



**QL**

**QL Archive**

©1983 PSION LIMITED  
by Dick de Grandis-Harrison (*Psion Limited*)

# CHAPITRE UN

## QL ARCHIVE

---

**QL ARCHIVE** est un système de gestion de base de données (SGBD) qui vous autorise à créer et à gérer des systèmes de fichiers, quel que soit le type d'information choisi. Vous êtes libre de décider de quelle manière cette information sera stockée et retrouvée.

Vous découvrirez rapidement comment utiliser **ARCHIVE** pour créer des systèmes simples par fiches, tels que listes d'adresses ou répertoires de clients. Quand vous maîtriserez la création de tels systèmes, vous pourrez songer à développer des bases de données relationnelles multifichiers entre lesquelles l'information est partagée. Comme par exemple, dans les fichiers d'Achats et de Stocks.

Les imprimés et les comptes rendus donnés à partir des informations du fichier, peuvent être présentés au format d'édition **d'ARCHIVE** ou à votre propre format.

L'une des caractéristiques les plus puissantes est la structure des commandes. Une fois que vous avez créé un fichier, et stocké quelques informations dedans, ces commandes sont utilisables pour accéder à un enregistrement précis, faire des recherches et des sélections ou bien afficher les informations contenues dans le fichier, dans un ordre donné.

Les commandes s'associent pour former un langage de programmation très performant, similaire au **SUPER BASIC**, et elles peuvent être employées pour construire une multitude d'applications spécifiques.

A tout moment, vous serez guidé et informé par une série de courts messages qui ne vous laisseront jamais dans le doute sur les options possibles ou sur ce que vous devez faire. Si vous avez besoin de renseignements supplémentaires, servez-vous du fichier **AIDE**. Vous pouvez demander de l'aide à chaque étape. Peu importe ce que vous êtes en train de faire. Automatiquement, des informations plus explicites vous seront données.

La réelle puissance **d'ARCHIVE** devient évidente quand vous écrivez, vos propres procédures dans le langage de commande. Vous pouvez créer une procédure dotée d'un nom pour faire exactement ce que vous désirez, puis l'utiliser comme si elle devenait partie intégrante des commandes de base **d'ARCHIVE**.

Les mécanismes de création et de modification d'un programme sont assistés par un éditeur de procédure qui, avec l'éditeur de ligne disponible à tout moment, rend l'édition simple et aisée.

Les fichiers de données exploitent eux-mêmes des enregistrements de longueur variable. Ce qui aboutit à une utilisation plus rationnelle de l'encombrement de la mémoire et de la cartouche du Microdrive<sup>1</sup>, et de plus simplifie la création du fichier. Vous n'avez même pas à décider à l'avance de la taille utile à donner aux enregistrements.

Ce manuel comprend bon nombre d'exemples de travail. Essayez-les pour voir la variété des options possibles. Ils contiennent des procédures d'ordre général, insérables dans vos propres programmes.

Si, cependant, vous n'étiez pas sûr de la marche à suivre, n'oubliez pas que la touche **(F1)** peut vous aider. Souvenez-vous que vous pouvez annuler n'importe quelle opération partiellement effectuée (comme l'entrée d'un nombre au clavier ou l'utilisation d'une commande) en pressant **(ESC)**.

**ARCHIVE** a été conçu pour donner la souplesse la plus grande possible. En conséquence, il ne peut vous offrir que le même type d'aide, dans la sélection des options, que les autres programmes du QL. Si vous êtes un utilisateur peu averti, lisez le **GUIDE DU DEBUTANT** avant de tenter d'écrire des programmes pour **ARCHIVE**.

