MASTER EN SCIENCES INFORMATIQUES

Un poste de travail statistique sur micro ordinateur

Cavez, Jean-Jacques

Award date:
1984

Awarding institution:
Universite de Namur

Année académique 1983-1984

Cavez.

Un poste de travail statistique sur micro ordinateur

Mémoire présenté pour l'obtention du grade
de Licencié et Maître en Informatique
par

Cavez Jean-Jacques

Je tiens à remercier tous ceux qui m'ont apporté un soutien, moral ou matériel, durant toute l'année d'élaboration de ce mémoire; et plus spécialement, les professeurs Jean-Luc Hainaut et Eugène Schifflers, qui m'ont consacré une large part de leur temps.

Je remercie également mon épouse, qui m'a aidé pour la mise en page de ce volume et m'a soutenu dans les moments difficiles.

A Vincent, mon fils, et à Martine, mon épouse.

but et méthodologie du travail

* Ce mémoire est destiné à préparer la réalisation d'un poste de travail statistique. Nous ne voulons dès lors créer aucun algorithme statistique, ni même en implémenter .... Ce que nous voulons faire, c'est dégager progressivement les concepts nécessaires à la mise en oeuvre d'un système informatique global qui puisse aider le statisticien tout au long des différentes phases de son travail.

* Pour réaliser cette étude, nous sommes partis de l'existant, c'est-à-dire des logiciels déjà disponibles, ainsi que de cas pratiques de travaux statistiques. Nous avons bien sûr profité de l'expérience de divers professionnels de la branche pour nous guider. Toujours avec l'accord de ceux-ci, nous avons alors dégagé les concepts propres à la statistique, ainsi que les propriétés désirées. L'étape suivante est de spécifier les concepts informatiques adéquats. Ayant une base solide, nous pouvons alors contruire une architecture logicielle théorique qui intègre ces concepts et qui détermine les relations entre composants. La dernière phase est de définir une architecture concrète (une machine, un O.S., etc... étant choisis) et ensuite, limités par le temps et la taille du mémoire, de sélectionner et réaliser un noyau.

plan

Chapitre premier

Introduction : La démarche d'analyse statistique

Chapitre deuxième

Caractéristiques fonctionnelles d'un poste de travail statistique

Chapitre troisième

Spécifications d'un poste de travail statistique

§ 3.3 Les instruments particuliers ...

3.3.1 De calcul

3.3.2 De présentation

3.3.3 D'édition de rapports

Chapitre quatrième

Architecture théorique

Chapitre cinquième

Architecture concrète

§ 5.1 Matériel de base

§ 5.2 Architecture

Chapitre sixième

Définition d'un noyau

§ 6.1 Introduction

§ 6.2 Spécifications

Chapitre septième

Réalisation du noyau

§ 7.1 Structure du programme

§ 7.2 Algorithmes

§ 7.3 Mode d'emploi du programme

Bibliographie

1. The common sense of object oriented languages.
Roy S.Freedman
COMPUTER DESIGN, february 1983, 111-118

2. A step beyond programming languages.
Ben Shneiderman
University of Maryland
COMPUTER, August 1983, 57-69

3. Can computers really be friendly ?
Jack W. Crenshaw and Joseph Philipose
COMPUTER DESIGN, february 1983, 103-107

4. Integrated data analysis and management
for the problem solving environment.
R. Erbe, R. Hartwig, H. Lehmann, G. Mueller and U. Schauer
IBM Scientific Center, Heidelberg
Inform. Systems, vol 5, 273-285

5. Computer Usage for Social Scientists
Douglas M.Klieger
Villanova University
Allyn and Bacon, Inc., 3-4,25-41,123,189,249

6. A report on the Accuracy of Some Widely Used
Least Squares Computer Programs
Roy H.Wampler
J.A.S.A June 1970, volume 65, number 330

7. An Appraisal of Least Squares Programs for the Electronic Computer
from the Point of View of the User
James W.Longley
American Statistical Association Journal, september 1967

8. A Comparison of Some Algorithms for the Nonlinear Least Squares
Problem
Håkan Ramsin and Per-Åke Wedin
BIT 17 (1977), 72-90

9. The Role of Computers in the Teaching of Statistics
A.Ronald Gallant
North Carolina State University
Discussion paper, American Statistical Association Annuel Meeting,
Boston, Massachusetts, August 1976

10. Comparison of Several Algorithms for Computation of Means, Standard Deviations and Correlation Coefficients.
Peter M.Neely
University of Chicago, Chicago, Illinois.
Communications of the ACM volume 9/number 7/July, 1966

Annexes

A Présentation de matériel existant

A1. STAR
A2. LISA
A3. SMALLTALK
A4. VISICALC
A5. VISIPLOT
A6. LISACALC
A7. LISADRAW
A8. LISAGRAPH
A9. LISAWRITE
A10. DB MASTER
A11. LOTUS 1-2-3
A12. EVERYMAN
A13. SPEEDSTAT
A14. STATPRO
A15. ELF, TWG/ARIMA
A16. HEWLETT-PACKARD
A17. BMDP
A18. GLIM
A19. SAS
A20. SPSS

B Exemples de travaux statistiques

C Documents annexes au chapitre premier

C1. Echantillon "mesure du poids de douze souris".
C2. Exemples de graphiques d'exploration des données.
C3. Graphiques "accroissement de poids en fonction du dosage".

D Texte du programme en Pascal réalisant le noyau

Chapitre premier

Introduction: La démarche d'analyse statistique

Le statisticien, qui désire utiliser un ordinateur pour son travail d'analyse, est confronté à un certain nombre de problèmes. Pour préciser ceux-ci, nous exposerons dans ce chapitre trois cas, choisis parmi d'autres, soumis au statisticien, et, six logiciels courants proposés à son usage. Nous tirerons alors quelques conclusions sur l'adéquation des logiciels aux problèmes rencontrés.

En annexe, se trouvent quelques exemples supplémentaires de travaux statistiques.

Felix qui potuit rerum cognoscere causas
<Géorgiques (II,489)>

§ 1.1 Trois types de problèmes

Il faut d'abord préciser que nous ne ferons pas ici de remarques quant à la démarche intellectuelle (les tests sont-ils biaisés ? , l'échantillon est-il représentatif ? , etc ...), mais que nous nous intéresserons plus spécialement à la démarche "physique" (display, manipulation de données, etc...), cette dernière pouvant être facilitée par l'usage de l'ordinateur.

1.1.1 La croissance des souris

1.1.1.1 Données disponibles et buts de l'analyse

Au départ de ce problème, nous disposons de douze courbes de croissances individuelles, correspondant à cent vingt mesures quotidiennes de poids de souris (six mâles et de six femelles); mesures prises avec les précautions expérimentales d'usage.

Nous disposons également du poids à la naissance et d'un numéro de code identifiant pour chaque animal. [voir annexe C1]

Le but du travail sera ici de mieux connaître les mécanismes de la croissance (influence du sexe, du passage à l'état adulte, etc...)

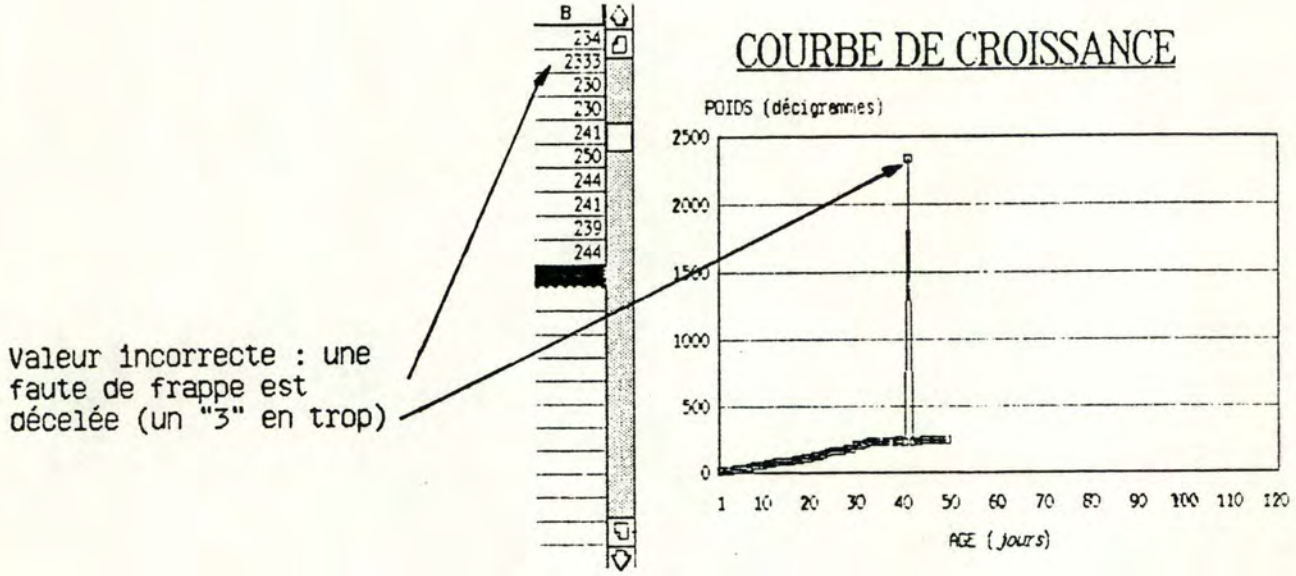
1.1.1.2 Démarche et problèmes de l'analyse

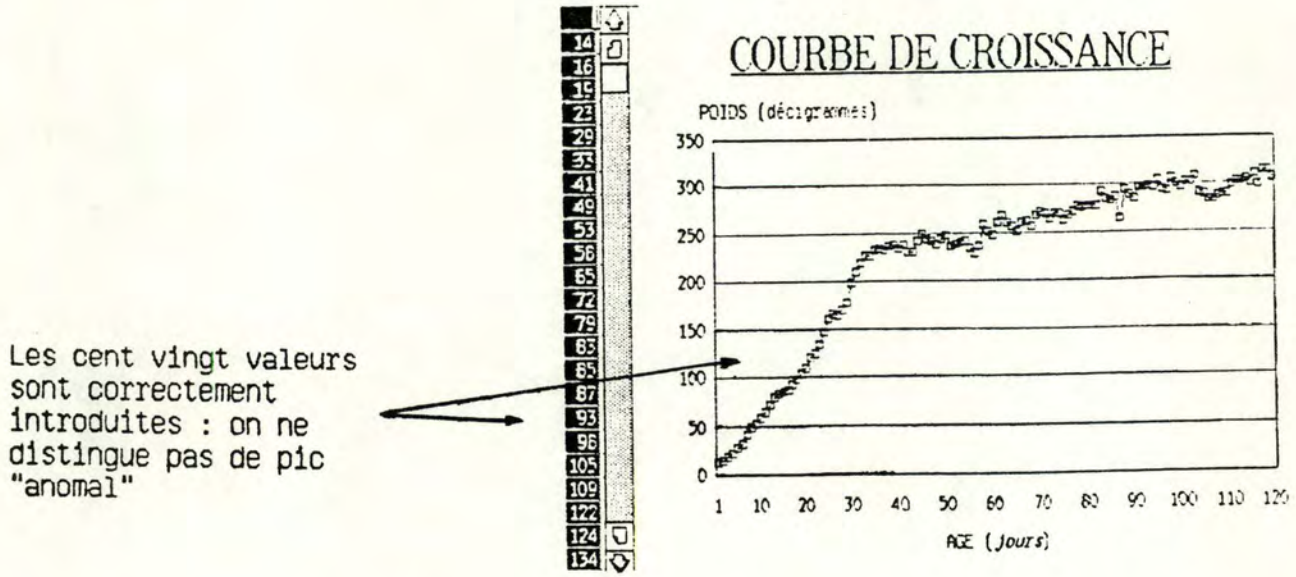
La première chose à faire, c'est d'introduire les données. Nous rappelons que dans ce cas-ci, il y a mille quatre cent soixante quatre valeurs à introduire, il faut donc tenir compte de l'aspect humain dans la conception d'un système de saisie de données si on veut minimiser le nombre d'erreurs dues à la fatigue de l'opérateur.

Lors de la seconde étape, débute le travail plus proprement statistique. Nous allons commencer par une petite étude "à vue de nez" des chiffres pour traiter les données aberrantes (par exemple poids trop petits

ou trop élevés, données manquantes , etc ...). Pour cela, on peut simplement faire voyager une "fenêtre" dans le tableau de chiffres, de façon à pouvoir le relire, mais on peut aussi demander d'afficher le graphique correspondant aux valeurs saisies pour déceler des pics ou des creux anormaux.

On peut également utiliser directement une saisie "sous graphique" pour combiner les deux premières étapes.





La troisième étape consiste en l'exploration des données. Dans notre cas, il s'agit de l'élaboration d'une série de graphiques. Nous pouvons notamment citer :

- Graphique séparé pour chaque souris du poids, du poids avec moyennes mobiles, de l'accroissement du poids, du logarithme de l'accroissement en fonction de l'âge.

- Graphique qui superpose les graphiques créés séparément (pour dégager une tendance générale)

Il faut remarquer que cette étape comporte deux parties : le calcul de valeurs à partir des poids mesurés, et l'affichage sous forme graphique de ces valeurs. [voir annexe C2]

Suite aux observations de l'étape d'exploration des données, nous avons décelé la présence de deux tendances dans les courbes de croissances. Ces tendances déterminent deux périodes : de zéro à quarante jours, et après quarante jours. Après réflexion, cette cassure est en relation avec le passage à l'âge adulte de la souris.

Nous allons donc étudier notre échantillon en le séparant en deux parties: Enfance et Maturité.

On peut itérer sur la troisième étape (exploration), c'est-à-dire séparer dans notre échantillon les mâles et les femelles car, le cycle menstruel a une influence sur le poids etc ….

Les étapes suivantes consistent en l'ajustement d'un modèle aux données. Un modèle pourrait décrire le poids à l'âge **a** comme fonction de **a** et d'un autre paramètre **b**. $W(\mathbf{a}) = H(\mathbf{a}, \mathbf{b}) + E(\mathbf{a})$. Cette fonction H peut être linéaire en les paramètres ou non, ce qui a des implications tant mathématiques qu'informatiques. Si H est linéaire, les algorithmes seront généralement peu complexes alors que si H est non linéaire, nous devrons recourir à des algorithmes d'optimisation etc … Ramsin en analyse quelques uns [10]. Il existe bien sûr d'autres modèles, tels les multivariés qui travaillent sur les données brutes ou sur les paramètres estimés, et qui

impliquent d'importants calculs ainsi que des problèmes de choix heuristiques.

Encore une fois, on aura la dualité calculs mathématiques / représentations graphiques, et nous rappelons qu'il s'agit d'un travail de recherche, c'est-à-dire qu'on procède en partie par essais et erreurs.

1.1.2 Les gorets piqués

1.1.2.1 Données disponibles et buts de l'analyse

Nous nous situons ici dans le cadre des techniques d'élevage porcin. Le statisticien désire connaître l'influence de l'injection d'une substance (par exemple une hormone) sur l'augmentation du poids des porcs.

Il dispose pour son étude de 16 valeurs:

$$X_i ; \quad i = 1, 4$$

$$Y_{ij} ; \quad i = 1, 4 ; \; j = 1, 3$$

où Y_{ij} est l'accroissement de poids du $j^{ième}$ goret qui est soumis à la piqûre dosée à X_i mg de la substance étudiée. (chaque goret est piqué une seule fois.)

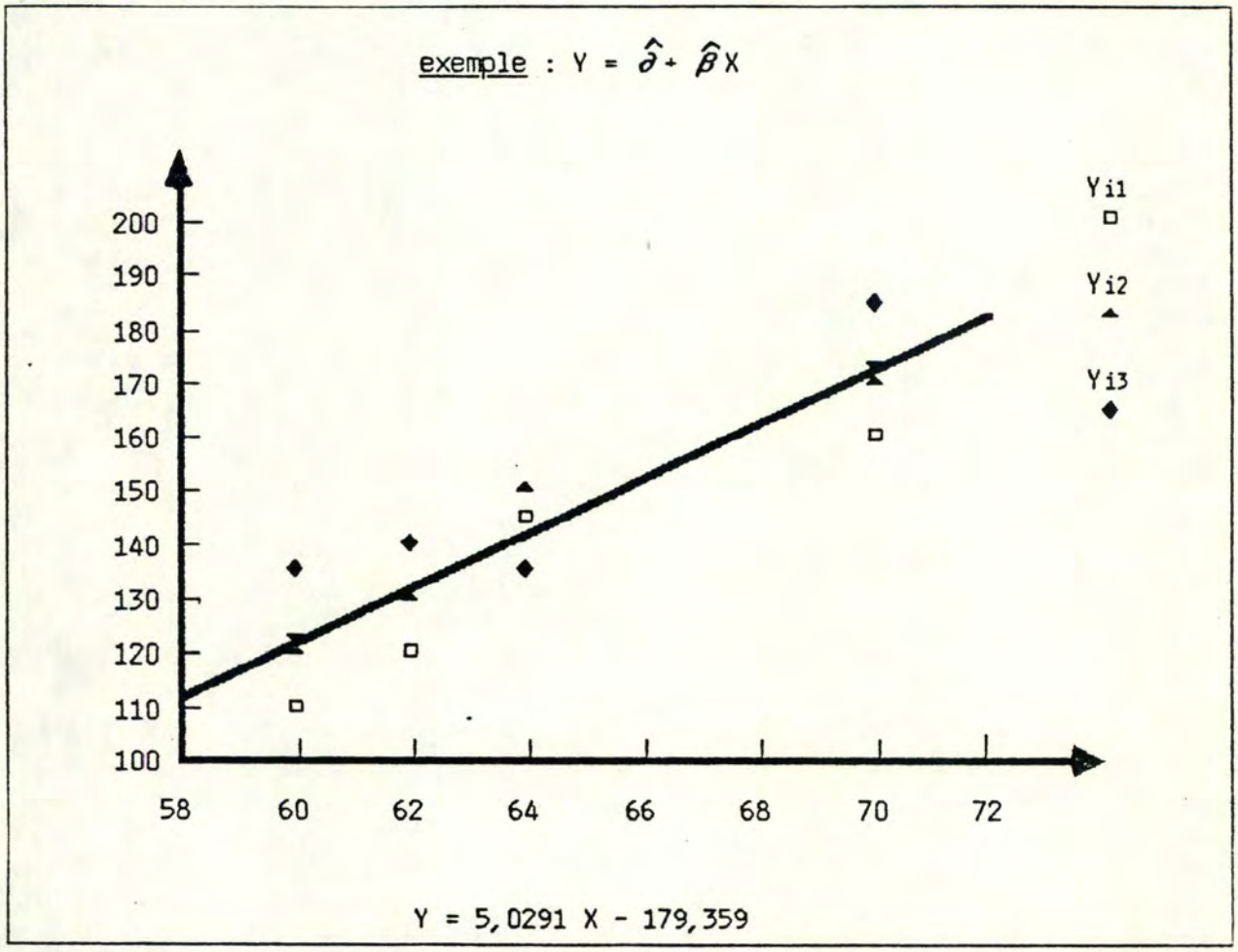
X_i	60	62	64	70
Y_{ij}	110	120	145	160
	120	130	150	170
	135	140	135	185

Le but de l'analyse sera ici de trouver un modèle linéaire suffisamment adéquat pour les données.

1.1.2.2 Démarche et problèmes de l'analyse

Après avoir saisi les données, le statisticien commence l'exploration au moyen de divers graphiques : augmentation du poids en fonction de la dose, avec différentes échelles : arithmétiques, logarithmique, échelle - 1/A+X, etc .... Il doit pouvoir passer facilement d'une échelle à une autre et voir évoluer les graphiques.

Ensuite, le statisticien peut, par exemple, exécuter la régression de Y_{ij} sur X_i : à partir de $Y = X\beta$, il passe à $\hat{\beta} = X^+ Y$ et obtient le graphe des droites $Y_j = \hat{\partial}_j + \hat{\beta}_j X$



Il peut procéder de la même manière avec $Y = \overline{Y_j}$ (moyenne de Y_{ij}, sur l'indice j) pour obtenir une droite $Y = \hat{\partial} + \hat{\beta} X$, et calculer les intervalles de confiance pour les $\hat{\partial}$ et $\hat{\beta}$. Il pourrait aussi rechercher des intervalles de prédiction pour $Y_0 = \partial X_0 + \beta_0$.

Le statisticien réalisera également quelques tests d'hypothèse (par exemple $\partial = \partial_0$).

Le dernier stade étant de pratiquer l'inférence statistique.

1.1.3 Des lézards sous les bananiers

1.1.3.1 Données disponibles et buts de l'analyse

Nous disposons d'une table de contingence à trois entrées qui comptabilise "Le nombre de fois qu'une espèce de lézard se promène sous une variété de bananiers, à une heure déterminée de la journée". On considère quatre espèces de lézards, deux variétés de bananiers et vingt quatre heures dans la journée.

Le but du jeu est de déterminer s'il y a ou non dépendance entre les facteurs lézard, bananier, heure; et si oui, d'établir un modèle qui décrit ces dépendances.

1.1.3.2 Démarche et problèmes de l'analyse

Suite à l'étape d'introduction des données, nous pourrions créer des graphiques en trois dimensions, car cela peut nous donner des indications.

Toutefois nous ne le ferons pas ici, et on ne le fait généralement pas car il est souvent difficile de visualiser la troisième dimension dans le plan (illustration en annexe A16); de toute façon , il est inutile de faire des graphiques avec un nombre de dimensions plus élevé que trois, étant donné les problèmes que l'on éprouve pour y distinguer quelque chose.

Nous allons faire des calculs de diverses valeurs :

- $m_{ijk} = E[n_{ijk}]$
- $N = n_{+++}$
- etc...

