On s'aperçoit qu'une commande est exécutée lorsque le curseur passe au début de la ligne suivante.
La totalité des commandes est décrite dans la partie « Manuel de Référence » mais nous aurons l'occasion d'en voir un certain nombre au fur et à mesure des exemples.

## I.3 DISQUETTES ET CASSETTES

Le L.S.E. permet d'utiliser indifféremment cassettes et disquettes. Ces unités sont connues du L.S.E. sous le nom de disques, numérotés de la manière suivante :
disque 0 : lecteur-enregistreur de cassettes
disque 1 : unité de disquettes du bas
disques 2 à 4 : autres unités de disquettes numérotées de bas en haut.
A la mise en route, L.S.E. fait le choix du disque par défaut : si il y a une unité de disquettes sous tension c'est le disque 1, sinon c'est le disque 0. (Voir dans le Manuel de Référence la commande DIsque).

## I.4 USAGE DE TOUCHES PARTICULIÈRES

- La touche « ENTRÉE » (Ø) valide tout message tapé.
- La touche « STOP » permet d'interrompre le L.S.E. Ce caractère provoquera l'apparition du message « PRET ».
- La touche « FLÈCHE GAUCHE » permet d'effacer le dernier caractère tapé.
- La touche « RAZ » permet d'effacer l'écran et de positionner le curseur en haut et à gauche de l'écran.
- La touche « m/M » : il s'agit sur TO7 de la touche marquée d'un point jaune, et sur MO5 de la touche jaune. En appuyant sur cette touche « m/M » et en tapant simultanément sur la touche « BARRE D'ESPACEMENT » vous passez en mode

« minuscules » (le voyant min s'allume sur TO7) :
Toutes les lettres que vous taperez alors seront en minuscules. Dans ce mode, vous pourrez obtenir une lettre majuscule en appuyant simultanément sur « m/M » et la lettre désirée.
Vous repasserez en mode majuscule en appuyant à nouveau simultanément sur la touche « m/M » et sur la « barre d'espacement ».
• La touche « ACC » permet d'obtenir 12 caractères qui ne figurent pas tous sur le clavier.
Il suffit pour cela d'appuyer sur la touche « ACC » puis sur l'une des touches situées en haut du clavier.
On obtient dans l'ordre :

| \ | \| | ← | [ | ] | é | è | ù | ç | à | { | } |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | – | + |

Pour vous en souvenir, dessinez ces caractères sur de petites pastilles et collez celles-ci sur le clavier.

Cette même touche doit être utilisée pour obtenir des minuscules avec accent circonflexe ou tréma*. La méthode à suivre pour obtenir ces minuscules est la suivante :
– se mettre en mode minuscules
– appuyer sur la touche « ACC »
– appuyer silmultanément sur :
  – la touche « m/M »
  – la touche « CNT » (pour le tréma seulement)
  – la touche sur laquelle est dessiné l'accent circonflexe.
– appuyer sur la minuscule désirée.

Il est ainsi possible d'obtenir : â ê î ô û et ë ï ü.*
Les erreurs de manipulation se traduisent par des taches uniformes sur l'écran (caractère de code 127).

*(*) Sur TO7 seulement.*





# CHAPITRE II
# DÉCOUVERTE DU L.S.E.

- **UN PROGRAMME SÉQUENTIEL**
- **ÉDITEUR DE LIGNE**
- **EXEMPLE DE MENU**
- **APPRENONS A LIRE**

# DÉCOUVERTE DU LSE

## II-1 UN PROGRAMME SÉQUENTIEL

Pour commencer, voici un petit problème tout simple. Il s'agit de faire un programme reproduisant le dessin ci-dessous.

* * * * *
*
*
* * * * *
*
*
* * * * *

Pour obtenir ce résultat, convenons de traiter les lignes du dessin l'une après l'autre en partant du haut. On peut traduire cela de la manière suivante.

- début de traitement
- on dessine d'abord * * * * *
- ensuite * sur la ligne suivante
- encore *
- à nouveau * * * * *
- on dessine *
- on dessine *
- on dessine * * * * *
- fin de traitement





Ce que réalise le programme suivant :

```
1 * Dessin d'un E
20
30 AFFICHER '* * * * *'
40 AFFICHER '*'
50 AFFICHER '*'
60 AFFICHER '* * * * *'
70 AFFICHER '*'
80 AFFICHER '*'
90 AFFICHER '* * * * *'
100
110 TERMINER
```

Comme tout programme, celui-là comporte :

- Un début : la ligne 1 qui comporte un commentaire (le symbole « * » indique que la suite est un commentaire et ne doit pas être interprétée par le système).
- La partie principale qui développe l'algorithme : ici les lignes 30 à 90.
- Une fin, ici en ligne 110.

Les lignes 20 et 100 ne servent qu'à « aérer » le listing. Ce programme L.S.E. est donc formé de lignes contenant soit des commentaires (ligne 10), soit des instructions (les lignes 30 à 90, ainsi que la ligne 110), soit rien (lignes 20 et 100).
Attention, quand on tape une ligne, on doit taper un espace après le numéro de ligne, sinon on voit apparaître sur l'écran le message :
ERREUR C 7
Un tel message (ERREUR C xx - voir en Annexe la signification des numéros d'erreurs) nous indique que la ligne n'est pas syntaxiquement correcte. Il nous faudra soit la corriger (voir plus loin « Editeur de ligne » soit la retaper.
Ce programme utilise deux instructions : l'instruction AFFICHER qui sert ici à faire apparaître des messages sur l'écran et l'instruction TERMINER. Cette instruction est la dernière qui est exécutée par le programme.
Regardons le titre du paragraphe, il annonce un programme séquentiel ; cela signifie que le déroulement du programme se fera en commençant par la ligne 10, puis la ligne 20, puis la 30 et ainsi de suite jusqu'à la ligne 110 sans en sauter aucune et sans revenir en arrière.
Le fonctionnement de l'instruction AFFICHER est le suivant ; regardons où se trouve le curseur (« _ ») avant l'exécution de cette instruction ; à partir de cette position, le curseur ira au début de la ligne suivante, puis dessinera la suite de caractères représentée entre ' ' (on dit entre apostrophes) et s'arrêtera à droite du dernier caractère dessiné. Il est temps de lancer l'exécution du programme ;

pour cela nous allons taper une commande (ce qui signifie que nous allons donner un ordre au L.S.E.). Il s'agit de la commande EXécuter. Cela se passe en trois temps :
- on tape d'abode EX puis on valide par ENTREE (@)
- le L.S.E. complète en EXECUTER A PARTIR DE et attend un complément
- on tape 1 et on valide

Ce qui donne à l'écran :

```
EX@ECUTER A PARTIR DE 1@

* * * * *
*
*
* * * * *
*
*
* * * * *
TERMINE EN LIGNE 110
```

## II.2 ÉDITEUR DE LIGNE

L'éditeur de ligne permet de corriger une ligne de programme sans avoir à la retaper. Sa syntaxe est la suivante :
/caractères à remplacer/caractères remplaçants
ou encore :
%caractères à remplacer%caractères remplaçants

Les deux formes sont nécessaires car l'utilisation de / ... / ... ne permet pas de remplacer une chaîne contenant le caractère « / » de même que % ... % ne permet pas de remplacer « % ».





*Exemples :*

```
320 AFICHER 'Bonjour'
ERREUR C 3
/F/FF
320 AFFICHER 'Bonjour'
/yu/ou
320 AFFICHER 'Bonjour'

   LI@STER LIGNES 400@
400 C←X*5/Y
%/Y%*z
400 C←X*5*z
LISTER LIGNES 400
400 C←X*5*Z

LISTER LIGNES 320
320 AFFICHER 'Bonjour'

/32/83
830 AFFICHER 'Bonjour'
//5
5830 AFFICHER 'Bonjour'
LI@STER LIGNES *@


.....
320 AFFICHER 'Bonjour'

.....

830 AFFICHER 'Bonjour'

.....

5830 AFFICHER 'Bonjour'

.....
```

Un seul « / » (ou un seul « % ») permet des ajouts en fin de ligne :

```
LISTER LIGNES 320
320 AFFICHER 'Bonjour'

/;ALLER EN 999
320 AFFICHER 'Bonjour';ALLER EN 999
```

## II.3 UN EXEMPLE DE MENU

Soit à réaliser un programme qui dessinerait sur l'écran le menu suivant :

```
M E N U

*****

SALADE NICOISE

*****

ENTRECOTE-FRITES

*****

FROMAGE OU DESSERT

*****
```





On peut réaliser le programme demandé sous la forme suivante :

```
1 * Menu
20
30 AFFICHER [2/,'      M E N U']
40 AFFICHER [2/,'        *****']
50 AFFICHER [2/,'  SALADE NICOISE']
60 AFFICHER [2/,'        *****']
70 AFFICHER [2/,'ENTRECOTE-FRITES']
80 AFFICHER [2/,'        *****']
85 AFFICHER [2/,'FROMAGE OU DESSERT']
90 AFFICHER [2/,'        *****']
100
110 TERMINER
```

Une nouveauté dans ce programme : les « 2/ » contenus dans chacune des instructions AFFICHER. D'une part « / » provoque le passage au début de la ligne suivante, d'autre part, le « 2 » indique que l'opération est réalisée 2 fois, ce qui revient à laisser une ligne vide.