---

<sup>1</sup> **Microdrive** : Microlecteur de cartouches

# CHAPITRE DEUX

## ENTRONS DANS LE VIF DU SUJET

---

### CHARGEMENT DE QL ARCHIVE

Chargez **QL ARCHIVE** comme indiqué dans l'introduction aux programmes du QL. Une fois chargé, **ARCHIVE** affichera le message suivant :

```
CHARGEMENT DE QL ARCHIVE
Base de données

Version X.XX

Copyright (C) 1984 PSION Ltd.
Tous droits réservés
```

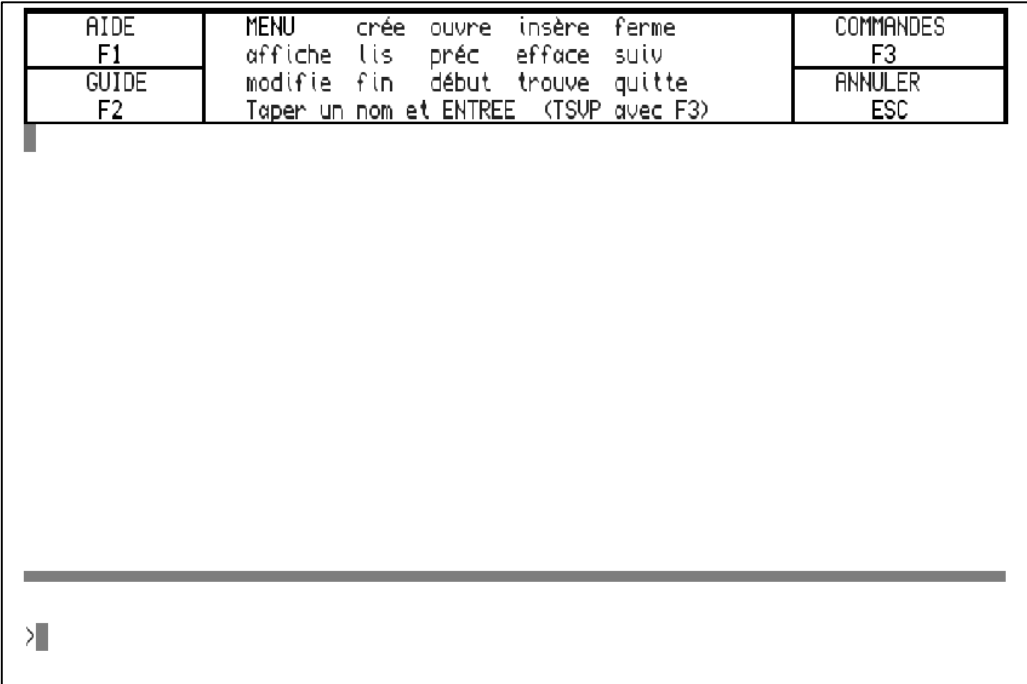
X.XX représente le numéro de la version, par exemple 2.00. Le programme attendra quelques secondes avant de démarrer.

Le fichier **AIDE** n'est pas chargé en mémoire en même temps que le programme. Il est simplement lu à partir de la cartouche **ARCHIVE** en cas de besoin. **Vous ne devez donc pas retirer la cartouche ARCHIVE du Microdrive 1 si vous avez l'intention de faire usage du fichier AIDE.**

### APPARENCE GENERALE

Après chargement **d'ARCHIVE**, l'écran aura l'aspect de la figure ci-dessous. Il s'agit là de l'écran type.

AIDE F1	MENU crée ouvre insère ferme affiche lis préc efface suiv	COMMANDES F3
GUIDE F2	modifie fin début trouve quitte Taper un nom et ENTREE (TSUP avec F3)	ANNULER ESC

Affichage type sur moniteur 80 caractères.

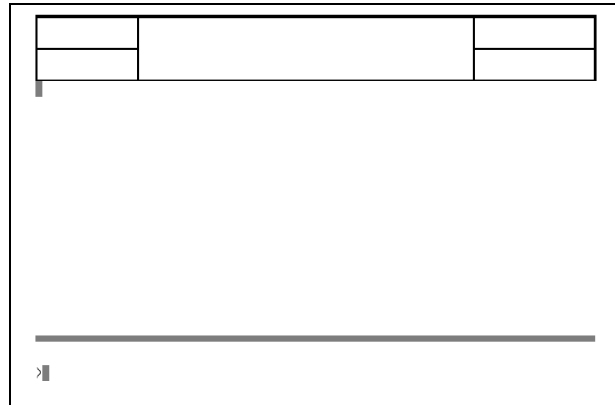
Si vous utilisez un téléviseur, l'écran se dispose un peu différemment, car en principe, une télévision n'est pas capable d'afficher 80 caractères par ligne. **ARCHIVE** ne montre donc que 64 caractères.