Nous allons essayer divers types de modèles log linéaires hiérarchisés, comme par exemple :

$$\log \hat{m}_{ijk} = \mu + \mu_{1(i)} + \mu_{2(i)} + \mu_{3(k)} + \mu_{12(i,j)}$$

Pour cela il faut :

-i- ajuster le modèle aux données par IPF -> $\hat{m}_{ijk}$

-ii- calculer le Chi-deux d'ajustement ou le G^2 (statistique du quotient de vraisemblance)

-iii- calculer les estimées si le modèle est suffisamment adéquat

Nous pouvons retenir un modèle et transformer les données (effondrement de la table, partition). L'effondrement est nécessaire quand nous constatons que certaines informations recueillies n'entrent pas en ligne de compte dans le modèle, nous pouvons alors les éliminer et ainsi réduire le nombre de dimensions de notre problème. Dans l'exemple des lézards, on peut aussi constater un clivage entre les heures du jour et de la nuit. Pour chacune de ces deux périodes nous disposons d'un modèle adéquat, on partitionne donc l'échantillon en deux parties suivant les périodes considérées.

§ 1.2 Trois logiciels sur gros ordinateur

Il existe un certain nombre de logiciels fréquemment utilisés, mais ils sont "complexes, denses, inaccessibles à l'utilisateur potentiel" ... comme l'affirme avec vigueur Klieger [5]. Ils rendent l'usage de l'ordinateur difficile et parviennent à mystifier le chercheur.

1.2.1 BMDP

1.2.1.1 Possibilités

Nous n'allons pas détailler toutes les possibilités de ce logiciel, pour cela il suffit de consulter l'annexe A17. Retenons simplement qu'il s'agit d'un ensemble de programmes pour diverses descriptions et analyses statistiques. Le tout permettant de résoudre certains problèmes statistiques, qui nécessitent des calculs particuliers, notamment dans le domaine bio-médical.

1.2.1.2 Marche à suivre

Dans l'annexe A17, se trouve un résumé de la méthode d'utilisation de BMDP ainsi qu'un exemple concret. Nous pouvons dire que l'interface est loin d'être attrayante : il s'agit d'une série de commandes fortement structurées et admettant peu de souplesse. Par exemple, on établit une distinction entre les lettres majuscules et les lettres minuscules; quand on demande huit caractères blancs, il ne faut pas en taper plus ou moins ! Il ne faut pas oublier les points à la fin des lignes, on ne peut pas inverser deux "cartes" (alors qu'il n'y a aucun problème pour le processeur, car il a tous les paramètres "en main"). Si un problème survient, aucun résultat n'est fourni (alors qu'on pourrait par exemple fournir les résultats pour toutes les opérations précédant l'incident, de façon à permettre une reprise sans devoir tout recommencer).

1.2.2 SPSS

1.2.2.1 Possibilités

L'annexe A20 précise de manière concise les possibilités de SPSS, il faut savoir que SPSS est un gros programme de description et d'analyses statistiques, de la même philosophie que BMDP, mais conçu pour des chercheurs en sciences sociales.

1.2.2.2 Marche à suivre

L'annexe A20 comporte le détail des opérations. De même que pour BMDP, on travaille par concept de cartes, l'interface est fort rigide ... .Nous pouvons dire que ce logiciel (comme le précédent) est, quoique fort répandu et utilisé, d'un abord désagréable à cause de sa conception obsolète de l'outil informatique. (Il date d'ailleurs de 1965.)

De plus , un défaut particulier à ce logiciel : la gourmandise en place mémoire nécessaire à son fonctionnement, ceci se traduit par exemple par la nécessité de travailler en "stall" sur le DEC 20/60 des Facultés de Namur.

Example 7.1 demonstrates the *minimum* cards required to perform a single task (in this case FREQUENCIES) on a file containing no subfile structure and input from cards.

```
1                16
VARIABLE LIST    AGE,SEX,RACE,INCOME,EDUCATN
INPUT MEDIUM     CARD
N OF CASES       10
INPUT FORMAT     FIXED (5X,F2.0,2F1.0,F7.0,1X,F1.0)
FREQUENCIES      GENERAL=AGE TO EDUCATN
OPTIONS          3
STATISTICS       ALL
READ INPUT DATA
     3011   9000 2
     3113   7500 1
     4022   8300 2
     2713  12500 5
     5411  13250 4
     3512  10150 5
     2523   9500 3
     5011  17500 3
     4211   8500 1
     3222   9500 2
FINISH
```

EXAMPLE 7.1

1.2.3 GLIM

1.2.3.1 Possibilités

Il s'agit d'un logiciel interactif pour l'analyse statistique par des modèles "linéaires généralisés". On voit donc que les possibilités sont plus spécifiques que pour BMDP ou SPSS.

1.2.3.2 Marche à suivre

Quoique gardant un certain manque de souplesse, et une grande pauvreté graphique, GLIM est certainement plus agréable à utiliser que les deux logiciels précédents.

L'annexe A18 est constituée d'une série d'exemples d'utilisation. Nous constatons notamment la facilité avec laquelle une erreur peut être récupérée, et comment dynamiquement nous pouvons orienter nos recherches dans l'une ou l'autre direction.

```
Glim>$unit 20
Glim>$data x y
Glim>$dinput 30
Glim>$yvar y
Glim>$fit y ^H$K^H$K^H$K^H$K^H$K^H$Kplot y x
Glim>$fit x$dis e r
   7.200       *
   6.900       *
   6.600       *
   6.300       *y
   6.000       *
   5.700       *  y
   5.400       *
   5.100       *   y
   4.800       *
   4.500       *
   4.200       *      y
   3.900       *       2
   3.600       *
   3.300       *          3y
   3.000       *
   2.700       *
   2.400       *                y y
   2.100       *                   y  y y y
   1.800       *                            y   y       y
   1.500       *
   1.200       *
^3.....*.....*.....*.....*.....*.....*.....*.....*.....*.....*.....*
      0.000E+00  20.0       40.0       60.0       80.0       100.       120.
```

§ 1.3 Trois logiciels sur micro ordinateur

Les logiciels sur micro ordinateur sont actuellement d'un niveau mathématique très bas. Leur attrait tient surtout à l'usage du graphisme et à la disponibilité inhérente au micro ordinateur. Les causes principales en sont la limitation de la puissance de calcul et la faible capacité de stockage.

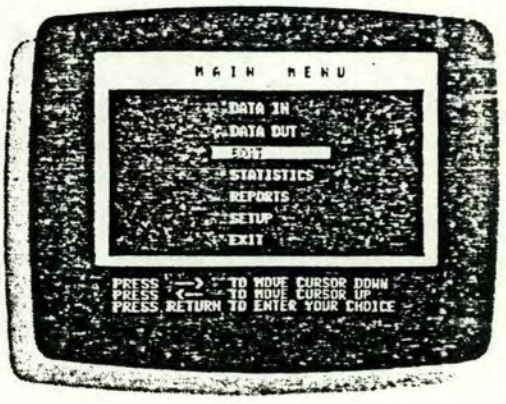
1.3.1 Speed Stat

1.3.1.1 Possibilités

Une documentation précise sur le logiciel est donnée en annexe A13. L'analyse de ces documents nous fait remarquer que les possibilités de celui-ci sont réduites par rapport à des produits tels que BMDP ou SPSS. (par exemple, on ne trouve aucune statistique descriptive en présentation graphique, rien en non linéaire, rien en non paramétrique, etc...)

1.3.1.2 Marche à suivre

L'interface est relativement agréable, en ce sens qu'on présente les menus, et qu'on choisit de manière aisée dans ce menu.



Les rapports aussi sont plus soignés que pour les précédents logiciels, mais de nouveau, on se trouve dans un système très rigide, où aucune liberté n'est admise. En conclusion, il s'agit d'une transposition, en réduction, des programmes classiques, en gardant la plupart des défauts.

```
                    SPEEDSTAT FREQUENCIES REPORT

PROJECT NAME: SPEEDSTAT SAMPLE REPORTS
REPORT TITLE: SAMPLE FREQUENCIES REPORT
REPORT DATE : 4 OCTOBER 1982
RESEARCHER  : SHAFFER & SHAFFER APPLIED P & D

COMMENTS    : - THIS IS A DEMONSTRATION OF THE SPEEDSTAT
                FREQUENCIES REPORT.
              - THE DATA USED TO PREPARE THIS REPORT IS FROM
                THE SAMPLE DATA SET PROVIDED WITH THE SYSTEM.

.....

*** TOY ***
-----------
VALUE LABEL              VALUE  FREQUENCY   FREQ %   CUMULATIVE FREQ %
-------------------------------------------------------------------
TOY #1          ..         1        3      21.4 %      21.4 %
TOY #2          ..         2        7      50.0 %      71.4 %
TOY #3          ..         3        4      28.6 %     100.0 %
-------------------------------------------------------------------
                      TOTAL     14        100.0 %     100.0 %

                MISCELLANEOUS STATISTICS
                ------------------------
MEAN            ..............      2.071
MEDIAN          ..............      2.071
MODE            ..............          2
STD DEV         ..............       .73
VARIANCE        ..............       .533
STD ERR         ..............       .195
KURTOSIS        ..............      -.856
SKEWNESS        ..............      -.119
RANGE           ..............          2
MINIMUM         ..............          1
MAXIMUM         ..............          3

VALID CASES     ..............         14
MISSING CASES   ..............          0

                      [ PAGE 1 ]
```

```
                    SPEEDSTAT FREQUENCIES REPORT

.....

*** SEX ***
-----------
VALUE LABEL              VALUE  FREQUENCY   FREQ %   CUMULATIVE FREQ %
-------------------------------------------------------------------
MALE            ..         M        7      50.0 %      50.0 %
FEMALE          ..         F        7      50.0 %     100.0 %
-------------------------------------------------------------------
                      TOTAL     14        100.0 %     100.0 %

                MISCELLANEOUS STATISTICS
                ------------------------
MEAN            ..............       N/A
MEDIAN          ..............       N/A
MODE            ..............       N/A
STD DEV         ..............       N/A
VARIANCE        ..............       N/A
STD ERR         ..............       N/A
KURTOSIS        ..............       N/A
SKEWNESS        ..............       N/A
RANGE           ..............       N/A
MINIMUM         ..............         F
MAXIMUM         ..............         M

VALID CASES     ..............        14
MISSING CASES   ..............         0

                      [ PAGE 2 ]
```

1.3.2 Statpro

1.3.2.1 Possibilités

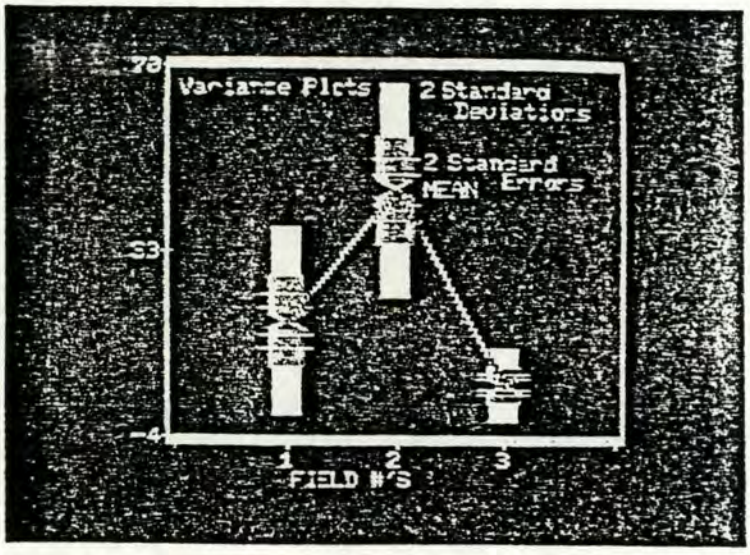
L'annexe A14 précise les différentes possibilités de Statpro.

Nous constatons que tout en offrant plus que Speedstat, Statpro n'atteint pas encore la puissance des "gros" logiciels.

Nous voyons également qu'il y a un embryon de base de données, mais que c'est surtout l'aspect graphique qui a été soigné (on peut notamment obtenir des effets en couleur).

1.3.2.2 Marche à suivre

La souplesse n'est pas encore au rendez-vous, en effet le système de menu est rigide, la présentation des rapports semble bien pauvre compte tenu des possibilités graphiques à l'écran. En bref, pour des analyses limitées, en interactif et pour des cas bien cernés, ce logiciel semble agréable … . L'erreur a sans doute été de trop miser sur le chatoyant et de ne pas assez pousser la réflexion mathématique.



Simulation of Forecasting 1 Period Ahead

Period	Y(t)	S(t)	Forecast	Error	Squares
Initial estimates		359.670			
1.00	365.000	359.717	359.670	5.330	5.429
2.00	381.000	360.142	359.717	21.283	452.982
3.00	317.000	359.279	360.142	-43.142	1861.26
4.00	297.000	358.024	359.279	-62.279	3878.72
5.00	399.000	358.853	358.024	40.966	1678.33
6.00	402.000	359.716	358.853	43.147	1861.65
7.00	375.000	360.022	359.716	15.284	233.567
8.00	349.000	359.811	360.022	-11.022	121.481
9.00	386.000	360.325	359.811	26.189	686.367
10.00	328.000	358.679	360.325	-32.325	1044.92
11.00	389.000	360.265	358.679	29.321	859.729
12.00	343.000	359.920	360.265	-17.265	298.090
13.00	276.000	358.242	359.920	-83.920	7042.56
14.00	334.000	357.757	358.242	-24.242	587.654
15.00	294.000	356.482	357.757	-63.757	1313.57
16.00	334.000	357.992	358.482	-24.482	599.250
17.00	384.000	358.511	357.992	26.008	676.416
18.00	314.000	357.422	358.511	-44.511	1981.33
19.00	344.000	357.350	357.422	-13.422	180.157
20.00	337.000	356.943	357.350	-20.350	414.103
21.00	345.000	356.704	356.943	-11.943	142.624
22.00	362.000	356.810	356.704	5.296	28.051
23.00	314.000	355.953	356.810	-42.810	1832.66
24.00	365.000	356.134	355.953	9.047	81.841

1.3.3 Hewlett-Packard logiciel

1.3.3.1 Possibilités

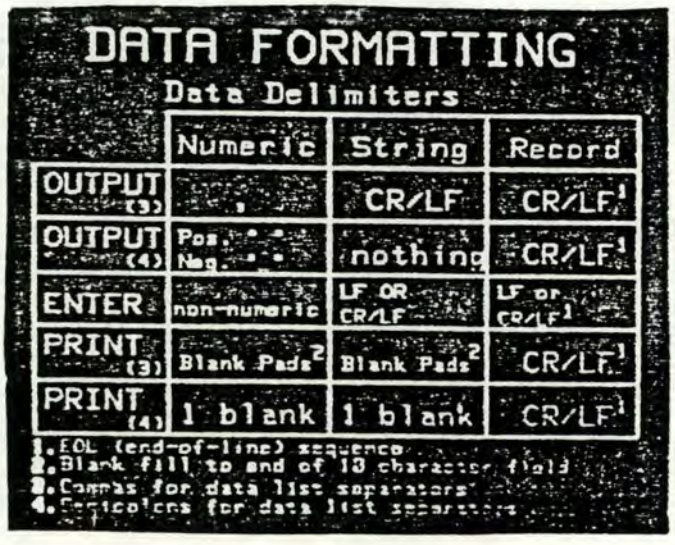
Nous renvoyons à l'annexe A16 pour des précisions complémentaires.

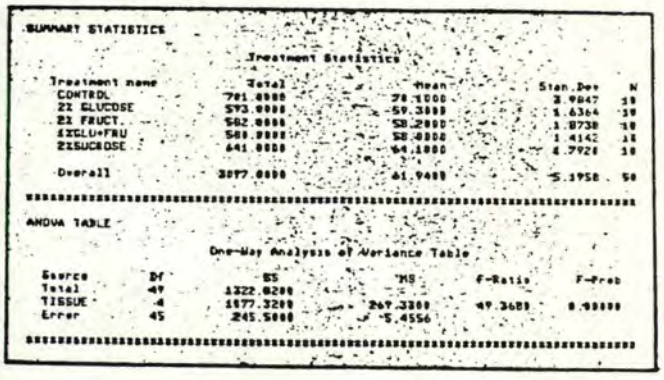
Nous constatons qu'il s'agit d'utilitaires classiques, présentés comme une librairie, ce qui est original par rapport à tous les systèmes fermés que nous avons rencontrés jusqu'à présent.

On peut notamment faire des manipulations des données, des statistiques de base, des statistiques générales, des graphiques, de la génération de nombres aléatoires.

1.3.3.2 Marche à suivre

L'introduction des données peut se faire très souplement : mémoire de masse, clavier, digitaliseur ....La souplesse d'utilisation se concrétise par la possibilité d'utiliser les routines sans devoir nécessairement créer un programme d'application. Nous pouvons toutefois regretter la présentation "classique" des écrans, qui n'est pas toujours très attrayante, surtout quand on dispose d'un instrument de travail de la qualité d'un HP série 200.





§ 1.4 Conclusions

1.4.1 Nature du problème statistique

La nature des problèmes statistiques varie beaucoup suivant les cas. Néanmoins nous pouvons dire que l'ordinateur peut jouer un rôle favorable dans la solution du problème, au moyen de calculs et aussi en définitive par la rapidité et la facilité de présentation "parlante", ainsi que par la possibilité d'utiliser différentes méthodes aisément. Gallant fait également remarquer le rôle pédagogique de l'ordinateur, qui permet de traiter des cas en vraie grandeur et donc de motiver l'étudiant [9].

1.4.2 Type de démarche

Même si la démarche varie d'un cas à l'autre, on peut isoler quelques principes assez généraux :

- une partie de recherche, fortement interactive

  * introduction/modification des données
  * présentations graphiques
  * petites fonctions simples
  * fonctions exigeant des calculs longs et fastidieux, mais systématiques et bien définis

=>exploration des données

- une partie de recherche en calculs

  * calculs longs

../..

* calculs courts mais

@ répétitifs avec de petites modifications de paramètres à chaque fois

@ variés (on essaie telle ou telle procédure)

=)ajustement d'un modèle

- une partie lourde en calculs

* tests de randomization

* simulations pseudo aléatoires

* etc ...

=)inférence

N'oublions pas non plus

- la dactylographie des rapports.

- la mise en page des résultats.

- l'élimination, l'archivage ou la récupération des données ou fonctions d'analyse.

- l'établissement de bibliographie.

- pour les adeptes de la théorie baysienne, il faut préalablement rechercher une loi à priori.

- en vue d'évaluer la puissance d'un test, on peut désirer la randomization d'une expérience, on fait alors de la simulation.

- on peut également faire de la simulation pour préplanifier une expérience, on doit alors pouvoir changer de scénario facilement. Les calculs sont lourds dans ce cas.

- Tout au long de sa démarche, le statisticien désire pouvoir faire en parallèle des choix aléatoires dans des listes, des calculs auxiliaires, des simulations.

1.4.3 Scénario habituel

Tout en nous gardant bien de vouloir systématiser ce qui ne peut l'être, nous dégageons un scénario qui se rencontre souvent, à quelques variantes près.

A Introduction des données

B Display & manipulation des données

C Calculs élaborés

D Bouclage sur les points B et C (*où les résultats de C servent de données à A et B et vice versa*)

E Elaboration d'un rapport

L'utilisateur désire souvent créer ses propres algorithmes ou utiliser des programmes qui se trouvent dans la littérature spécialisée . Nous pouvons donc ajouter, à cause du point C, qu'il faut prévoir un système ouvert. Par contre pour les points A et B nous pouvons concevoir un système sous forme de boîte noire, car les desideratas en cette matière sont voisins pour tout genre d'analyse, quelle que soit la "philosophie" du statisticien.

Nous pouvons dès à présent retenir certains concepts statistiques. L'échantillon, qui est la collection des données brutes est un premier concept, au chapitre deuxième nous en discuterons plus longuement. Cet échantillon subira des transformations diverses destinées à le rendre exploitable par d'autres méthodes, ou simplement pour obtenir un insight qui déclenchera un type particulier d'analyse. Un autre concept est celui de fonction d'analyse, une fonction d'analyse consiste en un traitement quelconque en vue d'obtenir un résultat, généralement numérique. Le plus souvent, on traite des données extraites soit directement de l'échantillon, soit des résultats d'une précédente fonction d'analyse.

Nous avons alors tout ce qui concerne la gestion de l'analyse statistique, la manipulation (création, élimination, archivage, récupération) de dossiers qui regroupent soit des échantillons, soit des outils d'analyse, soit des rapports et des notes au sujet d'une expérience.

Nous avons également des instruments tels que calculatrice, tables, revues, fichiers de références.

1.4.4 Desideratas auxiliaires

Ceci ne constitue pas un point primordial dans l'établissement d'un poste de travail statistique, pourtant, ce qui va suivre peut fortement soulager tout un aspect du travail d'analyse et de production d'un rapport.

Il serait agréable de disposer d'un moyen d'effectuer de petits calculs, et de consulter des tables, en parallèle avec l'analyse. Evidemment sans le recours à une calculette et un livre !

De plus, pour ceux qui ne disposent pas d'une secrétaire, ou qui ne font confiance qu'à eux-même, un éditeur de texte, un éditeur de dessins, un système de gestion des références bibliographiques, et autres ... seraient les bienvenus, à condition que tout cela soit "intégré", c'est-à-dire que ces différents outils puissent se communiquer des données. En clair, cela revient à dire que l'on puisse insérer un dessin dans une page de texte (ou l'inverse), que les résultats d'une analyse (effectuée au point C par exemple) puissent être insérés dans un texte, qu'un graphique (produit au point B par exemple) puisse être inséré dans un dessin, etc ....

1.4.5 Critique des outils étudiés

Suite à l'exposé de la démarche, des scénarios, des desideratas des statisticiens, nous pouvons maintenant critiquer de façon plus objective les logiciels étudiés dans le cadre du mémoire.