Une autre manière d'analyser notre problème est décrite dans l'algorithme suivant :

- début
- réaliser 4 fois :
  - affichage d'un message
  - affichage des astérisques
- fin

Bien sûr le message est différent à chaque fois. Nous allons mettre les 4 messages dans une table, nous désignerons chaque message par son numéro. Le troisième message est par exemple « ENTRECOTE-FRITES ». Ecrivons le programme :

```
1 * Menu (2ème version)
20
30 TABLEAU CHAINE MES[4]
40 MES[1]←'       M E N U';MES[2]←'   SAL
ADE NICOISE'
50 MES[3]←'  ENTRECOTE-FRITES';MES[4]←'
FROMAGE OU DESSERT'
60
70 FAIRE 100 POUR I←1 JUSQUA 4
80 AFFICHER [2/]MES[I]
100 AFFICHER [2/,7X,5'*']
110
120 TERMINER
```

En ligne 30 on désigne par « MES » une table contenant 4 éléments. Ces éléments sont des chaînes c'est-à-dire des suites de caractères. La déclaration « tableau chaîne » est imposée par L.S.E. Nous choisissons le nom et le nombre d'éléments. En ligne 40 et 50 on met les messages dans la table. La ligne 70 indique que les lignes 80 et 100 seront exécutées 4 fois chacune. En ligne 80 MES [ I ] désigne le ième message. Dans la ligne 100 il y a une liste de 3 éléments entre crochets :
- le premier indique 2 sauts de ligne
- le second le tracé de 7 espaces
- le troisième le tracé de 5 astérisques

Noter que le X désigne un espace lorsqu'il est entre crochets dans l'instruction AFFICHER. Si l'on exécute ce programme, on obtient le dessin désiré.
Pour remplir les éléments de la table on utilise l'instruction d'affectation symbolisée par le signe « ← » (remplacé par « _ » sur certains terminaux - cas de la majorité des imprimantes). C'est cette même instruction qui sert à initialiser la variable I de la ligne 70.





## II.4 APPRENONS À LIRE

Les exemples que nous avons rencontrés jusqu'à présent ne comportaient que des affichages figés et n'avaient pas besoin de données supplémentaires pour s'exécuter.
L'instruction LIRE permet de transmettre des données à un programme à partir du clavier.
Prenons l'exemple d'un programme calculant un quotient. On peut l'écrire :

```
1 * Calcul d'un quotient
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 TERMINER
```

Lors de la rencontre de l'instruction LIRE A en ligne 10, l'ordinateur s'interromp et attend la frappe d'une valeur numérique au clavier. Cette valeur sera rangée dans un emplacement mémoire repéré par l'étiquette A ; on dit que A est un identificateur. En ligne 20, la valeur tapée sera repérée par B et en ligne 30, la machine affichera « Le quotient est » suivi de la valeur de A divisé par B (A/B).

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Dividende ? 23
Diviseur ? 7
Le quotient est 3.2857143
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Dividende ? 3.55
Diviseur ? 0.71
Le quotient est 5
TERMINE EN LIGNE 40
```

Il est fastidieux d'être obligé de tapez EX@ et 1@ à chaque fois pour relancer le programme. Aussi, allons-nous le modifier : au lieu de lui dire de terminer en ligne 40, nous allons lui dire de recommencer, c'est-à-dire de revenir en ligne 10. Tapons :

```
40 ALLER EN 10
LISTER LIGNES *
1 * Calcul d'un quotient
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 ALLER EN 10
```

La nouvelle ligne 40 a remplacé l'ancienne.

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Dividende ? 32747
Diviseur ? 2672
Le quotient est 12.255614

PRET EN LIGNE 10
```

Notre programme ne s'arrête plus et « boucle » indéfiniment. Le seul moyen de l'interrompre est d'appuyer sur la touche « STOP ».
Ce moyen n'est guère élégant. Il est préférable de poser, à la fin du calcul, une question à l'utilisateur.





Tapons :

```
5 CHAINE REP
40 AFFICHER 'Voulez-vous continuer (O ou
 N) ? ';LIRE REP
50 SI REP='N' ALORS TERMINER
60 ALLER EN SI REP='O' ALORS 10 SINON 40
LISTEN LIGNES *
1 * Calcul d'un quotient
5 CHAINE REP
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Voulez-vous continuer (O ou
 N) ? ';LIRE REP
50 SI REP='N' ALORS TERMINER
60 ALLER EN SI REP='O' ALORS 10 SINON 40
```

Remarquons que les lignes que l'on vient d'ajouter se sont correctement insérées aux bons endroits.
La ligne 5 est une ligne de déclaration. Elle est obligatoire pour indiquer au système que l'identificateur REP représentera non pas un nombre comme A et B mais une chaîne de caractères (normalement réduite à un seul caractère : O ou N), formée par la réponse de l'utilisateur à la question qui lui est posée en ligne 40.
En ligne 50 nous avons une instruction conditionnelle : TERMINER ne sera exécuté que dans le cas où le contenu de REP sera effectivement un « N ».
En ligne 60, ALLER EN est suivi d'une expression numérique conditionnelle. Cette expression vaut 10 si le contenu de REP est un « O », elle vaut 40 dans tout autre cas.

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Voulez-vous continuer (O ou N) ? O
Dividende ? 63
Diviseur ? 7
Le quotient est 9
Voulez-vous continuer (O ou N) ? NON
Voulez-vous continuer (O ou N) ? n
Voulez-vous continuer (O ou N) ? N
TERMINE EN LIGNE 50
```

Une autre manière de résoudre ce problème est l'utilisation de la structure de « boucle ».
Tapons :

```
LISTER LIGNES 5
5 CHAINE REP

/;REP←'O'
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
EF@FACER LIGNES 60@

LISTER LIGNES *
1 * Calcul d'un quotient
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
```





```
EXECUTER A PARTIR DE 1

Dividende ? 39
Diviseur ? 3
Le quotient est 13
Tapez "O" pour continuer O
Dividende ? 72
Diviseur ? 9
Le quotient est 8
Tapez "O" pour continuer o
TERMINE EN LIGNE 50
EXECUTER A PARTIR DE 1

Dividende ? 45
Diviseur ? 9
Le quotient est 5
Tapez "O" pour continuer ZUT
TERMINE EN LIGNE 50
```

Cette boucle est équivalente à un saut conditionnel (SI REP = 'O' ALORS ALLER EN 10) en fin de ligne 40.

Essayons quelque chose :

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 0
ERREUR E 31 EN LIGNE 30
```

Dans la liste des erreurs d'exécution (voir Annexes) nous trouvons :
ERREUR E 31 : Division par zéro.
Effectivement nous avons donné une valeur nulle au diviseur, B valait donc zéro et la machine a tenté, en ligne 30, d'exécuter A/B. Or, comme chacun sait, une division par zéro est impossible. Modifions notre programme pour lui apprendre cette règle et lui éviter de partir en erreur.

Tapons :

```
25 SI B=0 ALORS DEBUT AFFICHER 'Une divi
sion par zéro est impossible !';ALLER EN
 40 FIN
LISTER LIGNES *
1 * Calcul d'un quotient
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
25 SI B=0 ALORS DEBUT AFFICHER 'Une divi
sion par zéro est impossible !';ALLER EN
 40 FIN
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
```

Les mots DEBUT et FIN en ligne 25 ont un rôle de parenthésage afin de regrouper les deux instructions AFFICHER et ALLER EN en une seule instruction conditionnelle. Ainsi, si B est nul nous enverrons le message « Une division par zéro est impossible ! » à l'écran et nous sauterons la ligne 30 afin de ne pas évaluer A/B et donc ne pas produire d'erreur.