Cet écran est divisé en trois secteurs : la zone d'affichage, la zone de travail et la zone de contrôle.

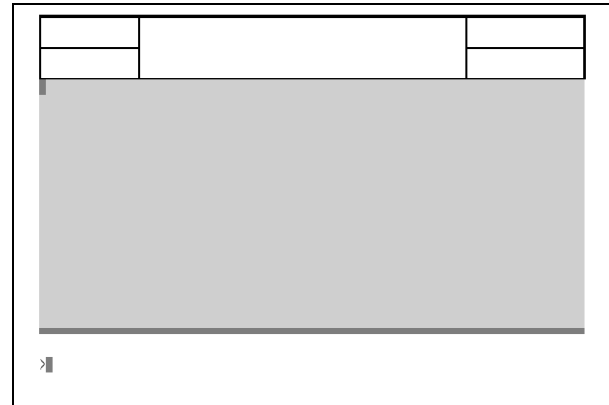
### Zones d'affichage et de travail

C'est dans la zone d'affichage, comme l'indique son nom, que toute information produite par **ARCHIVE** est visualisée.

La zone de travail utilise les quatre dernières lignes de l'écran. C'est ici que toutes les commandes que vous entrez et les messages d'erreur apparaissent.



Zone d'affichage



Zone de travail

Ces deux zones fonctionnent toujours ensemble puisque les commandes entrées dans la zone de travail produisent leurs résultats dans la zone d'affichage.

Pour illustrer, tapez ce court programme :

```
que x = 13: tantque x >0:écris x: que x=x-1: ftantque (ENTREE)
```

Ce texte apparaîtra sur la première ligne de la zone de travail. Quand vous pressez la touche **ENTER**, les nombres de 13 jusqu'à 1 s'écrivent successivement sur les lignes de la zone d'affichage. La dernière ligne de cette zone reste vierge à l'exception d'un curseur rouge qui indique la prochaine position d'affichage. On a donc au total, 14 lignes occupées dans la zone d'affichage.

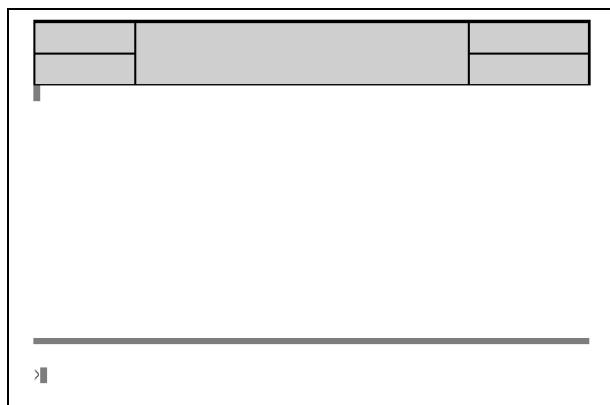
La commande

```
éponge (ENTREE)
```

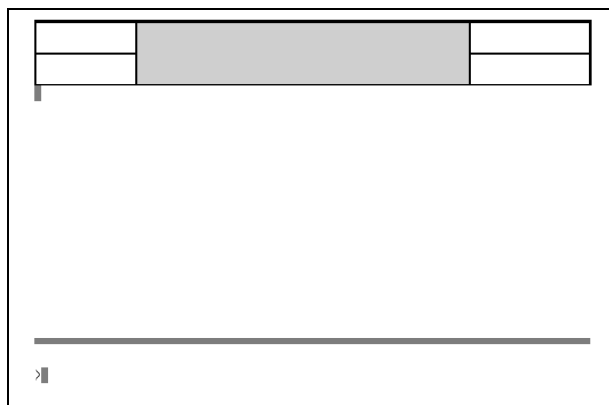
effacera complètement l'écran.