1.4.5.1 "Le pour"

Ces outils existent, on les utilise, et ainsi leur principale qualité est de pouvoir servir de référence lors de la parution d'articles scientifiques. En effet, les algorithmes sont connus, et il n'est donc pas nécessaire de créer ou de réécrire des algorithmes pour chaque cas présenté au public.

Quoique la qualité numérique de certains de ces logiciels ait été par le passé assez mauvaise pour certains problèmes particuliers (Wampler obtient des résultats où la moitié des chiffres significatifs sont erronnés), la progression des techniques de calcul [7] a permis d'améliorer la robustesse des programmes, et actuellement les utilisateurs leur accordent habituellement leur confiance. Pour les petits logiciels, on pourrait craindre une fiabilité plus douteuse, étant donné l'investissement moindre. Mais ceci n'est pas fondé, en effet les petits logiciels ne proposent souvent que des analyses très limitées, le risque d'erreurs est donc trivialement réduit.

Il faut également faire remarquer le coût généralement assez faible d'un logiciel sur gros ordinateur, on ne paie ordinairement que le prix de revient du matériel utilisé. Ceci peut être dû à ce que le coût réel (développement, maintenance) est de toute façon non récupérable car trop élevé, et que le seul bénéfice escompté est le prestige qu'apporte une telle réalisation.

1.4.5.2 "Le contre"

Les reproches les plus souvent formulés sont que les logiciels pêchent par manque de convivialité, tant au point de vue interactif qu'au point de vue production de rapports.

Le style directif et la présentation désuète des logiciels sont mal ressentis par la communauté scientifique. De plus, ces systèmes ne sont pas "auto-constructifs", c'est-à-dire que la seule façon de combler une lacune est de se procurer le "release" correspondant (s'il existe !).

Actuellement beaucoup de statisticiens souhaitent disposer d'un système où ils peuvent programmer eux-même, utiliser des programmes pré-conçus, moduler le graphisme, ne pas perdre leur temps avec des problèmes de syntaxe et au contraire posséder un outil agréable à utiliser. Alors qu'au contraire, ils sont obligés pour le moment d'effectuer un travail de bénédictin en déchiffrant les volumes de mode d'emploi et de choisir l'algorithme qui "se rapproche le plus de celui qu'ils désirent utiliser".

Remarquons encore, que le rapport qualité/prix des logiciels sur micro ordinateur est désastreux!

Chapitre deuxième

Caractéristiques fonctionnelles d'un poste de travail statistique

Nous développons ici les concepts statistiques et nous exposons les propriétés, qualités, possibilités qu'un poste de travail statistique devrait offrir pour satisfaire un utilisateur.

Et nunc reges, intelligite; erudimini, qui judicatis terram.
<Psalmiste (ps. II, 10)>

§ 2.1 Le point de vue des données

2.1.1 Type principal

2.1.1.1 Présentation des données

Les données se présentent la plupart du temps sous forme d'un échantillon. Un statisticien définit un ***échantillon*** comme une itération d'items où un item est une itération de valeurs, et où chaque item provient de l'observation d'un individu pris (au hasard) dans une population d'individus comparables.

Un informaticien traduit donc dans son jargon : un échantillon est un ensemble fini d'articles de même type, un article correspondant à l'observation d'un individu d'une population.

Chaque item du type d'article est obligatoire, simple, non répétitif ou à répétitivité fixée, mais aussi à longueur éventuellement variable pour un item du genre chaîne de caractères. Un item peut être du type entier, réel, booléen, chaîne de caractères, rang, date, ensemble.

Exemple :

nom	prénom	nombre de cigarettes / jour	taux de cholestérol

type d'article — items

Cavez	jean-jacques	0	0,0015

article — valeurs d'items

Dutronc	olivier	256	0,003
Cavez	vincent	0	0,0001
Bierman	martine	25	258,000

échantillon

Faisons remarquer une propriété de l'échantillon : certains items sont déterminants, et d'autres sont déterminés. Dans l'exemple, le nom et le prénom déterminent le nombre de cigarettes par jour et le taux de cholestérol. Cette propriété nous servira quand nous parlerons des manipulations de données.

2.1.1.2 Introduction des données

L'introduction des données se fait généralement à partir du clavier du poste de travail. Etant donné le nombre élevé de valeurs à introduire, le facteur de risque d'erreur est lui aussi élevé. Il est donc important de soigner l'ergonomie de la saisie.

Cette ergonomie peut être suscitée par différentes méthodes : facilité de correction des erreurs, facilité de visualisation des valeurs introduites (non seulement afficher la valeur à l'écran, mais aussi par exemple permettre la saisie de valeurs "sous graphique" pour détecter les valeurs aberrantes sur dessin), possibilité de sauvetage périodique des valeurs introduites, possibilité de découpe du travail dans le temps et donc de reprise du travail interrompu. De plus, toutes les fonctions "d'édition" doivent déjà être présentes afin de corriger des erreurs plus importantes (Si par mégarde on a sauté une feuille d'enquête, il faut pouvoir l'insérer à sa place après avoir constaté l'erreur)

Il ne faut pas oublier, enfin, que la personne qui va introduire les valeurs n'est généralement pas une spécialiste du système (étudiant jobiste, secrétaire ...). Cela veut dire que la maîtrise de toutes les commandes nécessaires à l'exécution correcte du travail doivent pouvoir être acquises en un minimum de temps, estimé à quinze minutes.

En d'autres termes :

1° il faut simplifier l'utilisation des commandes

ceci peut être fait par le fait de :

-garder la même ligne d'idée tout au long des divers centres d'activités (*exemple : supprimer un mot, une ligne, une colonne, une lettre, un paragraphe, etc ... se fera toujours par la même commande*).

-rendre l'emploi des actions habituelles le plus simple possible, quitte à créer une deuxième vitesse pour l'emploi des actions sophistiquées.

-généraliser au maximum les manipulations directes et en corollaire créer le plus de transparence possible (*exemple : la manipulation d'échantillons se fera sous l'image d'une classique armoire à dossiers*)

===> ainsi on tendra vers un système où "the display becomes reality" **[1]** <===

2° il faut intégrer la documentation dans le logiciel (ce qui correspond à la fonction Help de certains systèmes)

3° il faut offrir la possibilité de réversibilité des commandes, afficher immédiatement les conséquences des commandes, ne jamais lire quelque chose au clavier sans message d'erreur ou d'accusé de réception.

4° les messages d'erreur doivent être : visibles, complets et détaillés, avec les explications nécessaires pour éviter cette erreur à l'avenir et pour continuer la session compte tenu de l'erreur qui vient de survenir.

2.1.1.3 Validation

L'échantillon recueilli contient un certain nombre de données déclarées mauvaises. Sous ce vocable on réunit : les données manquantes, les données illisibles, les données manifestement hors de proportion etc... . La validation consiste alors à traiter cet échantillon de façon à uniformiser ces données.

Il faut d'abord repérer ces données mauvaises, ceci peut généralement se faire par visualisation des valeurs (tableaux de chiffres), par visualisation sous forme graphique (existe-t-il des pics anormaux ?), par

détection automatique des valeurs hors d'un certain rang spécifié.

Ensuite ces données sont traitées soit manuellement (suppression, correction, etc...), soit automatiquement (moyenne globale, approximation linéaire, positionnement d'un indicateur qui sera utilisé ultérieurement etc...).

Le but final de la validation est d'obtenir un échantillon qui soit cohérent pour l'analyse statistique qui lui sera appliquée ‹ *exemple : si pour un article d'un échantillon qui concerne l'étude du rapport entre taux de cholestérol et âge, la donnée âge manque, cet article sera effacé ; si pour un article d'un échantillon qui concerne l'étude de l'évolution du temps de réaction d'une vache à une impulsion électrique, une donnée temps manque, on pourra éventuellement estimer celle-ci par la moyenne entre le temps qui précède et celui qui suit le temps manquant* ›

2.1.1.4 Gestion

A partir d'un certain nombre d'échantillons enregistrés, il faut prévoir un gestionnaire des données. Ce gestionnaire s'occupera nécessairement de tout ce qui concerne l'espace disque : recherche de la place nécessaire pour enregistrer un échantillon, suppression physique des échantillons, réorganisation de l'espace disque, détection des zones dégradées, prévention contre les incidents pouvant survenir.

De plus l'interface de ce gestionnaire devra être aussi attrayant que possible, ceci peut être réalisé de la même manière qu'au point 2.1.1.2 : simplicité, transparence, manipulations directes, documentation, réversibilité, messages d'erreur. Ceci pour éviter l'écueil qu'est la présentation classique d'un directoire pour un novice : de nouveau les analogies et métaphores guident l'utilisation intuitive (avec d'éventuels essais et erreurs) du système [A1]. Pour permettre cette gestion, le concept de dossier est très utile, en effet, il permet de regrouper plusieurs

échantillons sous une même couverture et ainsi de traiter ce groupe comme on traiterait un seul élément. De plus, l'analogie avec un dossier du monde réel est telle qu'on voit immédiatement ce que l'on peut faire : retirer ou ajouter un échantillon dans un dossier, ouvrir ou fermer un dossier, jeter un dossier complet à la poubelle etc....

2.1.2 Manipulation des données

2.1.2.1 But

Le but des manipulations est d'effectuer un travail d'exploration des données. Il faut donc offrir une série de primitives courantes qui permettent de faire un premier travail de traitement des données. Ce traitement peut comprendre transcodage, filtrage, transformations arithmétiques (additions, soustractions, exponentiations, logarithmes sous diverses bases, sinus, etc..., etc...), calcul de valeurs à partie de l'échantillon, calcul de moyennes & variances, jointure de plusieurs échantillons, élimination de certains articles, vérification ou création d'un identifiant. Si A, B, C sont trois items déterminants nous pouvons demender d'éliminer les lignes telles que C = c . Nous pouvons également demander la somme des valeurs d'items des lignes telles que A = a et B = b . L'identifiant sera bien sûr basé sur les items déterminants.

Certaines explorations exigent des calculs longs et fastidieux, néanmoins les fonctions sont le plus souvent assez simples, ce qui incite à une forte interactivité avec l'utilisateur.

Cette exploration constitue une première description des données, et fera ressortir d'éventuelles caractéristiques qui permettent de guider le statisticien dans la phase d'analyse qui suivra.

2.1.2.2 Qualités principales désirées

L'accent sera mis sur la facilité d'utilisation, en effet, cette étape de manipulation des données se présentera dans quasi tous les cas d'analyse, et il ne faut pas perdre son temps dans une jungle de commandes diverses et variées.

On peut concevoir cette partie de système comme un éditeur amélioré, spécialement conçu pour travailler sur des tableaux de valeurs. Le concept de

feuille électronique de calcul utilisé par nombre de logiciels [A4, A6, A11] convient généralement assez bien.

Toutefois, ces feuilles électroniques de calcul sont essentiellement destinées à des travaux "statiques", ce qui entraîne des difficultés lorsqu'une série de valeurs doivent être obtenues de façon "dynamique" par un algorithme.

Prenons par exemple le calcul du périodogramme d'une série temporelle X(t), t=1..70

$$\text{périodogramme} = y(i)_{i=1,65} = \left(\left(\sum_{j=1}^{70} \cos(\pi/64*i*j)*X(j) \right)^2 + \sum_{j=1}^{70} \sin(\pi/64*i*j)*X(j) \right)^2 * 2/70$$

On peut représenter le calcul du périodogramme par une feuille électronique du type de celui qui suit :

Série temporelle	Périodogramme	
X(1) X(2) . . . X(70)	Y(1) Y(2) . . Y(65)	zone de calculs

Cela ne présente pas de grosses difficultés conceptuelles, mais l'utilisation effective de cet outil est désastreux tant au point de vue de la place mémoire nécessaire, qu'au point de vue temps de calcul, de plus toute manipulation de la feuille électronique qui n'est pas un déplacement de la fenêtre est rendu fastidieux étant donné le temps nécessaire pour retrouver en mémoire, ou recalculer les valeurs concernées.

Nous pouvons donc estimer que ce concept, quoique très utile et très attrayant n'est pas suffisant dans notre cas. Il faut pouvoir soit enrichir; soit rajouter d'autres concepts, d'autres outils, éventuellement basés sur le même principe, mais avec des spécifications beaucoup plus étroites.

Si nous reprenons l'exemple du périodogramme, nous construirons un concept de feuille à deux colonnes

Série temporelle	Périodogramme
X(1) X(2) . . . X(70)	Y(1) Y(2) . . Y(65)

dont les spécifications seront : afficher dans la deuxième colonne le périodogramme de la série temporelle introduite dans la première colonne.

Nous pouvons également construire une feuille électronique de calcul (également appelée tableur) avec certaines primitives de haut niveau du genre du périodogramme, ou avec la possibilité d'associer des sections de programme, ou algorithmes, à une case ou à une série de cases.

De plus, un dessin étant souvent beaucoup plus significatif qu'une liste de valeurs, la présentation des valeurs sous forme graphique est importante. Il faut notamment rendre très facile le passage de valeurs vers le graphique, et l'obtention de graphiques sous diverses formes (fromages, barres, lignes, ...).

2.1.3 Types auxiliaires

2.1.3.1 Présentation et introduction

Il peut arriver que l'utilisateur désire introduire des données autrement que par le clavier, par exemple, s'il possède des fichiers contenant des informations recueillies par des collègues ou encore par le biais d'un digitaliseur. Nous ne considérons pas d'autres méthodes d'acquisition de l'information (par exemple : sonde médicale, impulsion de caméra électronique, ...) car ces périphériques sont directement liés à un processus, or nous examinons ici un poste de travail pour un statisticien "généraliste" plutôt que pour un type bien précis d'analyse.

L'introduction des données de type auxiliaire se fera alors via une section de programme écrite spécialement à cette fin et qui lira le fichier ou interprétera les impulsions du digitaliseur. Ce programme produira des données du type principal.

2.1.3.2 Gestion et manipulations

Il faut que d'une façon ou d'une autre, les données ainsi acquises puissent être exploitées par le statisticien. Il devra donc avoir à sa disposition, soit un système qui lui permette de traiter n'importe quel type de données (ce qui est utopique), soit un moyen de convertir ces données en un type utilisable par le système.

§ 2.2 Le point de vue du traitement

2.2.1 But

Les données introduites doivent être traitées par des moyens plus sophistiqués. Il existe une grande variété d'algorithmes qui permettent de pratiquer l'inférence statistique. De plus, ces algorithmes peuvent s'enchaîner de telle façon que les résultats de l'un deviennent les données de l'autre, ce qui est également valable pour un seul algorithme qui traite ses propres résultats.

Le but retenu ici est de faciliter l'utilisation "sur-mesure" du système ("user-taylorable"). En effet, la principale pierre d'achoppement dans les systèmes classiques est la "conduite sur rails" (voir introduction). Nous proposons donc un système où l'utilisateur prend ou rejette les algorithmes proposés, les assemble de façon très facile *(à la façon dont les enfants empilent des cubes, pris dans une boîte, pour créer une tour)*.

De plus, l'utilisateur devra pouvoir créer ses propres algorithmes, copier ou modifier des algorithmes existants, introduire des algorithmes qui furent créés hors système, etc...

Le contact avec l'extérieur sera ici aussi très important, tout comme dans le cas des données. Nous pourrions ici reprendre toutes les considérations faites pour l'introduction des données. Retenons simplement ces mots clés : Simplicité, Transparence, Manipulations directes, Documentation intégrée (help), Réversibilité, Messages d'erreur.

2.2.2 Moyens

2.2.2.1 Fonctions standard d'analyse

Une fonction standard est une fonction qui s'utilise correctement dans le système, c'est-à-dire qui ne nécessite pas d'interfaçage particulier pour exploiter le type principal de données. Usuellement, il s'agit soit d'un algorithme conçu par le constructeur du système, soit un algorithme construit en tenant compte de la philosophie et des principes de base du système.

Seules ces fonctions seront utilisables dans le système, elles peuvent soit être considérées comme des blocs de base dans la construction d'autres fonctions qui seront automatiquement standard, soit être considérées comme "programmes".

Ces fonctions peuvent utiliser toutes les ressources disponibles de l'ordinateur sur lequel elles fonctionnent (disques, imprimantes, écran graphique, digitaliseur ...).

De plus, elles peuvent créer, lire, supprimer des fichiers ou autres supports logiques de données. Toutefois, pour garder l'homogénéïté du système, il sera recommandé d'éviter d'utiliser d'autres supports logiques que celui du type principal de données.

2.2.2.2 Fonctions non standard d'analyse

Il est fréquent de rencontrer une fonction non standard, c'est-à-dire une fonction qui fut programmée "hors-système" et qu'on désire faire fonctionner "dans-système".

Par exemple, on extrait d'un livre, d'une revue, un algorithme qu'on ne désire pas réécrire en fonction des types de données disponibles dans le système.

Ces fonctions ne peuvent être utilisées comme telles dans le système

(sauf exception rarissime), il faudra donc prévoir des possibilités d'intégration facile de ces fonctions.

2.2.3 Propriétés

2.2.3.1 Propriétés des fonctions d'analyse

Outre les propriétés évidentes d'adéquation aux problèmes du statisticien, il faut considérer aussi certaines qualités vis-à-vis de "l'informaticien".

Il faut standardiser les données en entrée comme les données en sortie, de façon à ce qu'on puisse autant que possible passer les valeurs produites par une routine, comme arguments d'une autre (ou de la même) routine, sans devoir "reformater" les données.

Pour la souplesse d'utilisation, une conception modulaire des routines facilitera la combinaison des fonctions pour une analyse complexe.

Il faut faciliter au maximum la création de scénarios propres à l'utilisateur, et ne pas le forcer à utiliser des scénarios tout faits, qui ne le satisferont sans doute pas. Dans la même optique, il est utile de pouvoir visualiser les résultats de plusieurs fonctions d'analyse de façon simultanée, nous aurons donc besoin de la notion de fenêtres.

2.2.3.2 Propriétés d'interfaçage

Comme nous l'avons déjà expliqué, il existe un problème avec les fonctions non standard.

Prenons par exemple une fonction de traitement qui travaille sur des matrices. Si nous considérons par exemple que le type de données standard est le tableau, il faut pouvoir traduire le tableau en matrice pour le traitement, et traduire la matrice en tableau pour la sortie des résultats. Ce procédé permet d'obtenir des fonctions de traitement qui, vues de l'extérieur, travaillent toutes avec la même structure de données

Nous proposons de semi-automatiser cette traduction. L'utilisateur adjoindra à son programme quelques lignes avec des primitives de traduction

de haut niveau et passera son programme dans "une machine de traduction" qui produira du code, ce qui permet alors d'intégrer ce programme au système puisqu'il connaîtra la notion de tableau.

Les fonctions d'analyse communiqueront entre elles au moyen d'une zone spéciale où elles pourront envoyer ou chercher des informations, cette zone s'appelle clip-board dans certaines applications [A6..A9].

2.2.3.3 La gestion des fonctions d'analyse

De même que pour les données, il sera très utile d'offrir des facilités de gestion [consultation, copie, suppression, etc...] des différentes fonctions d'analyse qui seront stockées.

Ce gestionnaire doit bien sûr s'organiser en fonction des buts assignés au point de vue du traitement (voir le point 2.2.1.). Nous devons donc pouvoir facilement passer en revue les fonctions d'analyse déjà existantes, ou du moins leurs caractéristiques; de préférence en parallèle avec l'élaboration d'une nouvelle fonction.

Le gestionnaire devra aussi s'occuper de vérifier les liens entre les versions des fonctions d'analyse, en effet, si on modifie une fonction et que cette fonction est utilisée pour plusieurs analyses, il y a risque que les modifications effectuées perturbent le déroulement habituel de l'analyse. Il faudra donc indiquer pour chaque fonction, de quelle version il s'agit, et éventuellement quelles modifications y ont été apportées.

Nous voudrions également, comme dans le cas des tableaux, avoir un concept de dossier, ou plutôt de boîte à fonctions. Ceci nous permettrait de classer, de supprimer, de choisir, de répertorier facilement les différentes fonctions d'analyse statistique.

2.2.4 Fonctions spéciales

Il s'agit ici de fonctions offertes par le concepteur du système, la fonction help étant intégrée en sus dans tout le système.

2.2.4.1 Calculette

Nous offrons quelques fonctions de calcul, analogue à ce que pourrait donner une calculatrice de poche, plus un volume de tables diverses (x^2 etc...).

Ces fonctions doivent pouvoir être utilisées en "parallèle" avec les fonctions standard. (par exemple pour comparer le résultat obtenu avec la valeur d'une table).

2.2.4.2 Display

Le display permet, de façon très souple, de présenter sous forme graphique une série de résultats numériques. Il créera à la demande : graphiques, histogrammes, lissages, etc... [A5, A8]

De plus, on prévoiera une série de primitives de manipulations graphiques, de façon à pouvoir attirer l'attention sur certains points (par exemple, un histogramme avec un élément hors du dessin). [A7]

2.2.4.3 Secrétariat

Le secrétariat est destiné à pouvoir rédiger avec facilité et précision tous les rapports, au moyen d'un éditeur de texte avec insertion de tableaux de chiffres, de dessins, de graphiques. On peut éventuellement prévoir une gestion de bibliographie, et autres facilités.

Chapitre troisième

Spécifications d'un poste de travail statistique

Les concepts informatiques nécessaires, les fonctions à réaliser, pour l'élaboration d'un poste de travail statistique sont ici exposés.

Quand les valeurs sont en place, le plus gros est fait.
‹Marquet (1875-1947)›

§ 3.1 Les tableaux

3.1.1 Idées de base

3.1.1.1 Présentation des données

Le type principal des données, qui fut exposé au point 2.1.1, nous permet de concevoir logiquement un échantillon comme étant un ***tableau*** de valeurs, avec les lignes correspondant aux articles et les colonnes aux valeurs prises par les items pour chaque article.