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Tapez "O" pour continuer O
Dividende ? 12
Diviseur ? 0
Une division par zéro est impossible !
Tapez "O" pour continuer O
Dividende ? 5
Diviseur ?              ... etc
```





Notre programme est maintenant muni d'une alternative et est capable d'exécuter le traitement adapté aux données qui lui sont communiquées.
Le problème que nous nous sommes posé est un problème très simple et il n'est point besoin d'utiliser un ordinateur pour calculer un quotient (une calculatrice le fera tout aussi bien); il nous a cependant permis de voir, outre les instructions LIRE et AFFICHER, les notions de répétition et d'alternative.

# CHAPITRE III
# NOTIONS FONDAMENTALES DU L.S.E.

- MODE CALBUR
- MODE PROGRAMME
- LIGNES
- CONSTANTES
- VARIABLES SIMPLES
- VARIABLES INDICÉES
- OPÉRATEURS
- EXPRESSIONS

# NOTIONS FONDAMENTALES DU LSE

## III-1 MODE CALBUR ou calculateur de bureau

Ce mode permet d'évaluer toute expression L.S.E. et d'en afficher la valeur.
– L'expression doit être précédée du symbole ? signifiant AFFICHER, et suivie du caractère ENTRÉE signalant la fin de l'instruction.
– Toutes les opérations et les fonctions L.S.E peuvent être utilisées.

***Exemple :***

```
? LGR('Bonjour'!' Madame')
14
```

– On peut aussi écrire une ligne de programme (sans numéro de ligne). Après ENTRÉE la ligne est analysée syntaxiquement et, si elle est correcte, elle est immédiatement exécutée.

***Exemple :***

```
A←2;B←5;?A>B
.FAUX.
```

Les instructions interdites sont PAUSE, TERMINER, EXECUTER, PROCEDURE, RESULTAT, RETOUR et les instructions faisant référence à un numéro de ligne (RETOUR, RETOUR EN, ALLER EN).
– En mode CALBUR on peut également effectuer les opérations sur fichiers (GARER, CHARGER, SUPPRIMER).

Le mode CALBUR est aussi souvent utilisé pour la mise au point des programmes. En utilisant la commande PAs à pas ou l'instruction PAUSE on peut connaître les valeurs prises par certaines variables.





## III-2 MODE PROGRAMME

– Un programme est une suite d'instructions qui permettent à la machine d'effectuer l'ensemble des opérations nécessaires à la réalisation d'un travail donné.
– Un programme est constitué de lignes commençant par un numéro. L'exécution des traitements contenus dans ces lignes est différé jusqu'à ce que l'on envoie à la machine la commande EXécuter.

## III-3 LIGNES

Pour écrire une ligne de programme, on doit taper dans l'ordre :
– Un numéro de ligne (compris entre 1 et 32000) suivis d'au moins un espace.
– Une ou plusieurs instructions séparées par des points virgules. La dernière instruction peut être suivie d'un commentaire : celui-ci doit être alors précédé d'une étoile (*).

***Exemple :***

```
60 A←2;B←5;* Initialisation des variable
s A et B
```

Le caractère ENTRÉE, signalant la fin de la ligne.
A ce moment la ligne est analysée :
Si elle est syntaxiquement correcte, elle est intégrée au programme (s'il existe déjà une ligne de même numéro, elle la remplace).
Sinon un message d'erreur est envoyé :
ERREUR C n (erreur dite de compilation, voir liste en Annexe)
On peut corriger cette ligne (voir Editeur de ligne), ou la retaper.
Une ligne ne peut contenir plus de 250 caractères.

## III-4 LES CONSTANTES

Une donnée introduite une fois pour toute dans le programme s'appelle une constante.

### IV-4.1 Les constantes numériques

Ce sont des nombres écrits sous leurs formes usuelles à deux détails près :
La virgule décimale est remplacée par un point

***Exemple :***

4.5 s'écrit 4.5

Les puissances de 10 sont symbolisées par la lettre E

***Exemple :***

1.234567 . $10^6$ s'écrit 1.234567E6
0,25 . $10^{-4}$ s'écrit 2.5E-5

### IV-4.2 Les constantes chaînes

Une constante chaîne est une suite de caractères. Ces caractères peuvent être des lettres, des chiffres, des caractères particuliers ( + − ✳ /...) et d'une manière générale n'importe lequel des 256 caractères pouvant être représentés sur un octet.
Une constante chaîne peut être représentée de plusieurs manières :
– En encadrant les caractères qui constituent la chaîne par des apostrophes (').
Dans ce cas si l'on veut mettre des apostrophes dans la chaîne, on mettra 2 apostrophes (ne pas confondre avec le guillemet). A l'impression le texte apparaîtra avec une seule apostrophe.

***Exemple :***

```
?'L''enseignement de l''informatique est
 utile'
L'enseignement de l'informatique est uti
le
```

– En désignant les caractères par leurs équivalents décimaux, donnés par le code ASCII, séparés par un espace ; la chaîne est alors encadrée par deux points.





***Exemple :***

```
? .76 39 73 78 70 79 82 77 65 84 73 81 6
5 69.
L'INFORMATIQUE
```

Ce mode est surtout utilisé pour représenter les caractères de contrôle ou de gestion d'écran ainsi que les caractères non disponibles au clavier.
– On peut aussi mélanger les deux types de représentations.

***Exemple :*** (le code 10 représentant un passage à la ligne)

```
? 'A'.10.'B'
A
 B
```

***Remarques :***
– '' est une chaîne vide ; elle ne contient aucun caractère.
– Les constantes numériques et les constantes chaînes sont de types différents.

***Exemples :***

– ne pas confondre le nombre 10 avec la chaîne '10'
– l'expression '10' + '10' n'a aucun sens en L.S.E.

### III-4.3 Les constantes booléennes

Ces constantes sont au nombre de 2 :
la constante notée .VRAI.
la constante notée .FAUX.

## III-5 LES VARIABLES SIMPLES

Une variable simple est un objet auquel on peut affecter une valeur, cette valeur pouvant être modifiée au cours du programme.
Il existe 4 types de variables en L.S.E. :
Les variables numériques
Les variables chaînes
Les variables booléennes
Les variables graphiques

Les noms de variables appelés IDENTIFICATEURS sont soumis à la règle suivante :
Le nom d'un identificateur est une suite d'au plus 5 caractères alphanumériques (lettres ou chiffres); le premier caractère est obligatoirement une lettre. Les minuscules du nom sont automatiquement transformées en majuscules.

***Exemple :***

A, TEXTE, X1, B12 sont des identificateurs
PAS, CIBLE, DEBUT ne peuvent pas être des identificateurs car ce sont des mots réservés par L.S.E. (voir liste en Annexe).
1A, T 4, PAULINE ne peuvent pas être des identificateurs.
En mode de bureau on ne peut créer que des identificateurs d'une lettre.
Les variables chaînes, booléennes et graphiques doivent être soumises à une déclaration préalable au moyen de l'une des instructions suivantes :
CHAINE , BOOLEEN , FORME

***Exemples :***

```
20 CHAINE CH,C,LISTE
30 BOOLEEN B,ENCOR
40 FORME CARTE,F,G;* Voir chapitre sur l
e graphique
```

Les variables non déclarées sont considérées comme numériques. Les déclarations se font généralement en début de programme et bien sûr avant leur première utilisation.
Une variable peut être définie par l'instruction d'affectation : "←".

***Exemples :***

```
130 R←RAC(2)
250 CHAINE TEXT;TEXT←'rien ne sert de co
urir'
```

Toute tentative d'utilisation d'une variable non définie provoquera une erreur d'exécution.





## III-6 LES VARIABLES INDICÉES

***1er exemple :***

```
10 LIRE [/,'Nombre d''éléments de la lis
te N=',U,/]N
11 * Au signal sonore, on entrera au cla
vier le nombre N puis l'on tapera sur la
 touche ENTREE
16
17 TABLEAU NOMB[N]
20 LIRE NOMB
21 *On entre au clavier la liste de nomb
res en tapant ENTREE après chaque nombre
25 MAX←NOMB[1]
30 FAIRE 36 POUR I←2 JUSQUA N
36 SI NOMB[I]>MAX ALORS MAX←NOMB[I]
40
45 AFFICHER 'Le plus grand élément est:
',MAX
50 TERMINER

EXECUTER A PARTIR DE 1

Nombre d'éléments de la liste N=12

45 2 -3 5 6 7 8 112 -1 13 17 59

Le plus grand élément est: 112
TERMINE EN LIGNE 50
```

Nous avons créé un tableau NOMB de N nombres. Ce tableau est de dimension 1.
NOMB[1] correspond au 1er élément de la liste.
. . . . . .

NOMB[3] correspond au 3e élément de la liste.
. . . . . .

Ce qui se trouve entre crochets est l'indice.
Ce tableau a été déclaré en utilisant l'instruction TABLEAU.

TABLEAU NOMB[N]

L'instruction de déclaration définit donc :
- la dimension du tableau,
- le nombre d'éléments du tableau.

***2e exemple :***

Proposer 5 mots en français et faire écrire à l'utilisateur du programme les traductions en anglais de ces 5 mots (si la réponse est fausse, on donne la « bonne » traduction).