## ZONE DE CONTROLE

Elle occupe les quelques lignes du haut de l'écran et montre les options normales : **(F1)** aide, **(F2)** fait apparaître ou disparaître la zone de contrôle, **(ESC)** annule l'opération en cours et **(F3)** donne accès aux commandes.



La zone de contrôle



Les commandes

## UTILISATION DES COMMANDES

**Les commandes d'ARCHIVE forment un langage de programmation, et on doit taper leur nom en toutes lettres.**

Cela peut sembler long a priori, mais vous verrez plus tard comment créer des procédures qui vous permettront d'entrer des commandes par une simple pression de touche.

Il y a quatre listes de commandes que l'on peut visualiser en pressant **(F3)**. Si une liste de commandes est déjà visible, une pression sur **(F3)** affichera la suivante. Ces commandes s'utilisent simplement en tapant leur nom suivi de **ENTREE**. Cependant, quelques commandes nécessitent des paramètres supplémentaires.

Vous pouvez utiliser n'importe quelle commande, même si son nom n'apparaît pas à ce moment-là dans la zone de contrôle.

## LE MODE COMMANDE

L'association des zones de contrôle et d'affichage en une seule zone s'effectue par la commande **mode**. Soit en tapant `mode`, soit `mode 0` suivis de **ENTREE**. Essayez, tout ce qui sera tapé ou affiché se partagera à l'écran.

`Mode 1` redivise l'écran en trois zones.

Vous pouvez aussi utiliser la commande **mode** pour modifier le nombre de caractères affichables par ligne. Pour ce faire, **mode** doit être suivi de deux arguments séparés par une virgule. Le deuxième paramètre, qui peut être 4,6 ou 8, sélectionne une largeur de ligne de 40,64 ou 80 colonnes.

Essayez de taper `mode 0,4` pour combiner les trois zones de l'écran et ce sur 40 colonnes.

Notez que le '0' qui était pris par défaut à l'entrée de la commande **mode ENTREE** doit être spécifié si l'on donne un deuxième argument.

Tentez différentes combinaisons pour en voir les effets sur l'affichage. Terminez par une commande qui laisse l'écran divisé en trois zones, et choisissez le nombre de caractères qui visualise toutes les colonnes sur votre moniteur ou votre téléviseur.

# CHAPITRE TROIS

## LES FICHIERS DE QL ARCHIVE

---

### LES ENREGISTREMENTS ET LES CHAMPS

Un fichier **ARCHIVE** se comporte quelque peu comme un fichier manuel. C'est à dire une boîte qui contient une série de fiches munies de divers renseignements. Pour qu'un tel fichier soit utilisable, il doit y avoir des règles qui déterminent la place de chaque information.

Supposons que l'on ait un fichier de noms et d'adresses. On note en principe le nom des personnes en tête, suivi de l'adresse, puis du numéro de téléphone (éventuellement). Il suffit de feuilleter les fiches en ne lisant que la ligne du haut pour trouver rapidement le nom recherché. Si on notait le nom parfois en haut, parfois en bas, trouver la fiche serait une épreuve de force. Imaginons que l'on ait deux séries de fiches ; par exemple l'une de noms et adresses, l'autre de stocks. En principe on ne les garderait pas dans la même boîte. On aurait deux classeurs : "Fichier clients" et "Fichier Stocks".

Par l'analogie avec le système traditionnel de classement, on saisira aisément le fonctionnement **d'ARCHIVE**. Un fichier comme un répertoire, doit être identifié par un nom. Il se compose d'un ensemble d'enregistrements, et chacun d'eux joue le rôle d'une fiche de renseignements.

Comme dans une fiche, l'information dans chaque enregistrement est disposée de manière convenue. Les données individuelles (numéro de téléphone, par exemple) peuvent être conservées sur une zone spécifique de la fiche. Un enregistrement dans un fichier **ARCHIVE** s'organise de la même façon. Chaque détail est stocké sur une partie précise appelée **Champ**. Un enregistrement dans un fichier Clients, comme celui décrit ci-dessus, contiendrait un champ avec le nom, un champ avec l'adresse, un champ avec les rabais consentis, etc...