Ce concept de tableau n'est pas nouveau, en effet, on le retrouve dans différents softwares, tels que XXXCALC. Par contre, on semble l'ignorer dans SPEEDSTAT, STATPRO, GLIM, BMDP, SPSS, au profit du concept de liste de nombres ou de fichiers.

L'avantage du tableau est la possibilité de manipulation des cases, des lignes, des colonnes ….Ce qui est difficilement exprimable, par exemple sur un fichier. De plus, dans le cas du tableau, l'intuition guide la pratique de l'exploration, étant donné que le modèle conceptuel est à la fois riche et facilement maîtrisable.

3.1.1.1 Adéquation du modèle relationnel

Le modèle relationnel [A21, A22] semble cerner notre problème. En effet, le concept de tableau lui est caractéristique, et de plus, il nous offre une méthodologie rigoureuse et puissante de manipulation de ces tableaux, sur eux, et entre eux. Cette méthodologie permet de travailler sur les définitions des colonnes : par exemple si nous avons le tableau suivant

Echantillon	Valeurs
	0.03
	0.27
	0.32

Nous pouvons définir un nouveau tableau

Nouvel-Echantillon	Nouvelles-Valeurs
	Valeurs * 100

c'est-à-dire

Nouvel-Echantillon	Nouvelles-Valeurs
	3 27 32

Toutefois, ceci ne suffit pas, en effet, il est par exemple impossible d'exprimer par ce moyen les opérations du type : la nouvelle valeur est l'ancienne moins l'ancienne qui la précède dans la même colonne du tableau.

Nous modifions donc et enrichissons le modèle relationnel pour obtenir notre modèle de tableau convenant au statisticien. Notamment, en oubliant certains aspects spécifiques aux bases de données, et qui ne sont d'aucune utilité pratique pour nous : par exemple la sélection nous est très utile, notamment pour éliminer d'un échantillon les lignes correspondant à des données erronnées. De même, l'union, l'intersection, la différence, nous servent lorsque les données ne sont pas toutes introduites d'une seule traite. La projection sert pour effectuer des effondrements. Par contre la jointure et la composition n'ont pas de sens ici, car elles ne sont pas compatibles avec les méthodes de prospection aléatoire utilisées en statistique; le produit cartésien , le complémentaire et la division ne seront pas non plus utilisés.

Nous nous inspirerons des tableurs (ou feuilles électroniques de calcul) pour retailler notre modèle.

Le tableau étant ce qu'on appelle un "type abstrait", nous le définirons explicitement en donnant l'ensemble des opérations et manipulations qui peuvent lui être appliquées.

3.1.2 Concrétisation

3.1.2.1 Définition du tableau

En s'inspirant à la fois du modèle relationnel et des tableurs, nous créons le concept de tableau:

Le tableau est une matrice de cases telle que:

- un article est entièrement caractérisé par une ligne
- l'ensemble des valeurs prises par un item est représenté par une colonne (*toutes les valeurs d'une colonne sont donc du même type**)
- une case est l'intersection d'une ligne et d'une colonne et représente donc la valeur prise par un item d'un article

Remarques :

1 Cette définition implique que les contenus des différentes lignes ou colonnes ne sont pas forcément distincts, ce qui est plus large que le modèle relationnel, mais plus restreint que les tableurs car toutes les cases d'une colonne sont du même type.

2 On peut avoir plusieurs types de valeurs fausses :

-manquant

-impossible

-illisible

ce qui peut être généralisé à "type de valeur anormale$_i$" $i = 1 \ldots \infty$

Une case est donc composée de deux parties : la valeur proprement dite et un indicateur qui peut être positionné de différentes façons (pour refléter les

---

* nous allons abandonner cette contrainte pour une raison qui apparaîtra au point 3.1.2.2

différentes valeurs anormales et donner la possibilité à un programme de prendre une décision en conséquence)

3 Le tableau a un nom et un type, le type du tableau étant donné par l'énoncé dans l'ordre des types des valeurs admises pour chaque colonne.

4 Nous enrichissons cette définition par les opérations et manipulations possibles : voir 3.1.2.2 et 3.1.2.3

3.1.2.2 Opérations à effectuer sur un tableau

Nous ne développons ici que la sémantique, et non la syntaxe qui est le mode de déplacement dans le tableau, l'affichage et la présentation du tableau, le mode de présentation des formules de calcul etc .... En effet, cela relève plutôt de la réalisation physique du système, qui doit tenir compte des possibilités hardware et logiciel disponibles. De plus, les qualités souhaitées pour cet interface ont été présentées au chapitre deuxième.

Remarque : dans la suite de ce mémoire nous allons désigner par échantillon tant les résultats d'un échantillonnage que les données qui résultent de calculs effectués sur cet échantillon et qui conservent la même structure. (Que les puristes nous pardonnent.)

Les opérations peuvent être effectuées sur une case, sur une ligne, sur une colonne ou sur un tableau en entier:

pour une case :

Nous pouvons

- insérer / supprimer / copier / modifier une valeur* (sans devoir la réécrire en entier)
- marquer la case de façon à pouvoir en tenir compte dans les calculs (utile pour valeur fausse)
- obtenir une valeur dans la case au moyen des fonctions suivantes appliquées à une autre case:

x arithmétiques : ABS
EXP
INT
ROUND
TRUNC
LN
LOG
SQRT
Φ
Φ^{-1}

---

* (avec conservation de la cohérence au niveau des types et des fonctions éventuelles)

x trigonométriques : SIN
COS
TAN
ASIN
ACOS
ATAN

- obtenir une valeur dans la case au moyen des fonctions suivantes appliquées à une ou plusieurs autres cases

x logiques :

répond vrai, faux à

(¬)
<
>
=
≠
∨
∨
IF
≤
≥

x "conversion en fonction de tables" (la valeur de la case sert d'entrée à une table et est remplacée par la valeur trouvée dans la table (qui est incluse dans le tableau)).

Le but de ces opérations est de fournir dans une case, le résultat d'un calcul effectué sur une autre case, sur une colonne (ou une ligne) ou en appliquant un opérateur logique à deux cases. On peut bien sûr combiner toutes ces opérations de façon à obtenir des effets sophistiqués.

sur une ligne ou une colonne:

Nous pouvons

- insérer / supprimer / déplacer / copier une ou plusieurs lignes ou colonnes*
- créer / supprimer / modifier un type et/ou un format*
- répéter le type et le format sur une ou plusieurs colonnes

  (le format précise simplement le nombre de chiffres après la virgule qu'il convient d'afficher)
- générer une ligne ou une colonne de nombres aléatoires uniformément distribués entre deux bornes, ou distribués selon une loi normale dont δ et μ sont donnés.

- Nous pouvons également demander de donner dans une case la somme des valeurs, la somme des carrés des valeurs , la moyenne, la variance, le nombre d'éléments d'une ligne ou d'une colonne. C'est ici que nous voyons l'utilité d'avoir des cases dans une colonne qui ne soient pas toutes du même type. Pour cela nous avons ajouté un type de colonne qui est "mixte", et où chaque case peut avoir un type propre.

Pour le calcul des valeurs statistiques, nous nous basons sur une étude de Neely [10] :

$M1 := (\sum X)/N$

$MOYENNE := (\sum X)/N + \sum(X - M1)/N$

$STANDARD\ DEVIATION := \sum(X - M1)^2 - (\sum(X-M1))^2/N$

---

* (avec conservation de la cohérence au niveau des types et des fonctions éventuelles)

sur un tableau :

- copie d'une partie de tableau dans un autre (ou le même)*
- répétition du type et/ou du format d'une partie de tableau dans un autre (ou le même)
- impression (*hardcopy*) d'une partie de tableau : soit rien que les valeurs ou les fonctions, soit les valeurs et les fonctions
- remarque importante : Chaque fois que l'on copie ou supprime quelque chose, que ce soit une case, une colonne, une ligne, une section de tableau ... une copie vient se placer dans un ***buffer***, qui est une zone acceptant tout type de données. Nous pouvons exploiter ce qui se trouve dans le buffer, soit pour le recopier dans un autre tableau (on peut alors facilement communiquer de tableau à tableau), soit pour le recopier à une autre place dans le même tableau, soit comme input d'une fonction d'analyse ou d'une fonction spéciale. Nous reparlerons encore du buffer dans la suite de ce mémoire.

---

* (avec conservation de la cohérence au niveau des types et des fonctions éventuelles)

3.1.2.3 Manipulation de tableaux

Il s'agit ici de manipulations sur un tableau et entre tableaux, pour en obtenir d'autres. Nous avons la possibilité d'obtenir un nouveau tableau par

- changement d'unité, d'échelle : le nombre de variantes possibles étant trop élévé nous nous limitons à la possibilité d'appliquer une formule numérique à l'ensemble du tableau.
- application d'une fonction linéaire
- calcul de marginales
- application de statistiques simples (moyenne, variance, ...)
- tri sur un certain nombre d'attributs, dans un sens déterminé
- application d'un masque sur certaines colonnes/lignes
- création de lignes/colonnes virtuelles (calculées à partir d'autres)
- copie des lignes d'un tableau dans un autre, selon les critères sur les lignes
- vérification/établissement d'un identifiant. Dans certains cas il est nécessaire de pouvoir identifier chaque ligne du tableau de manière unique, notamment si nous voulons y faire un choix. Pour cela nous pouvons décider que la concaténation (par ligne) des valeurs de certaines colonnes formera un identifiant. On vérifie alors que chaque mot obtenu par concaténation est bien unique. On peut aussi demander une colonne spéciale (de comptage) qui identifie de façon univoque chaque ligne.
- protection de parties de tableau contre d'éventuelles erreurs de manipulation
- projection sur un sous-espace et application de primitives de plus haut niveau sur les vecteurs et les matrices à deux dimen-

sions

remarque : on ne reprend pas ici l'inversion d'une matrice ou autres primitives de même genre, car l'aspect numérique entre en compte et plusieurs algorithmes sont parfois nécessaires.

- croisement d'un jeu de colonnes avec un autre (tables de contingence)

De plus, il faut pouvoir gérer ces tableaux, la meilleure méthode étant celle qui se rapproche le plus de la vie courante, nous préférerons un système de fardes (visualisé sous forme de pictogrammes à l'écran) plutôt que le classique "directoire de fichiers".

Une farde, ou un dossier, permet de regrouper les tableaux sous un même label, de supprimer, de dupliquer... par groupe de tableaux et de les gérer facilement malgré leur nombre. Nous retrouverons le même principe avec les boîtes à outil. Pratiquement, cela se fera au moyen de pointeurs (que l'on peut créer, déplacer, supprimer) qui relieront les tableaux sous forme d'arbre où chaque noeud intermédiaire, ainsi que le sommet est représenté par une farde. On peut accrocher un tableau ou une farde à n'importe quel noeud, on peut également les décrocher et les raccrocher à un autre noeud, ou les supprimer.

§ 3.2 Les boîtes à outils

3.2.1 L'outil

3.2.1.1 But et définition

Suite aux considérations du paragraphe 2.2 (le point de vue du traitement) nous définissons "l'outil" comme étant une unité logique de traitement correspondant à une fonction d'analyse. L'outil peut utiliser en input comme en output les ressources offertes par le système d'exploitation et le langage de base du matériel sur lequel tourne le système.

Nous signalons toutefois, que si nous ouvrons également la porte à d'autres structures de données que le tableau, nous insistons sur la nécessité d'utiliser au maximum le concept de tableau. En effet, la standardisation des inputs et outputs est une des clés maîtresses pour l'obtention d'un système modulaire d'outils, comme exposé au point 2.2.1 .

La communication entre deux outils se fait de préférence via le buffer, qui est le même que celui utilisé pour les tableaux. Ainsi une section de tableau est exploitée facilement par un outil, il suffit de demander une copie, cette copie est placée dans le buffer et l'outil n'a plus qu'à chercher ce qu'il lui faut dans le buffer. Ce qu'un outil peut également faire, c'est de placer ses résultats dans le buffer, le manipulateur de tableaux est alors capable, pour autant que les structures de données soit correctes, de manipuler ces résultats.

Il faut également remarquer que nous ne développerons à aucun moment, dans le cadre de ce mémoire, un quelconque outil statistique particulier. En effet, le but de ce travail est de donner une idée d'un système global qui réponde aux besoins d'un statisticien du point de vue environnement logiciel; et non pas de créer une application qui puisse résoudre un problème statistique précis.

3.2.1.2 Réalisation

L'outil est composé d'une ou plusieurs unités physiques de traitement, c'est-à-dire routine, programme, ou même primitives extraites de mémoires ROM. Comme l'utilisateur peut créer lui-même ses propres outils, il le fait généralement sous forme de programme activable interactivement. Il peut donc insérer un outil dans le système (éventuellement le faire sortir), et peut l'activer en spécifiant les données à utiliser (c'est-à-dire *indiquer le tableau sur lequel travailler ?!...*). Nous désignons donc de façon univoque un outil (également vrai pour les données).

Evidemment, un programme peut toujours appeler un autre programme, ou une routine s'enchaîner à une autre routine, nous pouvons donc respecter les propriétés désirées pour les fonctions d'analyse. Pour faciliter "l'assemblage des algorithmes" nous associons à chaque outil un espace de commentaires destiné à expliquer le but de l'outil, les moyens utilisés et la composition de son interface.

Dans certains cas, l'outil peut même être destiné à la communication avec d'autres machines, via un réseau ou autre moyen. Pour certaines applications statistiques, il est utile de disposer de facilités de stockage et de puissance de calcul, le système ici proposé ne s'y oppose pas.

Le système Help est intégré, mais il ne porte que sur les opérations offertes par le système. Lorsque l'utilisateur crée son propre programme, il doit lui-même documenter son produit. Mais nous lui proposons une primitive qui permet, quand la touche "?" est enfoncée, d'afficher un message qu'il suffit de passer en paramètre.

3.2.2 La boîte à outils

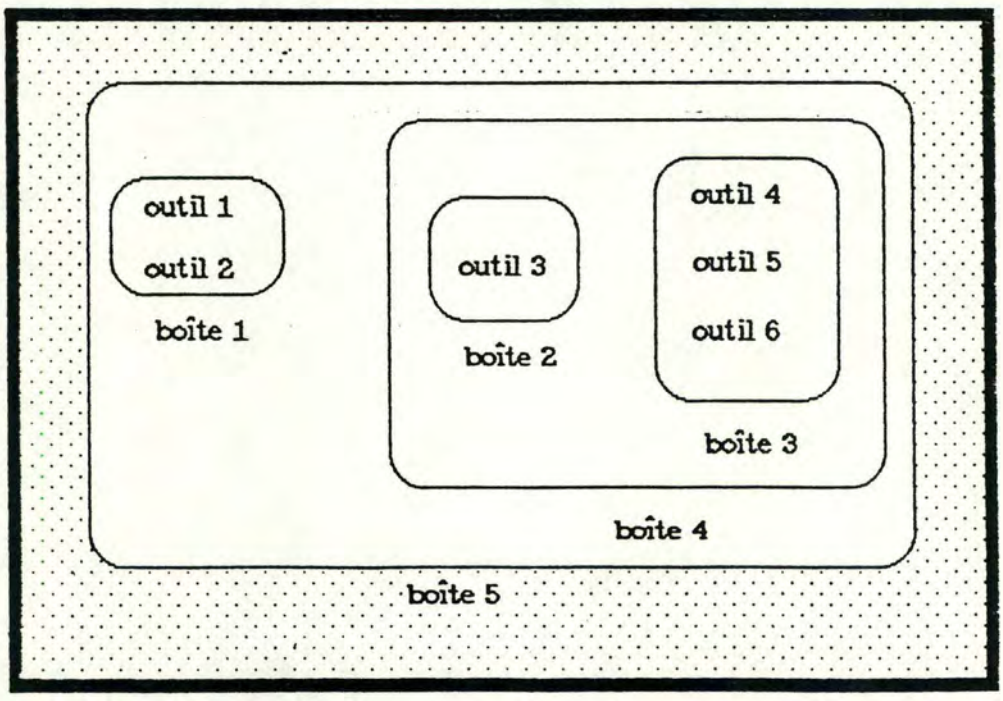
3.2.2.1 But et définition

La boîte à outils n'est pas indispensable, mais nous croyons qu'elle apporte une aide non négligeable.

En effet, le but de la boîte à outils est de pouvoir classer et regrouper les outils (selon les critères définis par l'utilisateur). Nous voyons donc facilement le gain de temps appréciable que fournit une telle aide : au lieu de devoir rechercher un outil dans une liste qui peut être longue, nous pouvons le retrouver beaucoup plus facilement si nous connaissons la boîte à laquelle il appartient.

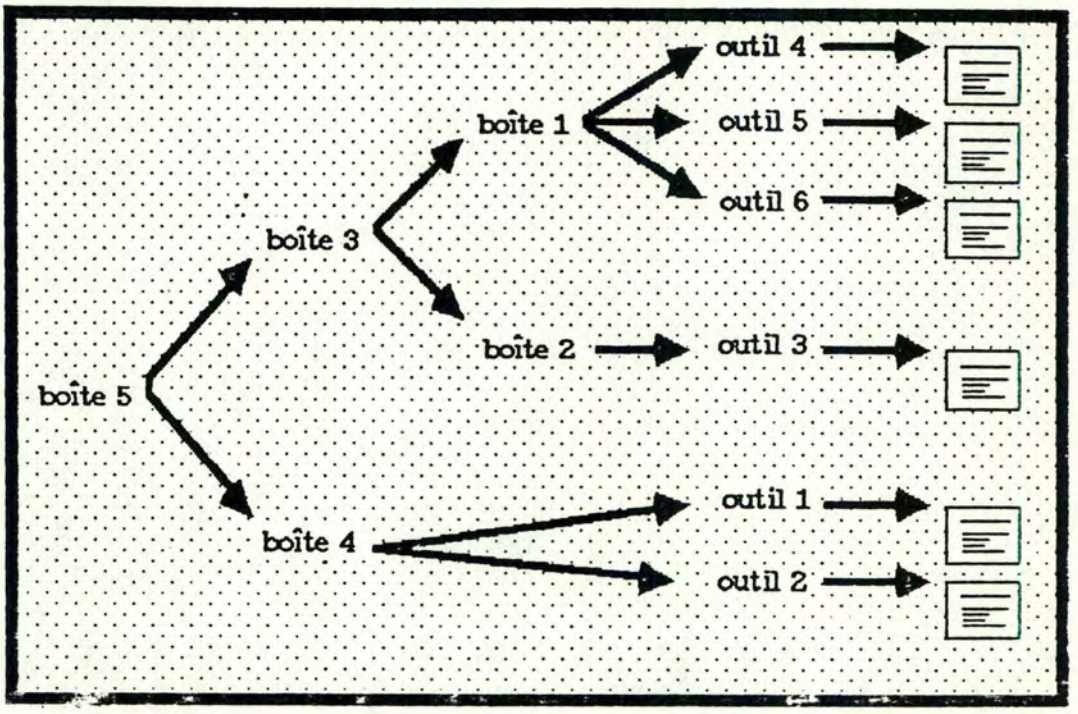
Nous généralisons d'ailleurs cette notion, en créant des boîtes de boîtes à outils etc ....Nous créons ainsi une structure qui peut être qualifiée de hiérarchique, ou en arbre. Il s'agit tout bonnement de la méthode la plus classique de rangement des objets dans un magasin (objet c paquet c boîte c caisse c étagère c ...).

-------- exemple ------



3.2.2.2 Réalisation

Pour réaliser le concept de boîte à outils, nous créons tout simplement des chaînes d'identifiants : identifiants d'outils, identifiants de boîtes. De façon schématique, si nous reprenons l'exemple précédent, en supposant les noms donnés comme étant identifiants, voici ce que nous obtenons



Il faut remarquer que certains micro ordinateurs possèdent déjà cette notion de boîte, que ce soit pour des "fichiers de données" ou des "programmes".

3.2.3 La gestion

3.2.3.1 La gestion des outils

Cette gestion consiste principalement à repérer et à entretenir les liens qui existent entre les outils, en effet on peut très bien avoir affaire à un outil qui utilise d'autres outils pour effectuer son travail (par exemple l'outil " Quelques statistiques élémentaires" utilisera certainement l'outil "Faire la moyenne" s'il existe déjà). On vérifie donc lors de la suppression d'un outil s'il n'y en pas d'autres qui sont mis en cause par cette suppression. On vérifie aussi si en modifiant un outil on ne remet pas en cause d'autres outils, et notamment on vérifie que l'on ne crée pas un bouclage (outil 1 utilise outil 2 qui utilise outil 1), ce qui peut provoquer des ennuis.

La gestion des outils comprend également les fonctions techniques telles que gestion de la mémoire et de l'espace sur le disque, compilation, linkage, précompilation ou autres opérations imposées par le matériel sur lequel on travaille ... de façon à ce qu'un outil appréhendable par l'utilisateur puisse être exécuté par la machine.

De plus, le gérant des outils, surtout quand il est utilisé en parallèle avec l'éditeur d'outils, permet de créer facilement des outils qui en utilisent d'autres. Que ce soit par sélection du texte d'un outil ou sélection de la partie nécessaire pour faire un appel correct, ou encore par affichage de l'espace de commentaires, le gérant des outils permet de ne rien devoir mémoriser puisque tout est accessible rapidement.

3.2.3.2 La gestion des boîtes à outils

Elle consiste à repérer les liens qui existent entre les outils et les boîtes, et entre les boîtes elles-mêmes. La gestion est un peu plus facile que pour les outils en ce sens que la modification du contenu d'une boîte n'a pas de répercussion au niveau des outils, et qu'on peut se passer de vérification.

Nous constatons qu'avec une succession d'outils bien conçus (c'est-à-dire qui reprennent les étapes successives d'un raisonnement statistique et qui possèdent des inputs/outputs compatibles) et de bons gestionnaires (qui permettent la création souple de combinaisons d'outils), nous atteignons les buts visés dans le point de vue du traitement.