```
1 * Traduction de mots français en angla
is
5
10 TABLEAU CHAINE MOT[2,5];CHAINE REP
20 MOT[1,1]←'CHIEN';MOT[1,2]←'CHAT';MOT[
1,3]←'POISSON';MOT[1,4]←'OISEAU';MOT[1,5
]←'VACHE'
25 MOT[2,1]←'DOG';MOT[2,2]←'CAT';MOT[2,3
]←'FISH';MOT[2,4]←'BIRD';MOT[2,5]←'COW'
30
35 AFFICHER [/,'On te propose 5 mots en
français, tu les traduiras en anglais',/
]
40
45 FAIRE 70 POUR I←1 JUSQUA 5
50 AFFICHER [/,U,' ----------> ']MOT[1,I
]
55 LIRE REP;REP←GRL(TMA(REP),1);* On "ne
ttoie" la réponse
60 SI REP=MOT[2,I] ALORS AFFICHER [/,'C'
'est bien',/] SINON AFFICHER [/,'Tu t''e
s trompé,',/,'la bonne réponse est: ',U,
/]MOT[2,I]
70
80 TERMINER
```





```
EXECUTER A PARTIR DE 1

On te propose 5 mots en français, tu les
 traduiras en anglais

CHIEN ----------> DOG
C'est bien

CHAT ----------> CAT
C'est bien

POISSON ----------> FICH
Tu t'es trompé,
la bonne réponse est: FISH

OISEAU ----------> BEARD
Tu t'es trompé,
la bonne réponse est: BIRD

VACHE ----------> COW
C'est bien

TERMINE EN LIGNE 80
```

Dans ce programme, on a créé un tableau à 2 lignes et à 5 colonnes. Ce tableau a été déclaré par l'instruction : TABLEAU CHAINE.

TABLEAU CHAINE MOT[2,5] définit un tableau à 2 dimensions et à 10 éléments. Pour repérer un élément de ce tableau, il suffit de connaître son numéro de ligne et son numéro de colonne.
MOT[2,3] est l'élément situé à la 2e ligne et à la 3e colonne
MOT[2,3] est une variable indicée.
Un tableau est un ensemble de variables de même type (numériques, chaînes, booléennes ou graphiques).
Les éléments d'un tableau (variables indicées) sont désignés par :
- le nom du tableau
- une liste d'indices (le nombre d'indices est égal à la dimension du tableau).

Il existe 4 types de tableau en L.S.E. :
les tableaux numériques; les tableaux chaînes; les tableaux booléens; les tableaux formes.
Ces tableaux doivent être déclarés par l'instruction TABLEAU.

Exemple :

```
20 TABLEAU N[5];* Déclaration d'un table
au numérique
30 TABLEAU CHAINE TEXTE[24];* Déclaratio
n d'un tableau de chaînes
40 TABLEAU BOOLEEN EXIST[3,25,25,40];* D
éclaration d'un tableau de booléens
50 TABLEAU FORME SCENE[8];* Déclaration
d'un tableau de formes - voir chapitre
sur le graphique
```

## III-7 LES OPÉRATEURS

### III-7.1 Les opérateurs numériques

- 1 opérateur unaire
- 5 opérateurs binaires

- pour le changement de signe
+ pour l'addition
- pour la soustraction
* pour la multiplcation
/ pour la division
↑ pour l'exponentiation

Le calcul s'effectue en principe de la gauche vers la droite en tenant compte d'un ordre de priorité décroissante :
l'exponentiation
le changement de signe
la multiplication ou la division
l'addition ou la soustraction
Cet ordre peut être modifié par l'introduction de parenthèses. Dans ce cas l'expression calculée en premier est celle qui est placée entre les parenthèses les plus internes.





*Exemples :*

```
A←10;B←-3;C←4

?A-3*B+C/4
20

?(A+B)/C
1.75

?(-B+(B-3*A*C)2)/5
3026,4

?(A+3*B)/(4*C)
0.0625
```

Une expression numérique ne peut évidemment pas contenir 2 opérateurs l'un à côté de l'autre.
Il est interdit d'élever un nombre négatif à une puissance non entière.
Il est également interdit d'élever 0 à une puissance négative ou nulle.

### III-7.2 La concaténation

Le seul opérateur sur chaîne est l'opérateur de concaténation noté ! (point d'exclamation).
Il permet de réunir 2 chaînes en une seule, les 2 chaînes étant mises bout à bout.

*Exemple :*

```
CHAINE A,B,C;A←'François apprend sa leço
n ';B←'d''anglais'
C←A!B
?C
François apprend sa leçon d'anglais
```

### III·7.3 Les opérateurs booléens

| | | |
|---|---|---|
| 1 opérateur unaire | NON | |
| 3 opérateurs binaires | ET | |
| | OU | (ou inclusif) |
| | DIJ | (ou exclusif) |

Dans l'évaluation d'une expression booléenne l'ordre des priorités est le suivant :
NON
ET
OU ou DIJ
On utilisera des parenthèses si l'on veut modifier l'ordre des priorités.

### III·7.4 Les opérateurs de relation

Ces opérateurs permettent de comparer 2 expressions de même nature (numériques, chaînes ou booléennes) :

| | |
|---|---|
| = | pour égal |
| ≠ | pour différent |
| < | pour inférieur |
| > | pour supérieur |
| < = | pour inférieur ou égal |
| > = | pour supérieur ou égal |

Seuls les opérateurs = et ≠ peuvent être utilisés entre deux expressions booléennes. La comparaison de deux expressions graphiques est pour l'instant dénuée de fondement.

***Exemples :***

```
? 2<=2
.VRAI.

? 2<RAC(3)
.FAUX.

? 'FRANCE'>'FRANCOIS'
.FAUX.

? 'BEAU'<'BEAUCOUP'
.VRAI.
```





### III·7.5 Les opérateurs graphiques

Ces opérateurs sont au nombre de 2 :
- la juxtaposition (symbole :)
- la superposition (symbole %)

Voir le chapitre traitant du graphique.

## III·8 LES EXPRESSIONS

Une expression est une combinaison de variables, de constantes, d'expressions entre parenthèses, de résultats de procédures, de résultats de fonctions ou d'expressions conditionnelles, liés par des opérateurs. Cette combinaison doit respecter les règles de grammaire du L.S.E.
Selon que le résultat est numérique, chaîne, booléen ou graphique l'expression est appelée expression numérique, chaîne, booléenne ou graphique.

***Exemples :***

A ← 3; SI A < 2 ALORS 5 SINON 0 est une expression numérique de valeur 0
A ← 2; B ← 1; B * &F (A) est une expression numérique si le résultat de la procédure &F(A) est numérique.

C étant une chaîne :
C!SI Z < 10 ALORS 'ECHEC' SINON 'SUCCES' est une expression chaîne.

P ← 15; Q ← 7
P > 7 ET Q < 11 est une expression booléenne de valeur .VRAI.
P > 17 OU Q < = 7 est une expression booléenne de valeur .VRAI.
P > 7 DIJ Q < 11 est une expression booléenne de valeur .FAUX.
NON (P > 16) est une expression booléenne de valeur .VRAI.
NON (P < 7) ET Q > 11 est une expression booléenne de valeur .FAUX.
NON (P > 7 ET Q > 11) est une expression booléenne de valeur .VRAI.
CHAINE A; SI A > 'JEUDI' ALORS .VRAI. SINON .FAUX. est une expression booléenne.

## EXPRESSIONS CONDITIONNELLES

Il est possible en L.S.E. de créer des expressions dont la valeur dépend du résultat d'une expression booléenne.
Syntaxe : SI eb ALORS exp1 SINON exp2
eb est une expression booléenne
exp1 et exp2 sont des expressions de même nature.
Si eb est vraie l'expression vaut exp1. dans le cas contraire elle vaut exp2.

*Exemple :*

```
A←3

?SI A=1 ALORS 2+2 SINON 3+3
6

?'Plus '!SI A<5 ALORS 'petit' SINON 'gra
nd'
Plus petit
```





# CHAPITRE IV
# RUPTURES DE SÉQUENCE ET ITÉRATIONS

- INSTRUCTIONS CONDITIONNELLES
- INSTRUCTION ALLER EN
- BOUCLES

# RUPTURES DE SÉQUENCE ET ITÉRATIONS

## IV-1 LES INSTRUCTIONS CONDITIONNELLES.

Dans un programme, il est rare qu'un traitement puisse s'exécuter séquentiellement (c'est-à-dire que toutes les instructions puissent s'exécuter dans l'ordre croissant des numéros de ligne).
Le plus souvent, selon le résultat de certains tests, le traitement sera différent.

On utilise alors l'instruction SI ..... ALORS ....
ou l'instruction SI ..... ALORS ..... SINON ....

*1er exemple :*

instruction SI ... ALORS ...

Sachant qu'une disquette coûte 20 FRANCS, et que pour un achat d'au moins 25 unités, le commerçant vous fait une remise de 5%, écrire un programme qui permet, selon le nombre de disquettes achetées, d'afficher le prix à payer.





```
1 * UNE FACTURE
2
10 LIRE [/,'Combien de disquettes désire
z-vous ? N= ']N
15
20 PRIX←N*20
25 SI N>=25 ALORS PRIX←PRIX-PRIX*5/100
30
35 AFFICHER [2/,'TOTAL A PAYER :',U,' FR
ANCS',2/]PRIX
40
45 TERMINER

EXECUTER A PARTIR DE 1

Combien de disquettes désirez-vous ? N=
30

TOTAL A PAYER :570  FRANCS


TERMINE EN LIGNE 45
```

La variable PRIX n'est modifiée que si l'expression booléenne N > = 25 a la valeur .VRAI.

*2e exemple :*

instruction SI ... ALORS ... SINON

On se propose de lire 2 nombres et de les comparer

```
1 * Comparaison de 2 nombres
5
10 LIRE [/,'A=',U,'    B=',U,/]A,B
15
20 SI A<B ALORS AFFICHER A,'< ',B SINON
SI A>B ALORS AFFICHER A,'>',B SINON AFFI
CHER A,'= ',B
25
30 TERMINER

EXECUTER A PARTIR DE 1

A=2    B=5

2 < 5
TERMINE EN LIGNE 30
```

Lorsque plusieurs instructions conditionnelles sont imbriquées, chaque SINON est associé au ALORS qui le précède.

*3e exemple :*

instruction SI ... ALORS DEBUT ..... FIN
Même énoncé que dans l'exemple 2.