3.2.4 Les outils non standard

3.2.4.1 La raison

Comme expliqué au point 2.2.2.2, on peut être amené à utiliser des procédures qui ne respectent pas les préconditions établies pour le système, notamment, ils ignorent la notion de tableau . On a donc prévu des moyens de "standardisation" de ces procédures.

Notons encore que certains outils, quoique n'utilisant pas la notion de tableau peuvent être standard; en effet, seuls sont non standard les outils qui ont besoin de la notion de tableau, pour être utiles dans le système, et qui n'en disposent pas. Un outil destiné à produire un travail de calculette peut très bien se passer de la notion de tableau tout en étant intégré dans notre système.

3.2.4.2 Une solution

La solution adoptée est de créer un précompilateur capable d'interpréter des primitives de haut niveau, destinées à la conversion des tableaux en la structure de données utilisée par l'outil non standard, et aussi à la conversion inverse.

Par exemple un programme utilisant des matrices 2x2 peut recevoir en input un tableau si on intercale dans le texte quelques lignes destinées à lire un tableau, et à s'en servir pour remplir la matrice. De même en sortie, au lieu du "print matrice", on a des lignes de conversion matrice -> tableau et d'impression du tableau.

Nous aurons dès lors des outils dans différents états : des outils "non précompilés" aux outils "exécutables", le nombre d'états dépendant du fait qu'il s'agit d'un langage interprété ou compilé, ou nécessitant encore d'autres étapes.

Pour préciser les idées, voici une liste des primitives de conversion offertes.

Hypothèse : tab est l'identifiant d'un tableau (si pour une primitive tab = ∅ alors l'action n'est pas effectuée, et un message d'erreur apparaît)

1°IDENTIFIE(chaîne_de_caractères,tab)

permet, au départ d'une chaîne de caractères, d'obtenir l'identifiant du tableau qui lui est associé. Si la technologie permet par exemple de disposer d'une "souris", cette primitive sert à trouver l'identifiant du tableau "cliqué". On peut prévoir d'autres moyens de choisir un tableau, mais il faudra toujours en obtenir l'identifiant avant de pouvoir continuer la suite des opérations.

2°CREE(chaîne_de_caractères)

crée un tableau du nom donné par la chaîne de caractères. L'identifiant est attribué par le système et le tableau ne comporte encore ni lignes ni colonnes.

3°TUE(tab)

permet de supprimer un tableau.

4°NL(tab),NC(tab)

permet d'obtenir respectivement le nombre de lignes et le nombre de colonnes du tableau.

5°TYPE(tab,i,chaîne_de_caractères)

renvoie une chaîne de caractères qui caractérise le type de valeurs admis dans la $i^{ème}$ colonne. Le type admis est soit un des types autorisés par le langage de base et par le constructeur du logiciel (dans notre cas , nous n'admettons que les réels et les chaînes de caractères), soit le type "mixte" qui signifie que les valeurs prises par les cases d'une colonne ne sont pas toutes du même type.

6°TYPE_DE_CASE(tab,i,j,chaîne_de_caractères)

idem que pour 6°, mais en se basant sur la case placée à l'intersection de la $i^{ème}$ ligne et de la $j^{ème}$ colonne. Nous n'avons évidemment plus de type "mixte".

7°TRANSFERE_VERS_MATRICE(tab,i,j,k,l,mat,m,n,o,p)

sert à recopier une partie de tableau dans une partie de matrice.

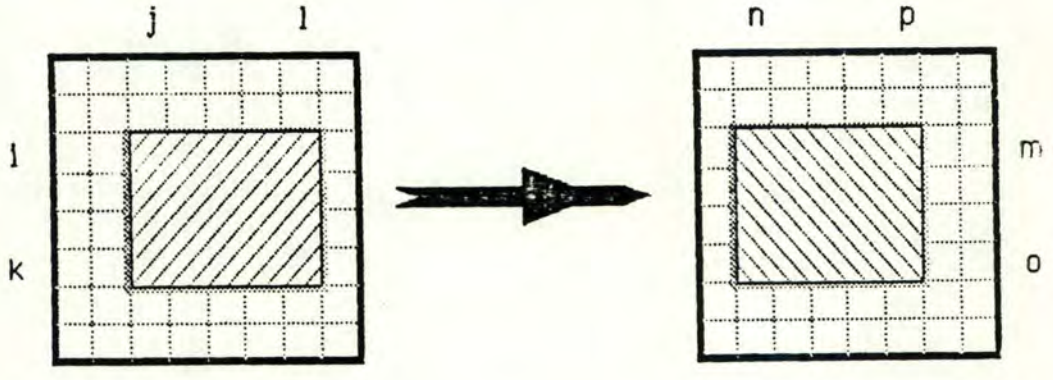


tableau : tab **matrice : mat**

Sous le couvert de cette primitive,il est vérifié si les lieux de départ et d'arrivée existent, si les types de données sont compatibles, si les tailles des deux "cadres" définis sont égales.

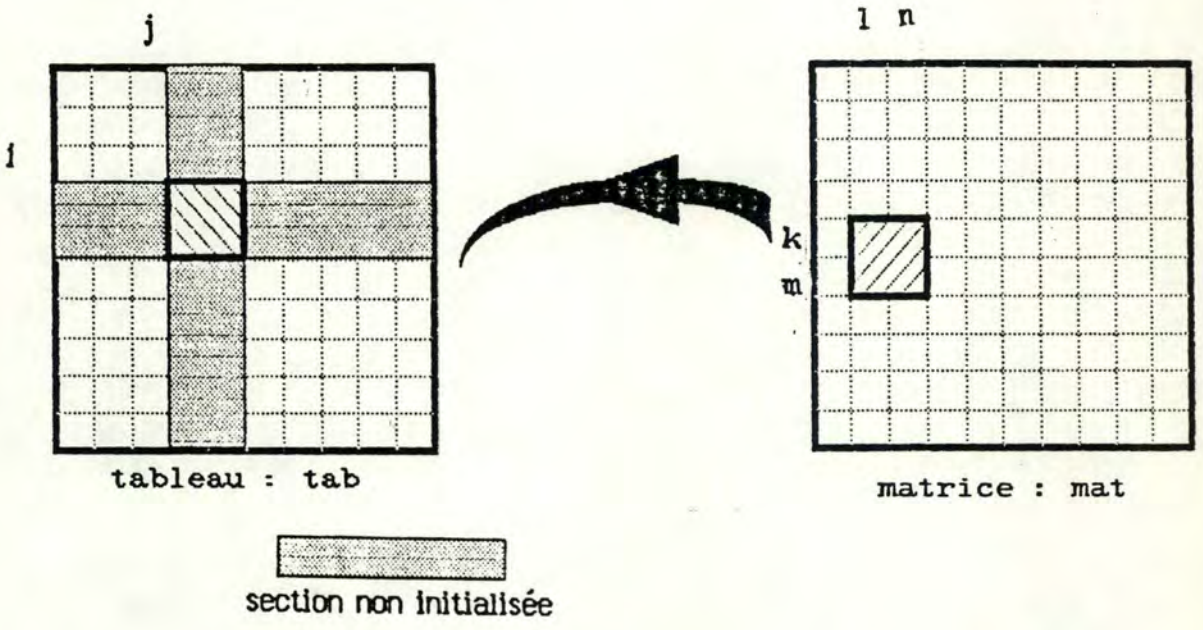
Toute erreur qui est détectée soit à la compilation soit à l'exécution engendre l'émission d'un message d'erreur.

L'opération inverse, de matrice vers tableau sera également présente :

TRANSFERE_A_PARTIR_DE_MATRICE(tab,i,j,k,l,mat,m,n,o,p)

8°INSERE(tab,i,j,mat,k,l,m,n)

nous pouvons ainsi insérer dans un tableau, après la $i^{ème}$ ligne et la $j^{ème}$ colonne, une section de matrice.



Si i ou j est négatif, cela veut dire qu'on ne veut pas insérer de ligne ou de colonne, auquel cas les sections non initialisées du tableau peuvent se réduire à néant. Les vérifications sont les mêmes qu'au point 7°.

9°COUPE(tab,i,j,n_i,n_j)

permet de supprimer n_i lignes après la $i^{ème}$ ligne et n_j colonnes après la $j^{ème}$ colonne.

Exemple d'utilisation :

Nous disposons d'un tableau appelé "pointillisme" constitué de deux colonnes, supposées de type réel, qui contiennent les coordonnées x et y d'une série de points de l'espace à deux dimensions R^2.

Nous disposons d'une routine appelée "regress" et de paramètres N, X et Y ; où N est le nombre de points et X et Y sont deux vecteurs réels qui contiennent en principe les coordonnées x et y des N points. Cette routine affiche l'équation de la droite de régression de Y sur X.

Construisons un outil qui, au partir du tableau "pointillisme", donnera la droite de régression de la deuxième colonne sur la première.

hypothèse : le langage de base est le Pascal, on indique au précompilateur les phrases à traduire en les faisant précéder d'un "@".

```
procédure exemple;
var x, y : array [1..100] of real;
    n: integer;
    tab: string[9];
begin
   @ IDENTIFIE("pointillisme", tab);
   @ n := NL(tab);
   (* pour simplifier nous supposons que le tableau est correct,
      notamment au point de vue des types de valeurs *)
   @ TRANSFERE_VERS_MATRICE(tab, 1, 1, n, 1, x, 1, n)
   @ TRANSFERE_VERS_MATRICE(tab, 1, 2, n, 2, x, 1, n)
   regress(n, x, y);
end;
```

§ 3.3 Les instruments particuliers

3.3.1 De calcul

Ce composant ne nécessite pas d'autres relations que celles d'accès à ses fichiers spécifiques, en effet, il n'est pas prévu pour qu'on lui passe des données autrement que par le clavier, ni pour qu'il passe des résultats à d'autres outils, ce n'est donc pas un outil à proprement parler.

Pour mémoire, ce composant est destiné à remplacer le travail de la calculette et la recherche dans les tables .

3.3.2 De présentation

Ce composant est destiné à produire des graphiques standard (scatter, barres, fromages, ...), il s'agit de graphique pur , c'est-à-dire qu'on ne peut ici effectuer aucune modification des données (sauf insertion/suppression), toutefois on pourra éditer le dessin.

Il faut pouvoir

- avoir une possibilité d'input à partir de tableaux
- superposer des graphiques
- choisir des échelles, des types de représentation (points croix, fourchettes, ...)
- insérer du texte
- insérer des dessins ou tracer des figures

3.3.3 D'édition de rapports

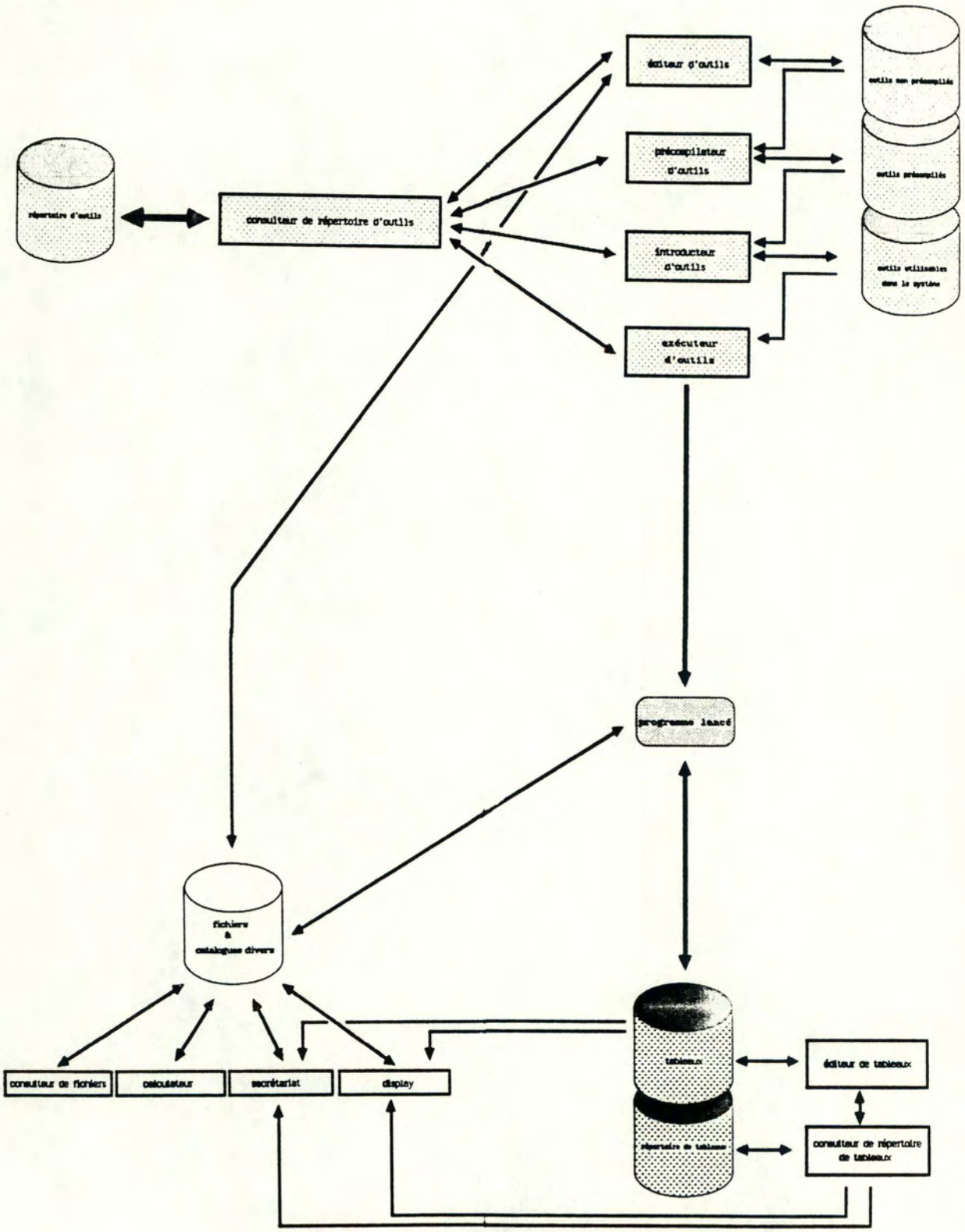
Ce composant est du type traitement de texte, mais chose fondamentale : on peut lui communiquer des documents qui proviennent de diverses sources (tableaux, graphiques, outils, production d'un outil ...), et les intégrer dans le texte. Nous sommes donc ici dans un cas de poste multifonction intégré.

Chapitre quatrième

Architecture théorique

Ktêma eis aei
<Thucydite, Guerre du péloponèse (I,22)>

§ 4.1 Schéma général



Il est à remarquer que les flèches indiquent uniquement les canaux de transmission des informations entre les différents composants du système, y compris les zones de stockage . Il ne faut donc certainement pas y voir des relations du type utilise ou appelle.

§ 4.2 Détail de l'architecture

4.2.1 Répertoire d'outils

Le répertoire d'outils consiste en une unité logique de stockage des informations relatives aux outils : nom donné par l'utilisateur à l'outil, identifiant interne, lien entre le nom et l'identifiant, état de l'outil(non précompilé, précompilé, utilisable dans le système), version de l'outil et localisation physique des textes de cet outil.

De plus, nous y stockons les informations sur les boîtes à outils : nom donné par l'utilisateur, identifiant interne, et lien entre le nom et l'identifiant. Comme une boîte peut contenir des outils, à chaque identifiant de boîte on associe les identifiants des outils qui se trouvent dans cette boîte. Comme celle-ci peut contenir des boîtes, à chaque identifiant de boîte on associe également les identifiants des boîtes qui s'y trouvent incluses.

Une dernière sorte d'information est stockée : les liens des outils entre eux. En effet, comme expliqué au point 3.2.3.1, un outil peut utiliser un ou plusieurs autres outils, ce qui veut dire qu'il existe parfois un graphe (nous verrons qu'on peut se limiter à un arbre) de tous les outils nécessaires à l'exécution correcte d'une fonction.

Le répertoire est donc un fichier du type

```
(
(nom_outil, identifiant_interne_outil, (état, version, localisation)*)*,
(nom_boîte, identifiant_interne_boîte, (identifiant_interne_outil)*,
                                       (identifiant_interne_boîte)*)*,
(identifiant_interne_outil, (identifiant_interne_outil)*)*
)
```

(où * signifie une ou plusieurs occurances)

4.2.2 Editeur d'outils

L'éditeur d'outils, comme son nom l'indique, est destiné à travailler le texte d'un outil; il s'agit donc d'un traitement de texte. Il faut remarquer que nous avons également demandé un traitement de texte pour la partie "secrétariat" (voir point 3.3.3). Toutefois, si nous restons au niveau logique, ces deux composants sont nettements distincts, en effet, l'éditeur d'outils n'a de sens que s'il travaille sur des outils, ce qui n'est pas le cas avec le secrétariat.

L'éditeur d'outils permet également de produire soit un listing, soit un fichier [qui ne sera plus un outil !], ceci pour les besoins de mise au point ou de communication (éventuellement par le biais du secrétariat).

Il peut également créer un outil à partir d'un fichier, ce qui permet soit la communication d'outils entre possesseurs de systèmes sur micro ordinateurs différents, soit l'édition d'un texte qui n'est pas un outil, mais qui peut le devenir moyennant quelques modifications.

L'utilisateur qui désire éditer un outil le fait en plusieurs étapes :

1° il choisit l'outil au moyen du consulteur de répertoire, ou donne un nom de fichier

2° il l'édite

3° il le replace dans le répertoire au moyen du consulteur, ou le sauve dans un fichier

L'utilisateur, en parallèle avec l'éditeur, peut constamment utiliser le consulteur de répertoire pour trouver et visualiser les divers outils ainsi que leurs caractéristiques (caractéristiques = ce qui est nécessaire de connaître pour pouvoir utiliser un outil à partir d'un autre, voir point 4.2.3).

L'éditeur facilite au maximum la copie à partir d'autres outils, c'est-à-dire qu'en parallèle avec l'édition, on peut copier une section d'outil sélectionné via le consulteur de répertoire.

Au moment ou l'éditeur prend en charge un texte, il reçoit du consulteur de répertoire les caractéristiques de l'outil : si l'outil est déjà utilisé par d'autres, il demande s'il faut lui donner un nouveau nom, s'il faut écraser l'ancien outil, ou s'il faut créer un nouveau "release".

A la fin du travail, lors de la remise de l'outil modifié au consulteur de répertoire, celui-ci fait le nécessaire pour que tout soit cohérent (voir le point 4.2.6).

4.2.3 Précompilateur d'outils

Le précompilateur a pour fonction de traduire un outil créé ou modifié dans l'éditeur d'outils, de façon à obtenir un outil compilable (ou interprétable) par l'utilitaire utilisé sur le micro ordinateur hôte.

Pourquoi a-t-on besoin de traduire ? Parce qu'on a glissé certaines lignes dans le texte qui ne sont pas correctes au point de vue syntaxe pour le langage de base. Dans quels cas ? Pour obtenir un outil 1 qui utilise un autre outil 2, le plus simple serait de recopier le texte de l'outil 2 à utiliser dans le nouvel outil 1. Or, ceci n'est pas à conseiller en pratique à cause des pertes de place mémoire et des redondances non contrôlées.

Aussi, lorsqu'on veut utiliser un outil 2, on insérera dans le texte de l'outil 1 une ligne (éventuellement repérée par un signe conventionnel) qui donnera les caractéristiques de l'outil et de ses paramètres, de façon à ce que le précompilateur puisse la remplacer par un appel de procédure, ou tout autre méthode destinée à déclencher un segment de programme auxiliaire.

Le précompilateur devra aussi être construit en fonction des primitives nécessitées par les outils non standard, c'est-à-dire essentiellement des conversions des formats de input et output. Autrement dit, conversions tableau <-> array, case <-> real/integer/char/string, colonnes/lignes <-> record/array, etc... ; ou plus éventuellement des primitives "d'interaction" (terminal, clavier) si le langage de base n'est pas agréable à ce point de vue.

Le précompilateur ayant reçu la localisation de l'outil à précompiler (par le biais du consulteur de répertoire d'outils), renvoie (si tout s'est correctement passé) la localisation du texte précompilé au consulteur, qui rajoute un état à la liste des états de l'outil, dans le répertoire d'outils.

4.2.4 Introducteur d'outils

Il s'agit d'un mécanisme qui permet l'utilisation d'un outil (précompilé) dans le système. Il faut donc d'abord compiler, linker, etc..., si nécessaire. Le rôle de l'introducteur d'outils est aussi de prévenir le consulteur de répertoire que l'outil est maintenant utilisable dans le système. Le consulteur peut alors ajouter un nouvel état pour l'outil, dans le répertoire d'outils

Pour terminer l'introduction de l'outil dans le système, il faut qu'on puisse le déclencher interactivement et il faut donc pour cela exécuter une série d'actions, qui dépendront des caractéristiques du matériel sur lequel on travaille.

Par exemple, si on conçoit un système à partir d'un programme principal muni d'un menu, et que la sélection d'un élément du menu provoque l'appel de la routine correspondante au moyen d'un "case PASCAL", le travail de l'introducteur d'outils est :

1° insérer un nouveau choix dans le menu

2° ajouter un nouvel élément dans le "case" et l'appel correspondant

3° compiler, linker etc... le programme principal

Pour une configuration comme celle du Lisa, il suffit simplement d'appeler l'utilitaire "Install" pour que le programme vienne s'ajouter aux applications déjà utilisables.

4.2.5 Exécuteur d'outils

Le programme principal exposé au point 4.2.4 est un exemple simple d'exécuteur d'outil.

Il est toutefois plus intéressant de garder la même ligne d'idées tout au long d'un système. Nous avons à notre disposition le consulteur de répertoire d'outils, c'est lui qui se chargera de la présentation des outils et qui donnera à l'exécuteur d'outils les coordonnées de l'outil à exécuter. L'exécuteur d'outils n'aura plus alors qu'à lancer l'outil désiré.

Remarquons que nous ne nous occupons pas ici de la gestion de la concurrence des outils. (On peut demander l'exécution de plusieurs outils "simultanément". Lorsqu'un outil est activé, il reçoit automatiquement (par un utilitaire de bas niveau) une fenêtre pour y afficher ses outputs à l'écran. Si on sort de cette fenêtre, les commandes sont communiquées au programme qui possède la nouvelle fenêtre, et le programme de l'ancienne fenêtre est "gelé" jusqu'à ce qu'on y retourne. Ainsi l'exécuteur d'outils possède une fenêtre, et on peut lancer plusieurs programmes "simultanément" en revenant à la fenêtre de l'exécuteur d'outils). En effet, cette concurrence est fictive car quoique les fenêtres soient ouvertes simultanément, il n'y a toujours qu'une seule fenêtre active à la fois, c'est celle de l'outil qui travaille.

4.2.6 Consulteur de répertoire d'outils

Ce composant à un rôle très important dans notre système. En effet, chaque composant que nous avons déjà décrit a besoin des services du consulteur de répertoire. Ceci tient au fait qu'il s'occupe non seulement de visualiser le répertoire d'outils ainsi que les boîtes, mais qu'il s'occupe également de toute la gestion de ces outils et boîtes.

Nous relevons :

- présentation des outils ainsi que des boîtes à outils et des caractéristiques de ces outils.

-suppression, déplacement entre les boîtes etc...

-sélection d'un outil et communication

* à l'éditeur d'outils
* au précompilateur d'outils
* à l'introducteur d'outils
* à l'exécuteur d'outils

des coordonnées de cet outil.

- insertion d'un outil, quel que soit son état (exécutable, précompilable ...), dans le répertoire.
- vérification du fait que la modification d'un outil ne détruise pas la logique des outils qui l'utilisent, lors d'une demande de l'éditeur d'outils. Cela revient à dire gestion et vérification des liens entre outils. En effet, pour un outil nous pouvons avoir de multiples versions de cet outil, et dans différents états. Les versions sont dues à des modifications de fonctionnalité (outil1, outil2,...), à des corrections d'erreurs décelées (outil1.1, outil1.2,...), et à des adaptations à des configurations particulières (outil1.1/DEC, outil1.1/IBM,...). Toute modification d'une version d'un outil qui est utilisée par un autre outil nécessite généralement une modification de celui-ci (compilation, linkage).

Pour chaque outil, nous avons la version courante de cet outil, c'est celle qui est utilisée par défaut par le consulteur de répertoire d'outils. Le consulteur permet de choisir une version, qui devient courante.

4.2.7 Editeur de tableaux

Nous décomposons l'éditeur de tableaux en deux composants logiques : un éditeur du type XXXcalc, qui nous permet de travailler un tableau, et un éditeur du type SQL/DS qui permet de manipuler un ou plusieurs tableaux. Les points 3.1.2.2 et 3.1.2.3 ont déjà détaillé l'ensemble des actions possibles.

Il faut remarquer que l'on ne considère pas un éditeur de tableaux comme étant un outil. En effet, nous considérons que le statisticien en a le plus grand besoin et qu'il est important de le lui livrer "prêt à utiliser". De plus, comme nous l'avons déjà signalé, l'étape d'exploration des données emploie les mêmes fonctions (avec peut-être d'autres buts) quelle que soient la nature du problème et la philosophie statistique qui sous-tend l'analyse, ce qui nous permet justement de créer un "super outil" universel.

Comme dans le cas de l'éditeur d'outils, nous pourrons utiliser en parallèle le consulteur de répertoire de tableaux, les outils éventuellement désirés, ainsi, bien sûr que les instruments particuliers, et spécialement l'instrument de présentation.

N'oublions pas que tous ces outils peuvent communiquer entre eux, nous verrons comment au point 4.2.10.

4.2.8 Répertoire de tableaux

Le répertoire de tableaux est destiné à associer un nom de tableau à son identifiant interne, à localiser ce tableau et à stocker les liens entre dossiers et tableaux. Le répertoire est simple, car il n'existe pas de relation entre les tableaux. Nous n'avons pas de tableau virtuel comme dans le cas d'une base de données relationnelle.

Le répertoire ne porte que sur les données "on-line". En effet, comme nous travaillons ici sur un micro ordinateur, nous ne pouvons garder en mémoire un répertoire portant sur des disquettes que l'ordinateur ne peut contrôler.

Le répertoire de tableaux est donc un fichier du type

```
(
(nom_utilisateur, identifiant_interne_tableau, localisation)*,
(nom_utilisateur, identifiant_interne_dossier, (identifiant_interne_tableau)*,
                                                (identifiant_interne_dossier)*)*
)
```

A chaque branchement d'un support (bande, disquette), le répertoire de tableaux est mis à jour.

4.2.9 Consulteur de répertoire de tableaux

Outre son rôle de présentation des tableaux, le consulteur permet de sélectionner un tableau pour servir d'input soit à un outil, soit à l'éditeur de tableaux.

Par ce moyen, on peut également faire un double, ce qui permet notamment d'obtenir un tableau vide de valeur mais dont les formules de calcul sont déjà en place. On a alors un tableau vide préformaté. On peut aussi supprimer un tableau. Comme on a également des dossiers pour regrouper les tableaux, on peut créer, supprimer des dossiers, insérer un tableau dans un dossier ou l'en extraire, déplacer des tableaux de dossier à dossier etc... (voir le desk de Lisa ou de Star)

4.2.10 Fonctions de base et gestion buffer

Nous reprenons sous cette rubrique les instruments particuliers.Le calculateur extrait ses informations de la zone de stockage "fichiers & catalogues divers", il y a notamment des listes de nombres aléatoires, des formules de calcul etc....

Le secrétariat (édition de rapport) peut chercher des tableaux (via le consulteur de répertoire de tableaux) pour l'insérer dans un rapport, et on peut également y insérer tout fichier texte ou dessin.

De même, le composant de display peut afficher sous forme graphique (voir [A5, A7, A8]) les valeurs données par un tableau et stocker le dessin dans un fichier.

Nous constatons qu'aucun tableau n'est créé ni modifié par le biais de ces instruments.

Remarquons néanmoins un composant particulier : le consulteur de fichiers. Son rôle est identique à celui des autres consulteurs, mais pour des fichiers autres que des outils ou des tableaux. L'éditeur d'outils peut parfois y avoir recours pour l'insertion d'une section de code dans un outil.

Nous parlons de composants, d'outils qui communiquent : une façon agréable d'implémenter ce concept est le buffer (appelé clipboard ou pince à papier dans certains systèmes bureautiques). Dans chaque composant qui le nécessite, il existe une action possible qui consiste à envoyer des informations dans une zone déterminée de la mémoire : le buffer.

L'action complémentaire est de saisir ces informations. Le buffer est bien sûr polyvalent, c'est-à-dire qu'il accepte tous les types de données. Chaque fois qu'on saisit quelque chose dans le buffer, on vérifie si le type d'information est compréhensible pour l'application. Pour cela, il existe une zone spéciale, fixée, dans le buffer qui indique le type des données qui y sont actuellement stockées. Il est alors facile pour un programme de vérifier si ce qui se trouve dans le buffer correspond à ce qu'il attend comme type de données.

Chapitre cinquième

Architecture concrète

De nihilo nihil
‹Satires, III, 24 ›

§ 5.1 Matériel de base

Nous avons décidé de baser notre architecture concrète sur ce que le Lisa nous propose. Les détails techniques se trouvant en annexe A2, nous allons donc simplement citer ici les éléments qui nous intéressent dans cet ensemble software et hardware.

Un écran graphique de diamètre de 30, 5 cm nous garantit un espace et une qualité de display suffisants pour n'importe quelle application statistique. Les ProFiles nous donnent une capacité de stockage suffisante, qui peut être complétée par deux lecteurs de disquettes. La souris nous donne un moyen souple de communiquer avec le software disponible, bien plus agréable que l'utilisation d'un clavier. Le processeur, un MC68000 32/16-bits, est remarquablement rapide dans les essais qui ont été effectués. L'imprimante, d'une résolution supérieure à celle de l'écran convient à la production de rapports de qualité.

Mais tout ceci n'est pas suffisant pour retenir notre attention, ce qui fait l'originalité de Lisa, c'est le logiciel qui lui est attribué. L' "Office System" (release 2.0, 1983, qui correspond à une couche juste au dessus de l' "Operating System") aborde l'informatique selon un angle tout à fait particulier. D'abord, le système est basé sur le visuel et la manipulation directe. Ainsi les fichiers, les programmes sont présentés sous forme d'icônes qu'on peut sélectionner ou activer. Pour supprimer un programme, il suffit de sélectionner son icône et de la faire glisser jusque sur l'icône de la corbeille à papier... Voyez donc la documentation en annexe A2.

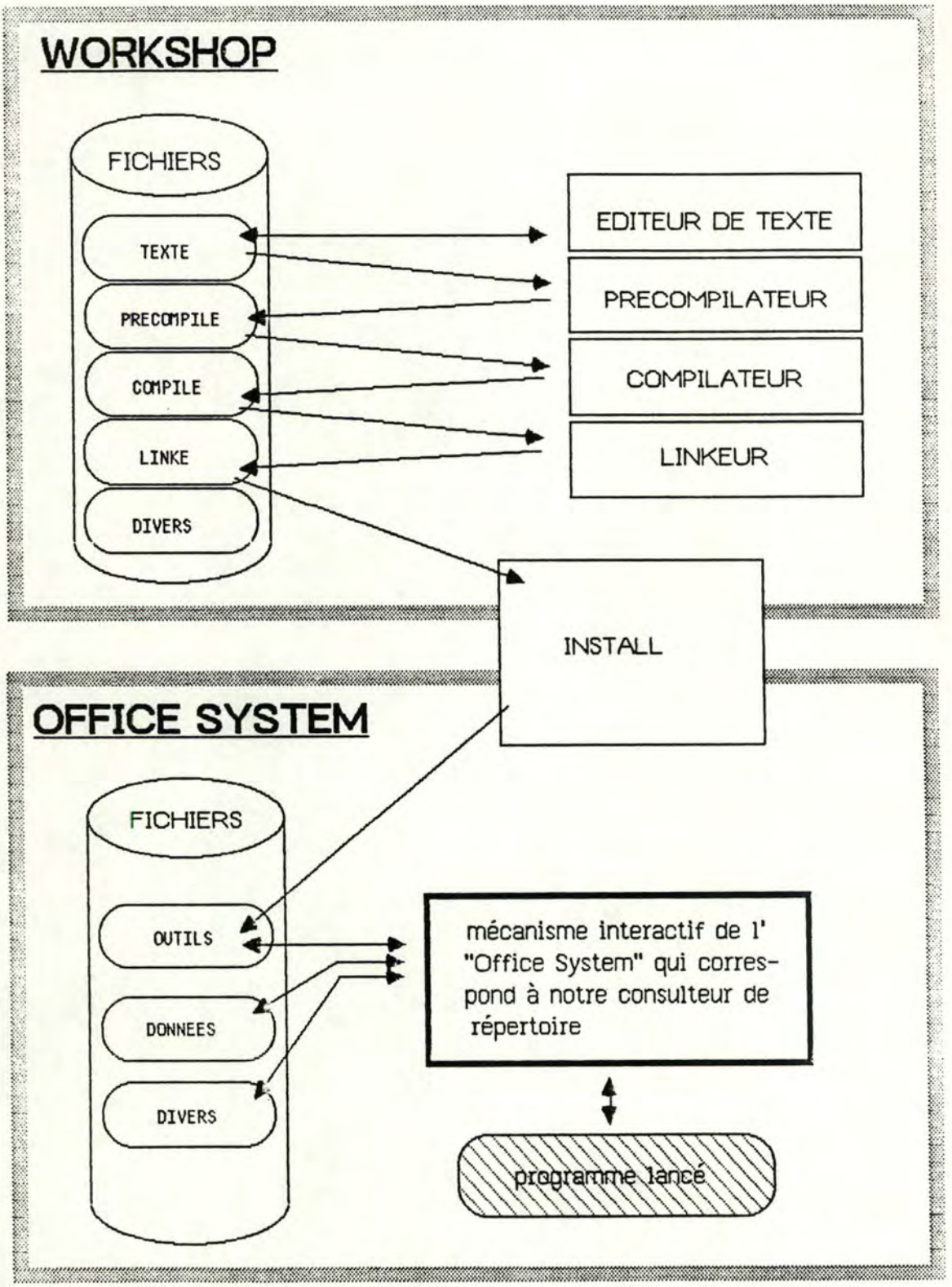
Remarquons surtout le système de fardes qui est proposé. Les fardes peuvent être vides, contenir des fichiers, contenir d'autres fardes... On peut ouvrir, fermer, déplacer, dupliquer une farde, et donc aussi tout ce qu'elle contient. Cela correspond à ce que nous désirons pour les boîtes à outils et les dossiers de données.

Nous voyons également le système de fenêtres, qu'on peut déplacer, agrandir etc ... Ce qui nous permet d'avoir plusieurs fardes ouvertes en même temps; le principe est d'application avec les programmes.

Ce qui est plus intéressant encore, ce sont les diverses applications offertes : Lisacalc (tableur), Lisawrite (traitement de texte), Lisagraph (graphiques), Lisadraw (éditeur de dessins). Malgré les défauts constatés à l'usage,(aucun de ces outils n'est techniquement complet) ils présentent une caractéristique qui fait leur force : l'intégration. Ils peuvent communiquer très facilement, au moyen du "copie et colle". On peut copier une partie des valeurs du tableur, cette copie est automatiquement placée sur la pince à papier, l'application graphique peut alors coller, ce qui signifie recopier dans sa zone de données ce qui se trouve sur la pince à papier. Le procédé est possible entre toutes les applications.

L' "Office System" et ses applications ne nous permettent pas de programmer, or ce point est essentiel pour le statisticien. Nous disposons heureusement d'une autre facette du Lisa : le "Workshop". Le "Workshop" présente l'aspect classique du micro ordinateur : directoire, éditeur de texte, compilateur, etc ... Nous pouvons donc y programmer comme sur n'importe quel micro ordinateur. Nous disposons de plus, de primitives qui nous permettent de créer des programmes avec le "look" Lisa. Malheureusement, la seule voie de communication entre l' "Office System" et le "Workshop" est l'installation d'un programme du "Workshop" comme nouvelle application de l' "Office System".

Nous pouvons visualiser l'existant comme suit :



(sans entrer dans les détails, les flèches indiquent un flux de données)

§ 5.2 Architecture

Lisacalc, Lisawrite, Lisagraph, Lisadraw, quoique étant dans l'optique générale du système proposé,nécessitent quelques améliorations : Lisawrite doit pouvoir inclure des dessins ou des graphiques dans son texte pour devenir un bon poste de secrétariat. Lisadraw doit posséder quelques possibilités supplémentaires, notamment pour la manipulation de texte dans les dessins. Lisagraph doit admettre, par exemple, d'afficher des courbes sans mettre de signe à chaque point et d'autres types de représentation (comme la fourchette). Lisacalc est un tableur classique, et comme nous l'avons exposé aux précédents chapitres , ceci ne suffit pas. Il faut donc le modifier, voire le remplacer, de façon à obtenir le manipulateur de tableaux désiré.

En ce qui concerne les outils particuliers, les tableaux et la gestion des tableaux, nous pouvons nous contenter de l'architecture de Lisa, avec les réserves émises.

Par contre, en ce qui concerne les outils, un travail plus important est à effectuer. Nous sommes obligés, par l'architecture Lisa, de faire une distinction entre le "Workshop" et l' "Office System". Dans le "Workshop", il faut constituer de toutes pièces un répertoire d'outils et un consulteur de répertoire. Le consulteur de répertoire consiste en un programme de commandes qui permet de lancer aussi bien des utilitaires que des programmes utilisateurs. Ce programme boucle jusqu'à la fin de la session (ou quand ce n'est pas possible, est activé quand on veut donner une commande). Il permet ainsi de contrôler les éditions, compilations, précompilations, linkages et installations dans l' "Office System". Avant d'exécuter la commande, il vérifie si les conditions d'exécution sont réalisées (notamment avant d'admettre de lancer une édition, il vérifie si le programme n'est pas utilisé par d'autres programmes); après le déroulement de l'utilitaire demandé par la commande, le répertoire d'outils est actualisé.

Par exemple, la commande "delete" produit la succession d'étapes suivante :

1° Quel outil deleter ? (réponse : toto/compilé)

2° Vérification si toto est utilisé par d'autres programmes.

si non => ok
si oui => production d'un message signalant le nombre et le nom des programmes qui utilisent toto et demande de confirmation.

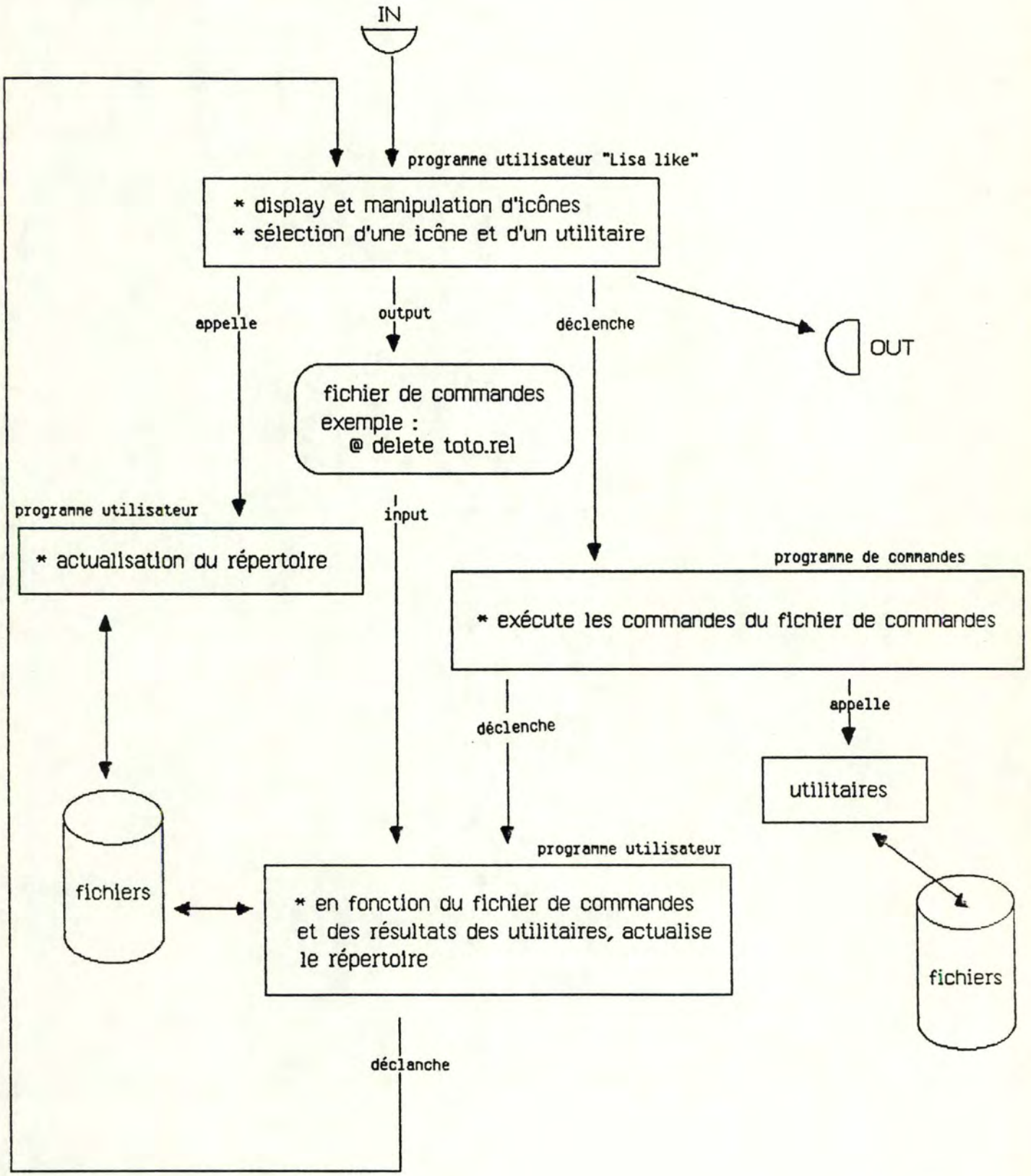
3° si ok ou si confirmation, passage à l'utilitaire qui correspond à la commande delete. On passe en paramètre le nom du fichier à deleter (ce nom est bien sûr stocké dans le répertoire).

4° Suppression de l'état "compilé" pour l'outil toto, dans le répertoire, si tout s'est correctement passé.