```
10 LIRE [/,'A=',U,'     B=',U,/]A,B
15
20 SI A<B ALORS DEBUT AFFICHER [U,'< ',U
]A,B;TERMINER FIN
25 SI A>B ALORS DEBUT AFFICHER [U,'> ',U
]A,B;TERMINER FIN
30 AFFICHER [U,'= ',U]A,B
35
40 TERMINER
```

Le traitement à effectuer dans le cas ou l'expression booléenne A > B a la valeur .VRAI. comporte 2 instructions. Il est obligatoire d'encadrer ces 2 instructions par DEBUT .... FIN.





## IV-2 L'INSTRUCTION ALLER EN

*Exemple :*

```
1 * exemple d' ALLER EN
10
20 AFFICHER 'Ce programme propose d''aff
icher'
25 AFFICHER 'soit: * * * * (choix 1)'
30 AFFICHER 'soit: & & & & (choix 2)'
40 AFFICHER 'soit: ! ! ! ! (choix 3)'
50 AFFICHER 'soit: - - - - (choix 4)'
60
70 TABLEAU CHOIX[4];CHOIX[1]←@110;CHOIX[
2]←@120;CHOIX[3]←@130;CHOIX[4]←@140
80 LIRE [/,'Quel est votre choix ? (donn
er le numéro)     ']I
90 ALLER EN CHOIX[I]
100
110 AFFICHER '* * * *';TERMINER
120 AFFICHER '& & & &';TERMINER
130 AFFICHER '! ! ! !';TERMINER
140 AFFICHER '- - - -';TERMINER

EXECUTER A PARTIR DE 1

Ce programme propose d'afficher
soit: * * * * (choix 1)
soit: & & & & (choix 2)
soit: ! ! ! ! (choix 3)
soit: - - - - (choix 4)
Quel est votre choix ? (donner le numéro
)     3
! ! ! !
TERMINE EN LIGNE 130
```

Noter l'utilisation du marqueur @ , qui permettra la prise en compte de tous les numéros de ligne par la commande de renumérotation ESpacer lignes.

# IV-3. LES BOUCLES

## IV-3.1. Présentation

Lors de l'exécution d'une tâche, nous avons souvent besoin de répéter un certain nombre de fois une action avant de pouvoir passer à la suite. Par exemple lorsque nous prenons notre café au lait matinal, nous devons le sucrer (3 sucres) puis tourner pour faire fondre le sucre.
Cette tâche peut se décrire de la façon suivante :
- Répéter 3 fois : mettre un sucre dans le café
- Si le sucre n'est pas fondu : tourner.

En utilisant le vocabulaire du L.S.E. cette action peut s'écrire :
FAIRE POUR NBSUC ← 1 JUSQUA 3
    - Mettre un sucre dans le café
FAIRE TANT QUE sucre non fondu
    - Tourner

Nous avons deux boucles, c'est-à-dire deux actions qui se répètent tant que certaines conditions ne sont pas réalisées. Ces deux boucles sont de natures différentes au niveau des conditions de sortie de boucle :
Dans la première, nous comptons le nombre de sucres et c'est la comparaison de ce nombre à 3 qui décide de la sortie de la boucle. Cela impose en particulier de connaître à l'avance le nombre de sucres nécessaire.
Dans la deuxième, la condition de sortie est différente : nous ne pouvons pas savoir à l'avance combien de fois ou combien de temps il nous faudra tourner, le seul critère d'arrêt étant que le sucre soit complètement fondu.

Il existe deux types de boucles les boucles « JUSQUA » et les boucles « TANT QUE ».

Imaginons que nous ne sachions pas combien il faut de sucres pour que notre café soit correctement sucré. Nous pouvons procéder de la manière suivante :
FAIRE TANT QUE café pas assez sucré
    - Mettre un sucre dans le café
    - FAIRE TANT QUE sucre pas fondu
        - Tourner

Nous avons ici deux boucles « TANT QUE » imbriquées.





## IV-3.2 Boucles TANT QUE

***Premier exemple :***

Nous allons écrire un programme qui lit une série de mots au clavier et les place « bout à bout » dans une chaine. Nous décidons de taper FIN pour arrêter la saisie.

```
1 * Exemple de boucle TANT QUE
10 CHAINE LISTE,MOT;MOT←'';LISTE←''
20 FAIRE 40 TANT QUE MOT≠'FIN'
30 LISTE←LISTE!MOT!'-'
40 LIRE MOT;AFFICHER [/];MOT←TMA(MOT)
50 AFFICHER 'La liste est: ',LISTE
60 TERMINER

EXECUTER A PARTIR DE 1
chien
chat
canard
oiseau
fin

La liste est: -CHIEN-CHAT-CANARD-OISEAU-
TERMINE EN LIGNE 60
```

***Remarques :***
- FAIRE est suivi du numéro de la dernière ligne de la boucle.
- Une telle boucle peut être écrite avec des ALLER EN. il suffirait de remplacer les lignes 20 et 40 par :

20 SI MOT = 'FIN' ALORS ALLER EN 50
40 LIRE MOT: MOT ← TMA(MOT): ALLER EN 20

mais le programme serait moins lisible.

*Deuxième exemple :*

Écrivons un programme qui calcule la moyenne d'une série de notes. Nous décidons de taper une note non valide (inférieure à 0 ou supérieure à 20) pour arrêter la saisie. Nous allons avoir besoin d'un compteur, qui nous indique le nombre de notes ainsi rentrées.

```
1 * Calcul de moyenne
10 TOTAL←0
20 AFFICHER 'Tapez les notes: '
30 LIRE NOTE
40 FAIRE 50 POUR NBRE←0 TANT QUE NOTE>=0
 ET NOTE<=20
50 TOTAL←TOTAL+NOTE;LIRE NOTE
60 AFFICHER NBRE, 'note(s) tapée(s)'
70 SI NBRE≠0 ALORS AFFICHER 'La moyenne
est: ',TOTAL/NBRE
80 TERMINER

exECUTER A PARTIR DE 1

Tapez les notes: 8 10 12 -1
3  note(s) tapée(s)
La moyenne est: 10
TERMINE EN LIGNE 80
```

Le compteur NBRE est incrémenté à chaque passage dans la boucle.





*Troisième exemple :*

Nous allons utiliser le mot PAS pour écrire un programme qui effectue des décompositions de nombres entiers en produits de facteurs premiers.
Pour décomposer un nombre en facteurs premiers, il faut le diviser par la suite des nombres premiers (soit : 2, 3, 5, 7, 11, 13, 17, 19, 23....). Cela allongerait par trop le temps d'exécution de faire générer cette suite par l'ordinateur ; aussi allons-nous nous contenter de faire des divisions par la suite des nombres entiers. Cependant nous pouvons faire l'économie d'un grand nombre de divisions : après 2, les nombres premiers ne peuvent pas être pairs, il nous suffit donc d'examiner les nombres de 2 en 2 ; après 3, les nombres premiers ne peuvent être des multiples de 3, (nous avons déjà enlevé les multiples pairs, il reste donc les multiples impairs qui sont disposés de 6 en 6). Pour les « sauter », il suffit donc de parcourir la suite des entiers avec un pas alternativement égal à 2 et à 4 (à partir de 7). Notre parcours sera donc : 2, 2 + 1 = 3, 3 + 2 = 5, 5 – 2 = 7, 7 + 4 = 11, 11 + 2 = 13, 13 + 4 = 17, 17 + 2 = 19, 19 + 4 = 23, 23 + 2 = 25, 25 + 4 = 29....
Nous éliminons bien ainsi tous les multiples de 2 et de 3.

```
1 * Décomposition en produit de facteurs
 premiers
10 AFFICHER 'Décomposition de ? ';LIRE N
15 SI N<2 OU ENT(N)≠N ALORS DEBUT AFFICH
ER 'Valeur incorrecte';ALLER EN 10 FIN
20 P←1
30 FAIRE 50 POUR I←2 PAS P TANT QUE I<=R
AC(N)
40 SI I*ENT(N/I)=N ALORS DEBUT AFFICHER
[U,X]I;N←N/I;ALLER EN 40 FIN
50 P←SI I>6 ALORS 6-P SINON SI I>2 ALORS
 2 SINON 1
60 AFFICHER [U,/]SI N=1 ALORS '' SINON N
70 TERMINER

EXECUTER A PARTIR DE 1

Décomposition de ? 1001 7  11  13

TERMINE EN LIGNE 70
```

La ligne 50 permet de gérer la valeur du pas : si I est égal à 2, P sera égal à 1 ce qui nous donnera bien la valeur de 3 pour I dans le passage suivant. Ensuite, tant que I est compris entre 3 et 7, le pas P vaut 2, ce qui nous donne pour I les valeurs 5 et 7. Ensuite P reçoit la valeur 6-P et est donc égal alternativement à 4 et à 2 (6 – 2 = 4, 6 – 4 = 2).

### IV-3.3 Boucles JUSQUA

On utilise ce type de boucle lorsque le nombre de répétitions est connu à l'avance.

*Exemple :*