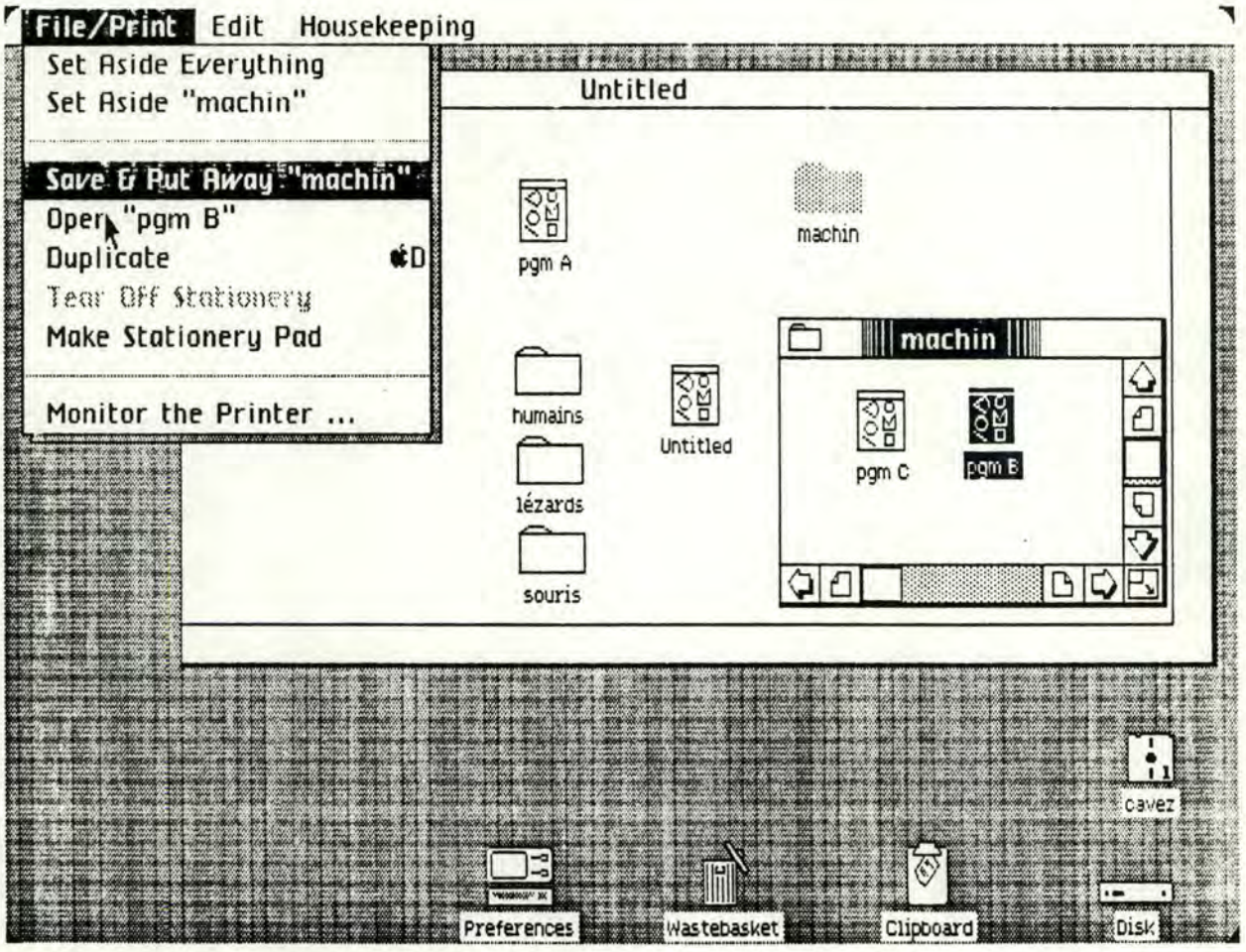
La même logique est applicable à toute commande.

Le consulteur de répertoire est amélioré par un programme qui permet de visualiser les outils (sous forme d'icônes avec possibilité d'afficher les caractéristiques (buts, spécifications) des outils). On peut alors manipuler directement les icônes des outils et des boîtes à outils, avec répercussion sur le répertoire d'outils, comme dans l' "Office System". On augmente également la convivialité du consulteur en cachant la syntaxe rébarbative des utilitaires par des manipulations d'objets à l'écran. Si on place l'icône d'un outil (toto/texte...) sur l'icône d'un utilitaire (compilateur...), le programme produit un fichier de commandes pour cet utilitaire et termine son exécution en laissant la main au programme de commandes qui lit ses ordres dans le fichier de commandes avant de relancer la boucle.

Schéma de fonctionnement du consulteur de répertoire :



Le seul problème impossible à éviter c'est la séparation (arbitraire à notre sens !) entre l' "Office System" et le "Workshop". Ainsi, rien ne sert d'exécuter les programmes dans le "Workshop" puisqu'on ne dispose pas des données acquises dans l' "Office System". Il faut donc que dans l' "Office System" on ait également un consulteur de répertoire d'outils. Or il n'y a plus de compilateur, linkeur etc... On peut donc se contenter de ce qui existe déjà.

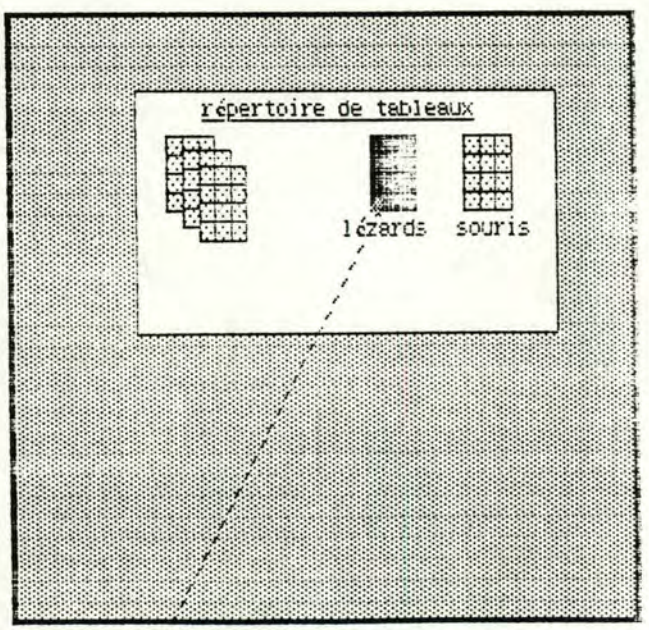


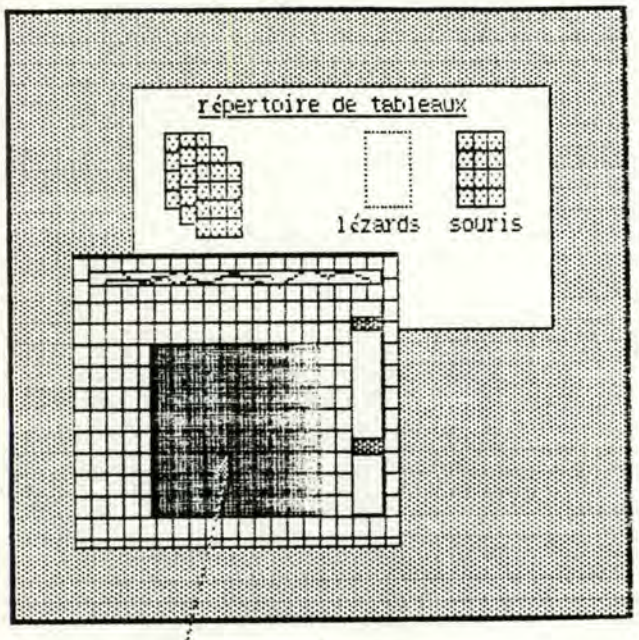
Mais un programme détruit (jeté à la poubelle) en "Office System" n'est pas pour autant détruit en "Workshop". Il faut donc, pour conserver la cohérence des répertoires d'outils, effectuer l'écho de chaque modification importante dans l'autre facette du système. Un travail à faire serait d'implémenter une couche supplémentaire entre le système d'exploitation et les utilitaires de façon à obtenir un système global plutôt que deux systèmes cloisonnés.

En "Office System" on peut facilement manipuler les différents outils et les faire se succéder, à condition qu'ils écrivent et lisent dans le clipboard avec des structures de données standard. Ainsi, si on veut utiliser successivement les programmes A, B, C (où B et C exploitent respectivement les résultats de A et B), on fait :

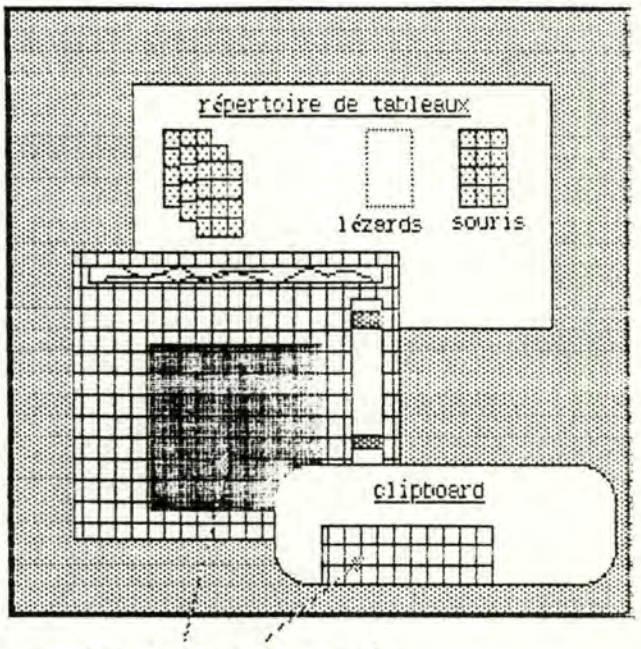
Sélectionner le tableau, copier la section qui nous intéresse.
Activer A (qui lit ses données dans le clipboard et y envoie ses résultats).
Activer B (qui lit ses données dans le clipboard et y envoie ses résultats).
Activer C (qui lit ses données dans le clipboard et y envoie ses résultats).
Coller du clipboard dans un tableau.

Voici d'ailleurs une illustration de ce principe :

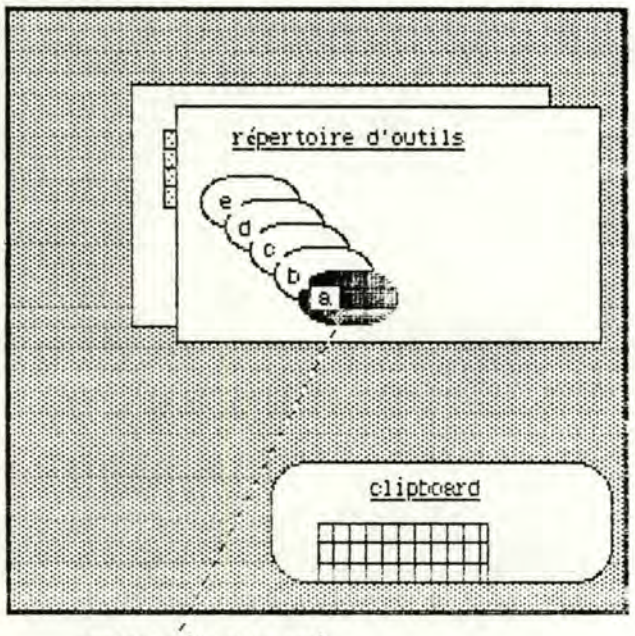




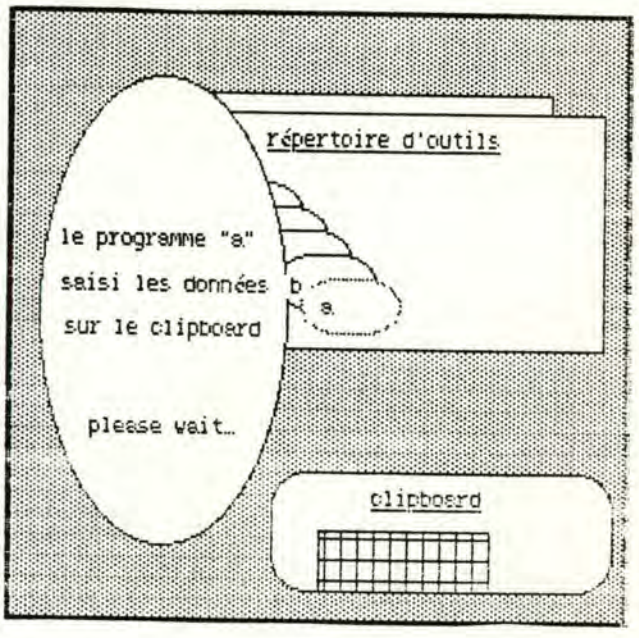
partie de tableau copiée

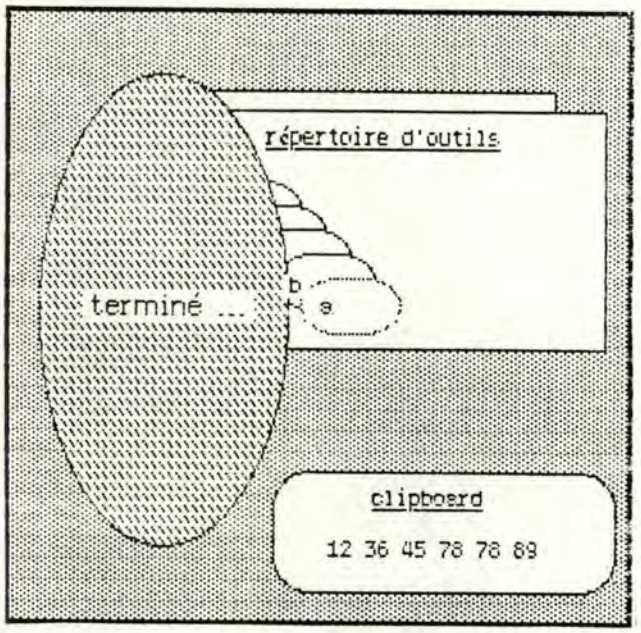


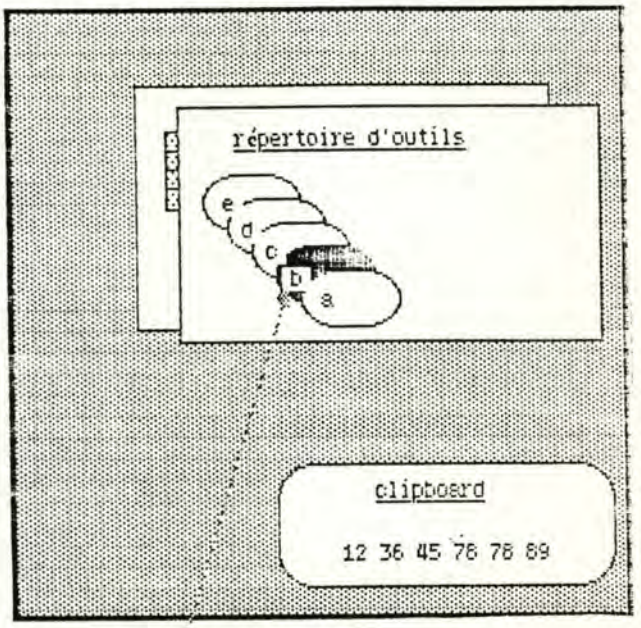
partie de tableau copiée



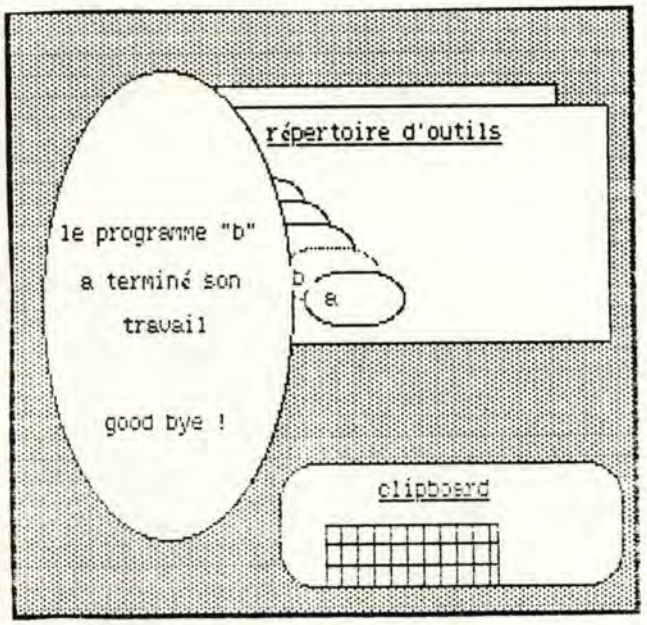
outil sélectionné

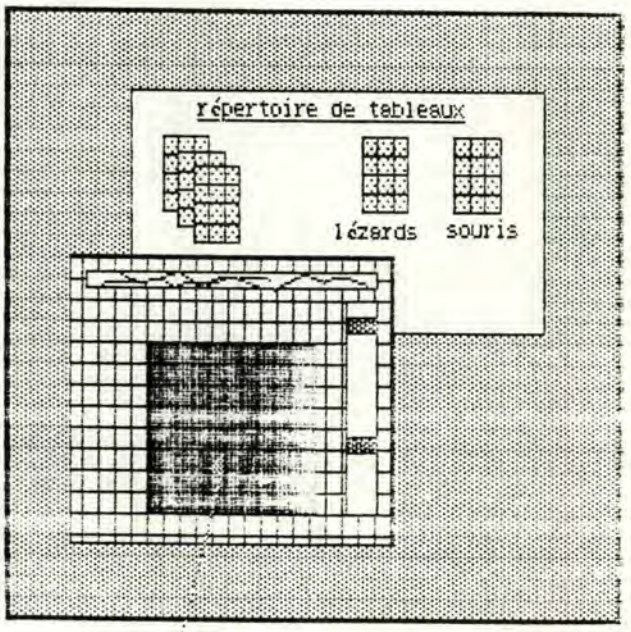






outil sélectionné





clipboard collé dans un nouveau tableau

Chapitre sixième

Définition d'un noyau

Un tiens vaut mieux que deux tu l'auras

§ 6.1 Introduction

Limités par le temps et la taille du mémoire, nous avons choisis de réaliser un tableur. En effet, plutôt que d'ébaucher un grand système ou de réaliser une maquette, nous avons préféré construire un programme plus petit, mais clair et facilement exploitable dans l'avenir.

Alors pourquoi choisir un tableur quand il en existe déjà plusieurs sur le marché ? Parce que les tableurs classiques ont certaines lacunes, et que nous avons essayé de les combler. Ainsi, nous avons implémenté un "Help" intégré, un système de formules qui portent sur des colonnes entières, des valeurs numériques qui peuvent être de plusieurs types, des cases où on peut ajouter une remarque (par exemple, noter si la valeur est fausse). Par contre, nous avons laissé de côté ce qui marche bien dans les autres tableurs, notamment le système de formules qui porte sur des cases et le formattage des valeurs, il est inutile de réinventer la roue.

Pour la réalisation du programme, nous avons dû nous rabattre sur le DEC 20/60 des facultés de Namur. En effet, le Lisa présentait des anomalies autant hardwares que softwares (disques défectueux, graphisme incomplet, ...). Nous n'avons donc pas mis l'accent sur le "look" puisque les outils de développement étaient indisponibles. Mais nous avons quand même tenté de faciliter au maximum l'usage du tableur. De plus, dans la construction du programme, toute la section "interaction écran/clavier" est clairement séparée et commentée de façon à faciliter une transposition sur un autre ordinateur ou une amélioration du display.

§ 6.2 Spécifications

Le tableur manipule un tableau de 50X50 cases, soit de A..AX colonnes et 1..50 lignes, mais ces dimensions doivent être modifiables par une seule intervention dans le programme. Ce tableau possède un nom de six caractères et le nom est modifiable.

Chaque case du tableau comprend trois zones, pour la valeur de la case, pour une remarque(un string de soixante caractères), pour une formule (un string de soixante caractères , sert à calculer la valeur de la case). La valeur de la case est au choix, du type : string[10], real ou integer.

Pour chaque colonne un type est fixé, ce type détermine le type des cases de la colonne . Exception, pour certaines colonnes (dites de type "mixte") où les cases peuvent être d'un type au choix, indépendamment les unes des autres.

On peut modifier le type d'une colonne, ainsi que le type des cases d'une colonne mixte. Chaque changement de type entraîne l'initialisation des valeurs des cases. Un changement d'un type d'une colonne entraîne le changement de type de toutes les cases de cette colonne.

A chaque colonne est associée une zone pour enregistrer une formule, cette formule permet de calculer les valeurs des cases de la colonne. Dans une formule, on peut utiliser les opérateurs ABS, ROUND, SIN , COS, EXP, LN, SQR, SQRT, +, -, *, /, (,), _ (opérateur moins unaire). Ces opérateurs portent soit sur le nom d'une colonne, soit sur une constante entière. Une commande permet de déclencher les calculs , les calculs se font de la première colonne à gauche (A) à la dernière colonne à droite (AX).

Le tableau complet (valeurs, formules, remarques, caractéristiques des colonnes) peut être sauvé sur un fichier du même nom que celui du tableau. En corolleire, on peut reprendre un tableau sauvé, à partir d'un fichier, et tout retrouver dans l'état précédant immédiatement le sauvetage. On peut également quitter le programme de manière à ne pas perdre l'état du tableau.

Dans un tableau on peut supprimer une ligne ou une colonne, on peut

également insérer une ligne ou une colonne à une position déterminée (remarque: le nombre total de lignes et de colonnes étant fixe, insérer veut dire prendre une ligne ou une colonne vierge et la déplacer).

On peut aussi recopier les valeurs d'une ligne ou une colonne dans une autre ligne ou une autre colonne, pour autant que les cases d'accueil soient du même type que les cases de départ.

A l'écran, le tableau est affiché sous la forme classique d'un tableur, colonnes verticales surmontées de leur nom, lignes horizontales précédées de leur numéro, affichage en dix caractères de la valeur de la case correspondante à l'intersection des lignes et des colonnes, un espace de quatre lignes (l'entête) en haut de l'écran est réservé à l'affichage du nom du tableau et des caractéristiques de la colonne et de la case sélectionnée. Les caractéristiques de la colonne sont le type et la formule , les caractéristiques de la case sont le type, la remarque et la formule. Une case (et donc une colonne) est sélectionnée lorsque le curseur est positionné dessus. On peut faire voyager le curseur dans tous les sens sur le tableau, mais il ne peut sortir des limites du tableau de par ces déplacements.

On peut déplacer une fenêtre (l'écran) sur le tableau, de façon à parcourir l'ensemble des lignes et des colonnes, l'entête affiche bien sûr successivement toutes les cases sélectionnées. On réserve deux commandes pour "geler & dégeler" l'entête et pour forcer l'affichage de l'entête gelée.