```
1 * Affichage d'une phrase N fois
10 CHAINE PH;PH←'Je ne dois pas bavarder
 en classe.'
20 AFFICHER 'Nombre de répétitions ? ';L
IRE N
30 FAIRE 40 POUR I←1 JUSQUA N
40 AFFICHER PH
50 TERMINER

EXECUTER A PARTIR DE 1

Nombre de répétitions ? 5
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
TERMINE EN LIGNE 50
```

Il est également possible d'utiliser cette instruction lorsque le nombre de répétitions n'est pas connu à l'avance mais peut être calculé dans la boucle (la valeur qui suit JUSQUA est recalculée à chaque passage).





*Exemple :*

On désire compter et supprimer toutes les voyelles d'une chaîne :

```
1 * Compter et supprimer les voyelles
10 CHAINE CH,VOY;VOY←'AEIOUYaeiouyàâéèêë
îïôûüù'
20 AFFICHER 'Tapez la chaîne: ';LIRE CH
25 NBRE←0
30 FAIRE 40 POUR I←1 JUSQUA LGR(CH)
40 SI POS(VOY,1,SCH(CH,I,1))≠0 ALORS DEB
UT NBRE←NBRE+1;CH←MCH(CH,I,1,'');I←I-1 F
IN
50 AFFICHER 'Il y avait ',NBRE,' voyelle
(s)'
60 AFFICHER 'La chaîne est maintenant: '
,CH
70 TERMINER

EXECUTER A PARTIR DE 1

Tapez la chaîne: bonjour madame
Il y avait 6  voyelle(s)
La chaîne est maintenant: bnjr mdm
TERMINE EN LIGNE 70
```

Il est possible dans une boucle « JUSQUA » de préciser la valeur de l'incrément.

*Exemple :*

Multiples de 7 inférieurs à 100 :

```
1 * Multiples de 7
10 FAIRE 20 POUR I←0 PAS 7 JUSQUA 100
20 AFFICHER [U,X]I
30 TERMINER

EXECUTER A PARTIR DE 1
0  7  14  21  28  35  42  49  56  63  70
  77  84  91  98
TERMINE EN LIGNE 30
```

Tout comme pour les boucles « TANT QUE » le PAS peut être variable :

*Exemples :*

```
 1 * Une façon originale d'obtenir les p
uissances de 2
10 FAIRE 20 POUR I←1 PAS I JUSQUA 1000
20 AFFICHER [U,X]I
30 TERMINER

EXECUTER A PARTIR DE 1
1  2  4  8  16  32  64  128  256  512
TERMINE EN LIGNE 30
```





Ou bien encore :

```
1 * Etude de la série des "1 sur 2 puiss
ance n"
10 AFFICHER 'Précision ? ';LIRE STOP;SOM
ME←0
15 AFFICHER ''
20 FAIRE 30 POUR I←1 PAS -I/2 JUSQUA STO
P
30 SOMME←SOMME+I;AFFICHER [U,X]SOMME
40 TERMINER

EXECUTER A PARTIR DE 1

Précision ? 0.02
1  1.5  1.75  1.875  1.9375  1.96875
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Précision ? 0
1  1.5  1.75  1.875  1.9375  1.96875  1.
984375  1.9921875  1.9960938  1.9980469
 1.9990234  1.9995117  1.9997559  1.9998
779  1.999939  1.9999695  1.9999847  1.9
999924  1.9999962  1.9999981  1.999999
1.9999995  1.9999998  1.9999999  1.99999
99  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2
ERREUR E 44 EN LIGNE 20
```

### IV-4.4 Remarques concernant les boucles (N1 FAIRE N2...)

– Si dans la boucle on rencontre l'instruction ALLER EN N3 (N3 non compris entre N1 et N2) on sort de la boucle, même si la condition de sortie n'est pas réalisée.

– Si l'on arrive dans la boucle par l'instruction ALLER EN N3 (N3 compris entre N1 et N2) alors la ligne N3 et les suivantes se comportent comme des lignes ordinaires (fortement déconseillé).
– Si dans le corps de la boucle, une instruction renvoie à l'instruction FAIRE, celle-ci est réexécutée, et tous les paramètres sont réinitiaiisés.





# CHAPITRE V
# PROCÉDURES

- INTRODUCTION
- INSTRUCTION PROCEDURE
- VARIABLES LOCALES, VARIABLES GLOBALES
- PASSAGE DE PARAMÈTRES PAR ADRESSE
- PASSAGE DE PARAMÈTRES PAR VALEUR
- INSTRUCTION RETOUR
- INSTRUCTION RETOUR EN
- INSTRUCTION RESULTAT
- PROCÉDURES BINAIRES
- PROCÉDURES EXTERNES
- PROCÉDURES ET CALBUR
- LE L.S.E. ET LA RÉCURSIVITÉ
- PARAMÈTRES NOMS DE PROCÉDURES

# PROCÉDURES

## V.1 INTRODUCTION

La notion de la sous-programme (terme qui englobe les procédures de type RETOUR et celles de type RESULTAT) est un des concepts les plus intéressants des langages modernes. Lorsqu'un problème complexe est posé, une des méthodes les plus efficaces pour le résoudre est de le décomposer en sous-problèmes de complexité moins grande. Ceci permet d'imaginer, de résoudre et de vérifier la solution d'un seul sous-problème à la fois. Ceci permet aussi le partage d'un travail par une équipe.

Cette méthode permet au niveau de l'écriture des algorithmes de les décomposer en sous-algorithmes, chacun plus facile à écrire et à tester que le programme entier.
On découpe le programme en composants accomplissant une action bien définie, particulièrement en ce qui concerne :
- les données d'entrée et de sortie
- les contraintes sur les données d'entrée
- le traitement des cas exceptionnels ou anormaux

Bien que cela ne soit pas obligatoire, il est de bonne pratique de ne pas utiliser dans un sous-programme de variables définies extérieurement, sans qu'elles soient passées en paramètres.

***Les deux types de procédures L.S.E.***

- celles qui font un certain travail mais qui ne retournent pas de valeur au programme appelant. Elles se terminent par RETOUR ou RETOUR EN. Elles s'utilisent dans un programme comme des instructions.

***Exemple :***

```
12000 PROCEDURE &EFECR()
12005 AFFICHER[.12.];RETOUR
```





Cette procédure pourra s'utiliser, par exemple, de la façon suivante :

```
30 C←X-1;SI C=0 ALORS DEBUT &EFECR();AFF
ICHER 'C''est fini!';TERMINER FIN
```

- celles qui retournent une valeur au programme appelant. Elles se terminent par l'instruction RESULTAT. Elles interviennent dans le programme comme des expressions. On peut les considérer comme des fonctions écrites en L.S.E. Elles peuvent néanmoins en plus de la valeur qu'elles rendent, effectuer un certain travail.

```
16300 PROCEDURE &DERCA(C);* Résultat: de
rnier caractère de la chaîne C
16310 RESULTAT SCH(C,LGR(C),1)
```

Ce qui pourra donner, à l'utilisation :

```
260 CAR←&DERCA(FRASE);SI CAR='.' ALORS .
....
```

## V.2 INSTRUCTION PROCEDURE

Cette instruction constitue la déclaration d'une procédure interne L.S.E. Elle obéit à l'une des syntaxes suivantes ;
PROCEDURE &NOM (liste1)
PROCEDURE &NOM (liste2) LOCAL liste2

Cette instruction doit se trouver en début de ligne.

NOM désigne le nom de la procédure. Tout nom respectant la syntaxe des noms L.S.E. peut être utilisé y compris les noms de fonctions et les mots réservés L.S.E. Ainsi, on peut à la fois utiliser la fonction LGR et la procédure &LGR.

liste1 désigne la liste des paramètres formels. Les éléments de la liste sont séparés par des virgules. Le nombre de paramètres est limité à 16 par le compilateur actuel. Ces paramètres formels sont des identificateurs (variables simples ou noms de tableaux) ou des noms de procédure. Chacun d'eux correspond au paramètre de même rang de l'instruction d'appel. Lors de l'appel, il prend la valeur et le type de celui-ci.

Tout élément de liste2 qui n'est pas dans liste1 est une variable locale (voir paragraphe suivant). Tout élément commun aux deux listes est un paramètre formel par valeur (voir plus loin). liste2 ne peut contenir plus de 16 éléments.

## V.3 VARIABLES LOCALES, VARIABLES GLOBALES

Dans les communications entre le programme appelant et la procédure appelée, il est possible, mais non souhaitable, d'utiliser des variables communes non transmises en paramètres; on les désigne sous le nom de variables globales.

***Exemple :***

Considérons une procédure &AFF destinée à afficher une chaîne en insérant un espace entre chaque caractère :

```
1 * Inconvénient des variables globales
5 CHAINE CH
10 AFFICHER 'Chaîne ? ';LIRE CH
20 &AFF(CH);TERMINER
30
1000 PROCEDURE &AFF(C);AFFICHER [/]
1010 FAIRE 1020 POUR I←1 JUSQUA LGR(C)-1
1020 AFFICHER [U,X]SCH(C,I,1)
1030 AFFICHER [U,X]SCH(C,I,1);RETOUR

EXECUTER A PARTIR DE 1

Chaîne ? Bonjour Madame
B o n j o u r   M a d a m e
TERMINE EN LIGNE 20
```





Nous pouvons avoir besoin d'utiliser la même procédure cette fois avec les éléments d'un tableau de chaînes :

```
1 * Inconvénient des variables globales
5 TABLEAU CHAINE T[3]
10 AFFICHER 'Tapez 3 chaînes:'
20 LIRE T
30 FAIRE 30 POUR I←1 JUSQUA 3;&AFF(T[I])
40 TERMINER
1000 PROCEDURE &AFF(C);AFFICHER [/]
1010 FAIRE 1020 POUR I←1 JUSQUA LGR(C)-1
1020 AFFICHER [U,X]SCH(C,I,1)
1030 AFFICHER [U,X]SCH(C,I,1);RETOUR

EXECUTER A PARTIR DE 1

Tapez 3 chaînes:
Bonjour Mesdames
Bonjour Mesdemoiselles
Bonjour Messieurs


B o n j o u r   M e s d a m e s
TERMINE EN LIGNE 40
```

Que s'est-il passé ? Pourquoi n'avons nous pas eu l'affichage des 3 chaînes du tableau T ? Demandons :

```
?I
17
```

La variable I est utilisée en deux endroits du programme : une fois en ligne 30 pour la gestion de la boucle parcourant les trois éléments du tableau T, une autre fois dans la procédure &AFF, en ligne 1010 pour extraire les uns après les autres les caractères de la chaîne C. Cette seconde utilisation a changé la valeur de I et donc perturbé le bon fonctionnement du programme.

Le premier remède qui vient à l'esprit consiste à remplacer l'un des deux I par un autre nom de variable, par exemple J pour la boucle qui est dans la procédure. Correction faite ça marche très bien. Pourtant ce n'est pas une bonne méthode car elle impose lors de l'écriture de la procédure de connaître le nom des variables des programmes appelant ce qui ne facilite guère la réalisation de procédures indépendamment du contexte de leur utilisation.

Une autre solution consiste à utiliser dans la procédure une variable locale et donc à modifier la ligne 1000 comme suit :

```
LISTER LIGNES 1000
1000 PROCEDURE &AFF(C);AFFICHER [/]

/;/ LOCAL I;
1000 PROCEDURE &AFF(C) LOCAL I;AFFICHER
[/]
```

Ainsi, dans le programme appelant, I représente la variable qui contrôle la boucle d'extraction des éléments du tableau T, et dans la procédure I désigne la variable qui contrôle l'affichage des caractères successifs de la chaîne. Les deux variables, tout en ayant le même nom ne représentent pas la même chose. A un instant donné, on n'a accès qu'à l'une d'entre elles. Noter en conséquence, que le fait de déclarer locale une variable, interdit l'usage d'une variable globale de même nom.

Essayons notre programme ainsi modifié :

```
EXECUTER A PARTIR DE 1

Tapez 3 chaînes:
Bonjour Mesdames
Bonjour Mesdemoiselles
Bonjour Messieurs


B o n j o u r   M e s d a m e s
B o n j o u r   M e s d e m o i s e l l
e s
B o n j o u r   M e s s i e u r s
TERMINE EN LIGNE 40
```

Ça marche !!!
Une variable locale à une procédure est donc connue :
- dans la procédure où elle est définie
- dans les procédures qu'elle appelle à condition dans ce cas qu'elle ne soit pas elle même déclarée locale dans la procédure appelée.





Une bonne manière de traiter la question consiste, dans une procédure, à déclarer locale toute variable qui n'est pas un paramètre.
Si les variables locales représentent des chaînes, des formes, des booléens ou des tableaux de type quelconque, elles devront être déclarées comme telles à l'intérieur de la procédure.
Lors du traitement de la fin de la procédure par les instructions RETOUR, RETOUR EN ou RESULTAT, les variables locales de la procédure sont libérées et leurs valeurs sont perdues.

Notons que les motifs, qui ne sont pas des variables, ont un statut global. Dès qu'un motif est défini, il est connu dans tout le programme mais aussi dans les procédures externes appelées par le programme (voir chapitre sur le graphique).

## V.4 PASSAGE DE PARAMETRES PAR NOM (ou adresse)

Les paramètres qui se trouvent dans l'instruction d'appel sont les paramètres effectifs de l'appel. Ceux qui se trouvent dans la déclaration sont les paramètres formels de la déclaration.

Dans le cadre de notre exemple précédent nous transmettons à la procédure le nom du paramètre effectif. Tout ce passe comme si le paramètre effectif (CH ou T [I]) s'appelait C à l'intérieur de la procédure. Au retour de cette procédure, les noms sont changés en sens inverse ce qui permet de récupérer dans les paramètres effectifs les résultats des traitements éventuels effectués dans la procédure. Cela signifie en particulier que si la procédure modifié le paramètre formel, le paramètre effectif sera également modifié.

*Exemple :*

```
1
10 A←3
20 AFFICHER 'Avant l''appel, A vaut ',A
30 &TRUC(A)
40 AFFICHER 'Au retour de la procédure,
A vaut ',A
50 TERMINER
300 PROCEDURE &TRUC(X)
310 AFFICHER 'Le paramètre transmis vaut
 ',X
320 X←2*X
330 AFFICHER 'Son double est donc ',X
340 RETOUR

exECUTER A PARTIR DE 1

Avant l'appel, A vaut 3
Le paramètre transmis vaut 3
Son double est donc 6
Au retour de la procédure, A vaut 6
TERMINE EN LIGNE 50
```

Le paramètre est transmis par nom (on dit aussi par adresse). Le paramètre formel X est utilisé dans la procédure à la place de la variable A. Il ne sera pas possible dans la procédure d'accéder à A en la traitant comme une variable globale, on ne pourra le faire que par l'intermédiaire de X. Par contre, si c'est nécessaire, on pourra déclarer une variable locale de nom A qui n'aura de commun avec la variable de même nom du programme appelant. De même, si le programme appelant utilisait une variable X, cette dernière ne serait pas accessible dans la procédure.

Tapons :

```
&truc(7)
ERREUR E 15 EN LIGNE 300 APPEL LIGNE 80R
```

En effet, 7 est une constante, n'a pas de nom et ne peut donc pas être transmis à la procédure.





Lors d'un appel de procédure par nom, les paramètres effectifs peuvent être :
- des variables simples
- des variables indicées (éléments de tableau numérique, booléen, chaîne ou forme)
- des noms de tableaux
- des noms de procédures
- Il ne peuvent être ni des constantes, ni des expressions, ni des noms de motifs.

## V.5 PASSAGE DE PARAMÈTRES PAR VALEUR

Modifions notre procédure &TRUC :

```
liSTER LIGNES 300
300 PROCEDURE &TRUC(X)

/ LOCAL X
300 PROCEDURE &TRUC(X) LOCAL X


&TRUC(7)
Le paramètre transmis vaut 7
Son double est donc 14



EXECUTER A PARTIR DE 1

Avant l'appel, A vaut 3
Le paramètre transmis vaut 3
Son double est donc 6
Au retour de la procédure, A vaut 3
TERMINE EN LIGNE 50
```

La modification de la ligne 300 provoque le passage du paramètre par valeur. Ce passage se fait en recopiant dans les paramètres formels les valeurs correspondant aux paramètres effectifs.

Au retour, les valeurs contenues dans les variables correspondant aux paramètres formels sont perdues et les variables retrouvent le rôle qu'elles avaient avant l'appel.
Lors d'un appel de procédure certains paramètres peuvent être transmis par nom et d'autres par valeur.
Lors d'un passage par valeur les paramètres effectifs peuvent être :
– des expressions (et en particulier des variables)
– des noms de tableau
Il ne peuvent être des noms de motifs.

Puisqu'il y a recopie de la valeur du paramètre effectif dans le paramètre formel cette transmission est coûteuse en place mémoire.
Noter que dans un appel du type &PROC (X) on ne saura de quelle manière se fait la transmission qu'en regardant la ligne de déclaration de procédure. Si la transmission se fait par nom, au retour la variable X aura pu être modifiée, si la transmission se fait par valeur elle sera restituée sans changement.

## V.6 INSTRUCTION RETOUR

Cette instruction termine les procédures qui sont utilisées comme instructions. C'est elle qui rend le contrôle au programme appelant (à l'instruction qui suit l'instruction d'appel).

***Exemple :***

10 A←5; &AFFIC (A); A←10; AFFICHER A; TERMINER
100 PROCEDURE &AFFIC (A); AFFICHER A; RETOUR
Le déroulement du programme principal se fait de la manière suivante :
– A prend la valeur 5
– la procédure AFFIC affiche 5 sur l'écran
– A prend la valeur 10
– la valeur 10 est affichée
– le programme est terminé.





## V.7 INSTRUCTION RETOUR EN

Cette instruction permet de forcer le retour à un endroit donné du programme.

***Exemple :***