Pour introduire une valeur dans le tableau,il suffit de taper la valeur qui s'inscrit dans la case sélectionnée. Il est vérifié que la valeur introduite correspond bien au type admis dans la case; au cas où la réponse serait négative, un message d'erreur explique la raison du refus d'introduire cette valeur. Lorsqu'une valeur est introduite, la case suivante est automatiquement sélectionnée, on peut choisir le sens (horizontal ou vertical de ce déplacement).

En général, tout caractère tapé au clavier qui ne peut être utilisé génère un message d'erreur explicatif; de même, chaque fois que l'on tape "?" un message explicatif en rapport avec la position du curseur apparaît.

Pour changer une formule ou une remarque, il suffit de se positionner dans l'entête, et de remplir le champ réservé. On peut voyager de champ en champ dans l'entête.

Pour éviter le problème des messages systèmes qui viennent s'afficher à l'écran, on peut exécuter une commande de rafraîchissement de l'écran.

Remarque : les spécifications ont été effectuées en langage naturel plutôt que de manière algébrique (ou autre), de façon à être plus compréhensibles pour le lecteur. Elles ne sont pas destinées ici à un programmeur, mais à un utilisateur non informaticien.

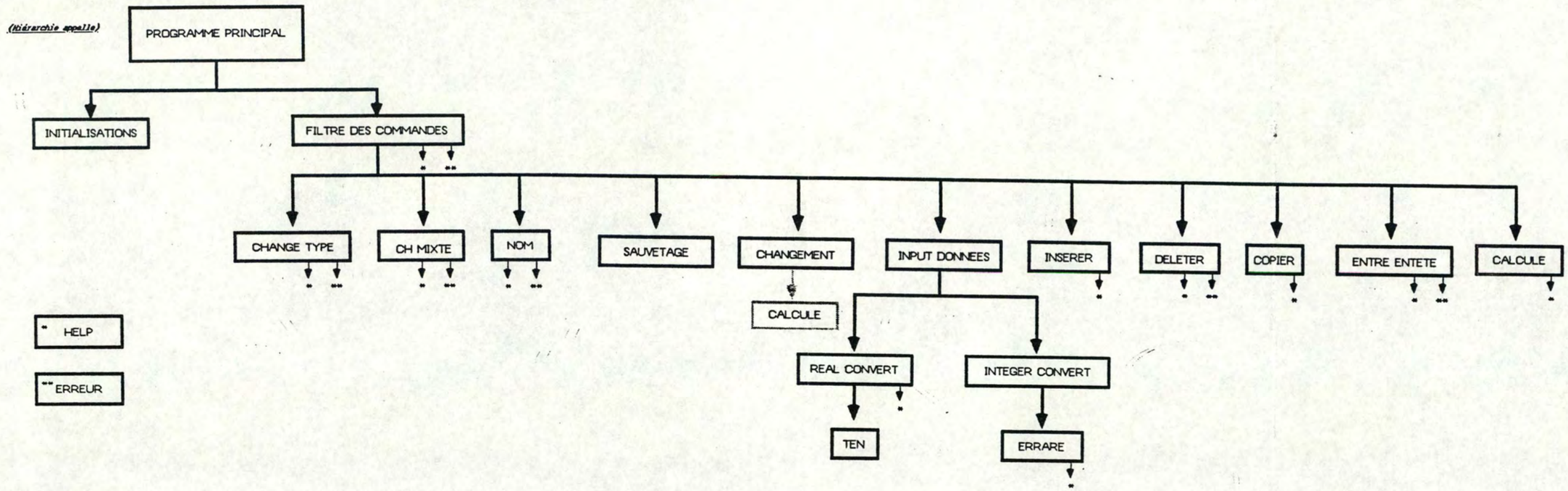
Chapitre septième

Réalisation du noyau

Vini, vidi, vici.

§ 7.1 Structure du programme

Nous avons schématisé ci-dessous la structure du programme, nous n'avons pas indiqué les appels de procédures d'affichage, de lecture et de gestion d'écran pour ne pas obscurcir inutilement la présentation. Les appels des routines d'aide et d'affichage de messages d'erreurs sont représentés par des * et ** .



§ 7.2 Algorithmes

Nous allons maintenant exposer les algorithmes utilisés pour l'implémentation du programme. Pour obtenir une bonne lisibilité de ces algorithmes, nous avons utilisé un pseudo langage qui se rapproche de la langue française, les différents blocs (qui en pascal se délimitent par begin et end) sont précédés d'un numéro qui indique leur niveau d'imbrication.

Pour la bonne compréhension, il faut d'abord citer les variables globales, et leur rôle. Nous avons tout d'abord deux tableaux, un tableau qui contient les valeurs introduites (on l'appelle le tableau des valeurs de base) et un tableau qui contient les valeurs à afficher après calculs (on l'appelle le tableau de display). Pour chaque case de ces tableaux, est indiqué le type de valeur qu'elle contient, la remarque qui lui est associée et la valeur proprement dite. Ensuite deux vecteurs stockent le type des colonnes et les formules qui portent sur les colonnes. Nous avons également deux variables qui mémorisent le nombre de lignes et de colonnes qui sont réellement utilisées dans le tableau, une variable (pas_arrêter) comme drapeau d'arrêt du programme. De plus, nous utilisons quelques variables globales pour la lecture et l'affichage, ainsi que pour la gestion d'écran.
Remarque: Pour chaque réponse donnée par l'utilisateur, si celle ci n'est pas acceptable, un message d'erreur est affiché et le programme n'effectue pas l'action demandée; ceci n'est pas repris dans les algorithmes, mais le lecteur de la source du programme verra que la structure n'en est pas pour autant modifiée.

programme principal

1 initialisations.
1 tant que pas_arrêter faire filtre_des_commandes.
1 effacer l'écran.

initialisations

1 pour toutes les cases des tableaux de display et des valeurs de base, initialiser à blanc : valeur, remarque et formule; initialiser le type des cases à string.
1 pour toutes les colonnes initialiser le type à string et la formule à blanc.
1 initialiser le nombre de lignes et de colonnes à zéro.
1 le nom du tableau est fixé à "défaut"
1 initialiser les autres variables globales, le terminal et afficher l'entête et le tableau.
(voir spécifications)

Filtre des commandes

1 lire un caractère
1 choisir le caractère parmi
? : 2 appeler la procédure Help;
δ..β : 2 déplacer la fenêtre;
ƒ..Ω : 2 appeler la procédure qui correspond à la commande donnée par le caractère;
Δ..■ : 2 effectuer la commande (pour de petites opérations, comme changer la valeur d'un boolean, nous n'avons pas créé de procédure);
autres:2 appeler la procédure d'introduction des données;
fin du choix.

Change type (modifie le type d'une colonne)

```
1 lecture du nom de colonne et du type voulu pour cette colonne.
1 si la réponse au type voulu est "?" alors 2 appeler Help
                                       sinon
   2 changer le type de la colonne.
   2 pour toutes les cases de cette colonne, et pour les deux tableaux,
     faire
     3 changer le type de la case.
     3 initialiser la valeur selon le type.
   2 afficher le nouveau tableau.
```

Ch mixte (modifie le type d'une suite de cases d'une colonne mixte)

```
1 si le curseur n'est pas sur une colonne mixte alors 2 message d'erreur
                                                  sinon
   2 lire l'intervalle des lignes dont il faut changer le type
   2 lire le type voulu
   2 pour toutes les cases qui sont dans l'intervalle faire
     3 changer le type
     3 initialiser la valeur selon le type
```

Changement (remplace le tableau par un autre, lu dans un fichier)

```
1 demande de confirmation (on perd le tableau s'il n'a pas été sauvé).
1 si confirmé alors
   2 lire le nom du nouveau tableau (c'est le nom du fichier).
   2 initialiser les variables. {à partir de maintenant on lit dans le fichier}
   2 lire les dimensions du tableau
   2 pour toutes les lignes et toutes les colonnes
      3 lire le type de case, la valeur, la remarque, la formule
   2 pour toutes les colonnes, lire le type et la formule
      1 calculer les formules et afficher le tableau
```

Deleter (supprime une ligne ou une colonne)

```
1 demande le numéro de ligne ou le nom de colonne à supprimer.
1 si la réponse est entre les bornes du tableau alors
   2 initialiser toutes les cases de la ligne ou de la colonne.
   2 pour une colonne, initialiser le type et la formule.
   2 placer cette ligne ou cette colonne en fin de tableau.
1 afficher le tableau
```

Copier (une ligne ou une colonne dans une ligne ou une colonne)

```
1 demande le numéro de ligne ou le nom de colonne de départ.
1 demande le numéro de ligne ou le nom de colonne de départ.
1 si la réponse est entre les bornes du tableau alors,
    en restant dans les dimensions du tableau,
   2 vérifier que toutes les cases sont deux à deux du même type.
   2 recopier chaque case de départ dans la case d'arrivée.
   2 pour une arrivée dans une colonne recopier la formule et le type
     s'ils existent.
1 afficher le tableau
```

Insérer (une ligne ou une colonne)

1 demande le numéro de ligne (le nom de colonne).
1 demande s'il faut insérer à gauche ou à droite (en haut ou en bas).
1 si la réponse est entre les bornes du tableau alors,
  2 si le tableau n'est pas entièrement occupé,
    3 insérer une ligne ou une colonne et décaler la suite du tableau.
1 afficher le tableau

Sauvetage

1 écrire (dans un fichier de même nom que le tableau) les dimensions du tableau.
1 pour toutes les lignes et les colonnes, écrire les cases (type, valeur, formule, remarque).
1 pour toutes les colonnes écrire le type et la formule.

Input données

1 lire le string à l'écran.
1 dans le tableau des valeurs de base, inscrire la traduction du string selon le type de case, pour autant que la traduction n'ait pas trouvé d'erreur syntaxique. La case à remplir est repérée par les coordonnées du curseur à l'écran, en sachant quelle case se trouve en chaque point de l'écran.

Entre entête

1 lire un caractère.
1 choisir le caractère parmi
   ?   : 2 appeler la procédure Help;
   ∂..ß : 2 déplacer le curseur sur une autre ligne de l'entête;
   ƒ   : 2 sortir de la procédure;
   autre: 2 lire le string et le placer dans la zone de stockage de la formule de la colonne, de la formule de la case ou de la remarque de la case; selon la position du curseur sur l'écran.

Help

1 nettoyer l'écran.
1 afficher une section de texte selon paramètre.
1 restaurer l'écran.

Erreur

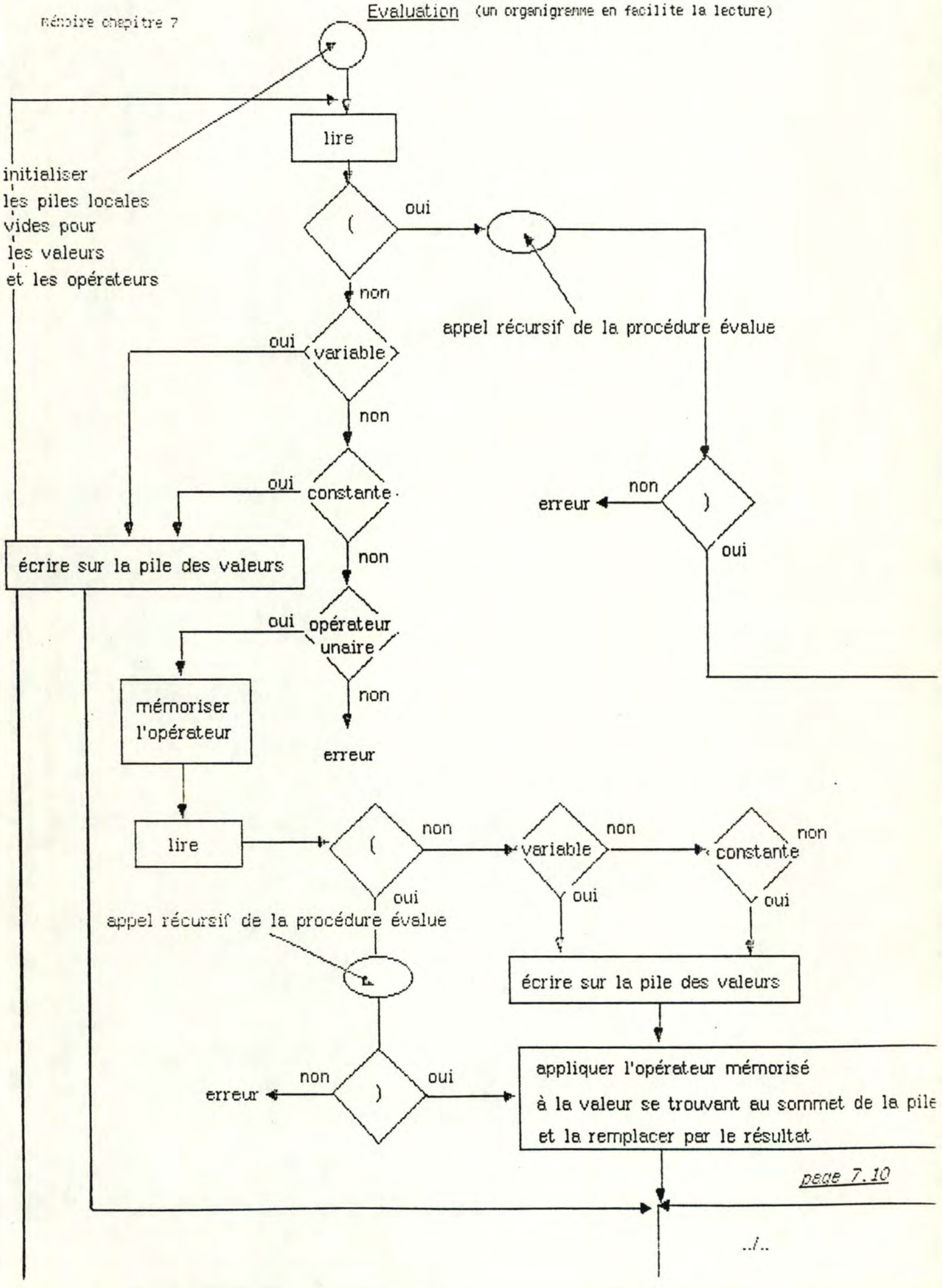
1 afficher une phrase sur la dernière ligne, selon paramètre.
1 nettoyer la ligne après quelques secondes.

Calcule (évaluation des formules provenant des colonnes)

(se décompose en deux parties: compactage de la formule, évaluation de la formule compactée pour toutes les lignes de la colonne)

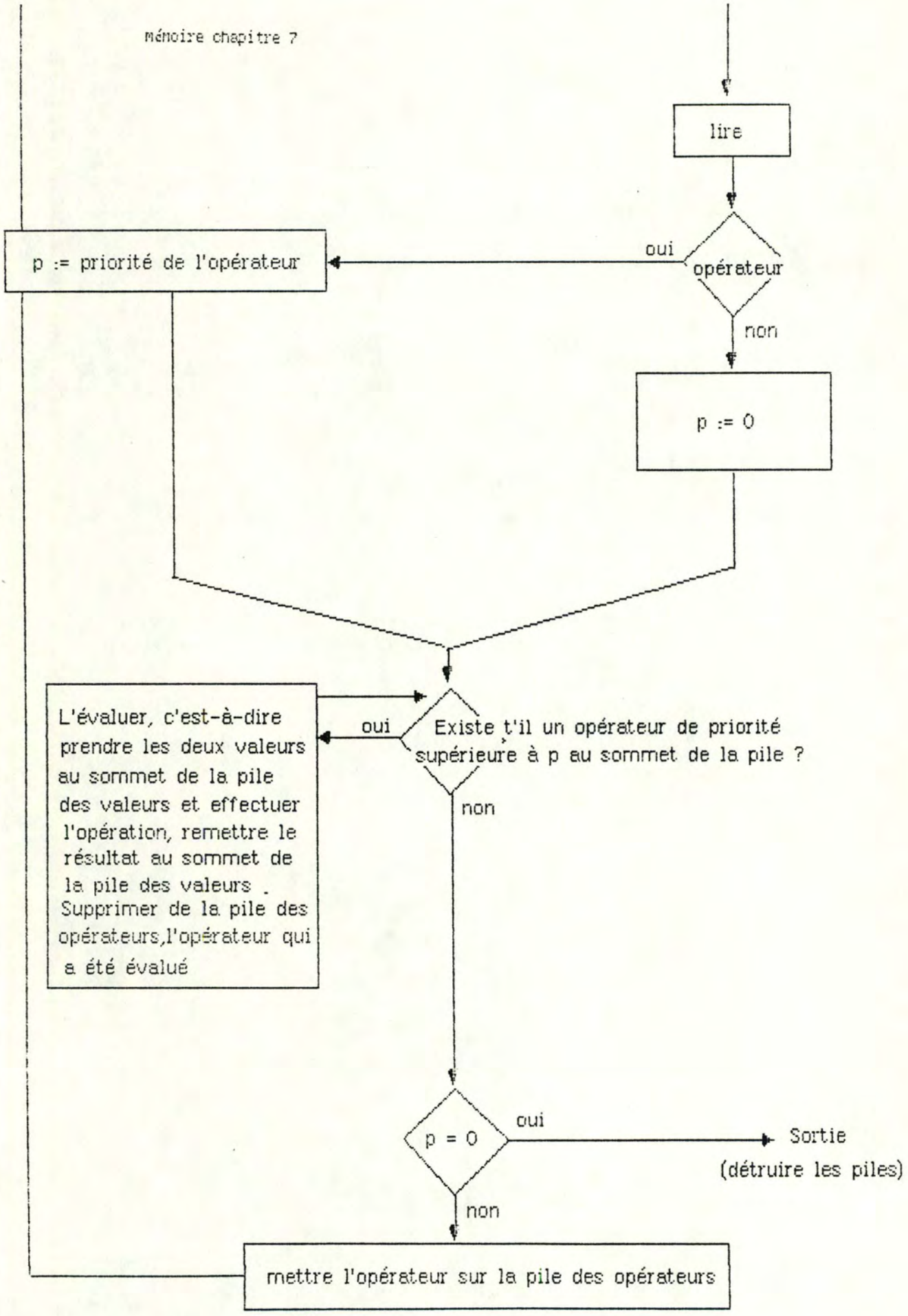
1 remplir le tableau de display avec le tableau des valeurs de base.
1 pour toutes les colonnes, si une formule existe, alors
  2 (compactage de la formule)
    analyser la formule, caractères par caractères, de façon à remplacer chaque caractère ou groupe de caractères par une seule valeur dans un array. (par exemple ['S']['I']['N'][' ']['2']['5'] est remplacé par ['S'][25])
  2 (évaluation)
    pour toutes les cases de la colonne, remplir le tableau de display avec la valeur qui est évaluée, en remplaçant dans l'évaluation chaque référence à une colonne par une référence à la case de la colonne qui correspond à la ligne traitée.

Evaluation (un organigramme en facilite la lecture)



page 7.10

../..


lire
opérateur
oui
p := priorité de l'opérateur
non
p := 0
Existe t'il un opérateur de priorité supérieure à p au sommet de la pile ?
oui
L'évaluer, c'est-à-dire prendre les deux valeurs au sommet de la pile des valeurs et effectuer l'opération, remettre le résultat au sommet de la pile des valeurs . Supprimer de la pile des opérateurs,l'opérateur qui a été évalué
non
p = 0
oui
Sortie
(détruire les piles)
non
mettre l'opérateur sur la pile des opérateurs

§ 7.3 Mode d'emploi du programme

Pour lancer le programme il suffit de donner la commande "EXECUTE q.pas".

Chaque fois que l'utilisateur hésite sur la prochaine action à exécuter, il peut taper "?", ce qui provoquera l'émission d'un écran d'aide, adapté au niveau de commandes où il se trouve.

Voici le détail des commandes qui peuvent être données:

(le signe "c-" signifie qu'il faut enfoncer la touche control en même temps que la lettre au clavier).

C-B déplacer le curseur d'une colonne vers la gauche
C-F déplacer le curseur d'une colonne vers la droite
C-N déplacer le curseur d'une ligne vers le bas
C-P déplacer le curseur d'une ligne vers le haut
C-A C-B déplacer le curseur d'une page vers la gauche
C-A C-F déplacer le curseur d'une page vers la droite
C-A C-P déplacer le curseur d'une page vers le haut
C-A C-N déplacer le curseur d'une page vers le bas
C-X C-B déplacer le curseur au bout du tableau à gauche
C-X C-F déplacer le curseur au bout du tableau à droite
C-X C-P déplacer le curseur au bout du tableau en haut
C-X C-N déplacer le curseur au bout du tableau en bas
C-H déplacer le curseur pour aller dans l'entête
C-N OU C-P déplacer le curseur de ligne à ligne dans l'entête
C-X C-Z quitter le programme
C-X C-X sauver le tableau
C-X C-V loader un nouveau tableau
C-X C-A changer le nom du tableau

$CCC copier une colonne dans une colonne
$CCL copier une colonne dans une ligne
$CLL copier une ligne dans une ligne
$CLC copier une ligne dans une colonne
$ICG insérer une colonne à gauche d'une colonne
$ICD insérer une colonne à droite d'une colonne
$ILH insérer une ligne au dessus d'une ligne
$ILB insérer une ligne au dessous d'une ligne
$SC supprimer une colonne
$SL supprimer une ligne
C-E changer le type d'une colonne
C-A C-E changer le type d'une rangée de cases d'une colonne mixte, quand le curseur est positionné sur cette colonne
C-L redisplay l'écran
C-V lance le calcul selon les formules
C-D change le sens du déplacement automatique du curseur lors de l'introduction de données
C-Y switche l'automatisme du display de l'entête
C-Z force l'affichage de l'entête

Remarque: pour les formules, on peut utiliser +, -, /, *, _, (,), sin, cos, abs, round, ln, exp, sqr, sqrt. On peut bien sûr combiner ces opérateurs au moyen des parenthèses. Les opérateurs portent soit sur des constantes, soit sur des noms de colonnes.

Le volume des annexes est disponible à la
bibliothèque de l'Institut d'Informatique.