```
1
10 AFFICHER 'Ce programme montre une uti
lisation de  RETOUR EN'
20 &BRAN(@10,@900)
30 AFFICHER 'Cette instruction permet de
 préciser un numéro de ligne de RETOUR'
40 &BRAN(@10,@900)
900 AFFICHER 'C''est fini !';TERMINER
10000 PROCEDURE &BRAN(DEB,FINI) LOCAL FI
NI,DEB,C
10005 CHAINE C
10010 AFFICHER 'Tapez "S" pour passer à
la suite,'
10020 AFFICHER 'ou "D" pour reprendre au
 début,'
10030 AFFICHER 'ou "T" pour terminer.',1
0 13.
10040 LIRE C;C←TMA(C)
10050 SI C='S' ALORS RETOUR
10060 SI C='T' ALORS RETOUR EN FINI SINO
N SI C='D' ALORS RETOUR EN DEB SINON ALL
ER EN 10040
```

```
exECUTER A PARTIR DE 1

Ce programme montre une utilisation de
RETOUR EN
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
S
Cette instruction permet de préciser un
numéro de ligne de RETOUR
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
d
Ce programme montre une utilisation de
RETOUR EN
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
s
Cette instruction permet de préciser un
numéro de ligne de RETOUR
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
s
C'est fini !
TERMINE EN LIGNE 900
```

## V.8 INSTRUCTIONS RÉSULTAT

RESULTAT ex
ex est une expression qui sera rendue au programme appelant et qu'il pourra afficher, affecter à une variable ou utiliser dans un calcul.
En dehors de cette valeur rendue, l'instruction RESULTAT réalise les mêmes opérations que l'instruction RETOUR.





***Exemple :***

En L.S.E. il n'existe pas de fonction permettant d'explorer une chaîne de la droite vers la gauche : on peut enrichir ses possibilités à l'aide de la procédure suivante qui extrait les derniers caractères d'une chaîne :

```
1000 PROCEDURE &FCH(C,N) LOCAL N,C
1010 RESULTAT SCH(C,LGR(C)-N+1,'')


?&FCH('Bonjour Madame',5)
adame
```

## V.9 PROCÉDURES BINAIRES

On désigne par ce terme des procédures en langage machine que l'on utilise comme les procédures L.S.E.
La manière d'écrire de telles procédures est décrite dans l'Annexe V.

Une procédure binaire est désignée par son nom qui respecte la syntaxe des noms L.S.E. S'il s'agit d'une procédure fonction, seul le nom la distingue des fonctions du L.S.E.

## V.10 PROCÉDURES EXTERNES

Une procédure est dite interne si elle est déclarée dans le programme. Dans le cas contraire, elle est considérée par le L.S.E. comme étant externe : elle est recherchée comme telle sur le disque de travail puis lancée. Ceci concerne aussi bien les procédures L.S.E. que les procédures binaires. Une erreur est détectée si L.S.E. ne la trouve pas.

Le temps de chargement d'une procédure externe est loin d'être négligeable. Afin d'optimiser ce temps, il a été décidé de conserver en mémoire une procédure externe inactive dans le cas où l'on ne manque pas de place.
Lors d'un appel de procédure L.S.E., l'interpréteur :
– recherche la procédure dans les procédures internes L.S.E.,

– s'il ne la trouve pas, il poursuit la recherche dans les procédures externes L.S.E. déjà chargées,
– s'il ne la trouve pas, il la recherche dans les procédures L.S.E. du disque de travail. Trois cas peuvent se présenter :
  – la procédure n'existe pas sur le disque, il y a dans ce cas ERREUR E 18
  – la procédure existe et il y a en mémoire assez de place pour la charger et la lancer, ce qui est fait.
  – la procédure existe mais il n'y a pas assez de place, alors il est décidé de supprimer les procédures externes chargées et inactives jusqu'à ce qu'il y ait assez de place. Deux cas se présentent :
    – l'opération est possible, ce qui est fait.
    – l'opération s'avère impossible après suppression de toutes les procédures externes inactives, alors il y a ERREUR E 3.

Cette suppression éventuelle commence par les procédures inactives chargées depuis le plus de temps. Cela permet par exemple de se débarasser d'une procédure d'initialisation qui ne sert plus.

Le fichier contenant une procédure externe doit :
– avoir pour nom de la procédure externe L.S.E.
– contenir une procédure L.S.E. ayant ce même nom.

En dehors de cela, le fichier peut contenir un véritable programme L.S.E. et pouvoir être utilisé indépendamment de l'usage en procédure externe. Cependant, il est préférable, pour des raisons d'encombrement mémoire et de portabilité par rapport à d'autres implémentations du L.S.E., de ne pas mettre dans ce fichier autre chose que la procédure et de déclarer celle-ci en ligne 1.

Notons de plus que toutes les variables autres que les paramètres d'une procédure externe sont systématiquement des variables locales et qu'une telle procédure ne peut donc avoir accès aux variables du programme appelant.

## V.11 LES PROCEDURES ET LE CALBUR

Il n'est pas possible de déclarer des procédures en CALBUR.

L'utilisation de procédures en CALBUR est possible, qu'il s'agisse de procédures L.S.E. ou de procédures binaires, qu'il s'agisse de procédures internes ou externes.

Néanmoins la rencontre dans une procédure L.S.E. de certaines instructions interdites en CALBUR provoquera une erreur d'exécution. Cela concerne par





exemple l'instruction PAUSE, qui, si elle était autorisée, conduirait à un dilemme lors de la commande COntinuer. D'autres instructions, telles ALLER EN ne sont pas concernées par cette restriction.

Lors d'une erreur d'exécution dans une procédure utilisée en CALBUR la pile d'exécution est remise dans l'état où elle était avant l'appel. L'exécution d'une procédure en mode CALBUR ne permet donc pas sa mise au point.

## V.12 LE L.S.E. ET LA RÉCURSIVITÉ

La récursivité simple est la possibilité de réaliser des procédures qui s'appelent elles-mêmes. La récursivité croisée concerne des procédures s'appelant les unes les autres de façon qu'indirectement la procédure appelante soit appelée par une autre.

Le L.S.E. permet la récursivité simple et croisée sans aucune limitation (si ce n'est celle de la place mémoire).

Voici 2 exemples d'utilisation du L.S.E. en récursivité simple :

```
1 * Calcul de n! (Factorielle du nombre
n: produit des entiers inférieurs ou éga
ux à n).
10 AFFICHER 'Factorielle de ? ';LIRE N;S
I N<1 OU ENT(N)≠N ALORS ALLER EN 10
20 AFFICHER &FACT(N);TERMINER
30
1000 PROCEDURE &FACT(X) LOCAL X
1010 RESULTAT SI X=1 ALORS 1 SINON X*&FA
CT(X-1)

exECUTER A PARTIR DE 1

Factorielle de ? 5
120
TERMINE EN LIGNE 20
```

```
1 * Tours de Hanoï
10 AFFICHER ['Tour de départ numéro 1',/
,'Tour d''arrivée numéro 3',/]
20 LIRE ['Combien de disques à déplacer
? ',U,/]N;SI N<1 OU ENT(N)≠N ALORS ALLER
 EN 20
30 &HANOI(1,3,N);TERMINER
40
1000 PROCEDURE &HANOI(DEP,ARR,NBRE) LOCA
L NBRE,ARR,DEP
1010 SI NBRE=1 ALORS DEBUT &DEPL(DEP,ARR
);RETOUR FIN
1020 &HANOI(DEP,6-DEP-ARR,NBRE-1);&DEPL(
DEP,ARR);&HANOI(6-DEP-ARR,ARR,NBRE-1)
1030 RETOUR
1040
2000 PROCEDURE &DEPL(D,A) LOCAL A,D
2010 AFFICHER 'Déplacez un disque de ',D
,' en ',A;RETOUR

exECUTER A PARTIR DE 1
Tour de départ numéro 1
Tour d'arrivée numéro 3
Combien de disques à déplacer ? 3

Déplacez un disque de 1  en 3
Déplacez un disque de 1  en 2
Déplacez un disque de 3  en 2
Déplacez un disque de 1  en 3
Déplacez un disque de 2  en 1
Déplacez un disque de 2  en 3
Déplacez un disque de 1  en 3
TERMINE EN LIGNE 30
```





## V.13 PARAMÈTRES NOMS DE PROCÉDURE

Il peut être intéressant de réaliser une procédure pouvant prendre en compte, indifféremment, le traitement effectué par d'autres procédures : On passe à cette procédure, le nom d'une procédure à prendre en compte dans le traitement. Voici un exemple d'un tel programme :

```
1
10 &REPET(&BONJ,3)
20 &REPET(&AURE,5)
30 TERMINER
40
100 PROCEDURE &REPET(&P,X) LOCAL X,I
110 FAIRE 110 POUR I←1 JUSQUA X;&P()
120 RETOUR
130
140 PROCEDURE &BONJ();AFFICHER 'Bonjour'
;RETOUR
150
160 PROCEDURE &AURE();AFFICHER 'Au revoi
r';RETOUR

exECUTER A PARTIR DE 1

Bonjour
Bonjour
Bonjour
Au revoir
Au revoir
Au revoir
Au revoir
Au revoir
TERMINE EN LIGNE 30
```