Si on s'en tenait là, utiliser un fichier **ARCHIVE** plutôt qu'un fichier manuel apporterait peu. Il y a cependant de nombreux avantages à se servir de fichiers informatisés. L'ordre alphabétique, dans lequel un fichier Clients est habituellement classé, permet de trouver rapidement la fiche d'un client donné.

Extraire un ensemble de fiches est moins évident. Supposons que vous vouliez envoyer une lettre à tous vos clients qui n'ont pas passé commande pendant les six derniers mois. Il serait fastidieux de consulter tout un fichier pour obtenir une telle liste. Avec **ARCHIVE** vous pouvez faire ce type de recherche grâce à quelques commandes simples. De plus, il est facile d'éditer sur imprimante l'ensemble des adresses sélectionnées.

Vous pouvez vous épargner beaucoup de temps et d'efforts en utilisant **ARCHIVE** pour stocker et manipuler vos enregistrements.

# CHAPITRE QUATRE

## EXAMEN D'UN FICHIER

---

Le meilleur moyen pour débiter dans le maniement d'ARCHIVE, c'est d'étudier le fichier de démonstration PAYS, fourni sur la cartouche ARCHIVE. C'est un fichier qui contient des informations sur les pays - Continent, Capitale, Monnaie, Population, Superficie et Produit National Brut par Habitant - .

La plupart des exemples des chapitres 4 et 5 font référence au fichier PAYS. Avant de vous en servir, faites-en une copie de la manière suivante : après chargement d'ARCHIVE, mettez une cartouche formatée dans le Microdrive 2 et tapez :

```
sauvegarde      (ENTREE)  
mdv1_PAYS_dbf  (ENTREE)  
mdv2_PAYS_dbf  (ENTREE)
```

Attendez que les deux Microdrives soient arrêtés ; soyez patient car le fichier est assez long, et sa duplication peut prendre un moment. Pour vos expériences, employez la copie qui est à présent sur la cartouche du Microdrive 2.

Désormais, nous n'écrivons plus **(ENTREE)** à la suite de chaque commande, mais rappelez-vous que vous devez toujours la valider par **(ENTREE)**.

La commande **lis** ouvre un fichier à la lecture, mais sans possibilité de le modifier, à l'inverse de **ouvre**. C'est une commande qui vous met à l'abri des modifications accidentelles quand vous vous contentez de regarder un fichier. Vous pouvez examiner la copie du fichier **PAYS** sur le Microdrive 2 en tapant :

```
lis "PAYS"
```

## AFFICHAGE D'UN ENREGISTREMENT

Pour regarder le premier enregistrement, tapez :

```
début  
affiche
```

N'oubliez pas de taper **(ENTREE)** après chaque commande, vous verrez alors apparaître à l'écran le premier enregistrement du fichier.

Notez que la première ligne indique le nom logique du fichier ; **ARCHIVE** donne automatiquement le nom "**maître**" à un fichier unique. Les noms logiques dont nous parlerons plus loin, sont en principe utilisés quand vous manipulez plus d'un fichier simultanément.



## EXAMENS D'AUTRES ENREGISTREMENTS

Quand vous avez lu le premier enregistrement, si vous voulez examiner le suivant, tapez :

`suiv`

et l'enregistrement suivant dans le fichier s'affiche à l'écran. L'utilisation de tout ordre simple met constamment à jour la zone d'affichage pour montrer l'enregistrement courant. Vous pouvez employer la commande `suiv` pour visualiser tous les enregistrements du fichier, un par un, jusqu'à la fin (sans dépasser le dernier).

Il y a trois autres commandes du même genre qui permettent de contrôler quel enregistrement du fichier est affiché :

<code>préc</code>	- affiche l'enregistrement précédent
<code>début</code>	- affiche le premier enregistrement
<code>fin</code>	- montre le dernier enregistrement du fichier

Essayez d'utiliser ces commandes pour circuler dans le fichier et visualiser l'enregistrement que vous voulez. Notez que les quatre commandes **début**, **fin**, **suiv** et **préc** n'affichent pas elles-mêmes l'enregistrement. Elles circulent simplement de l'un à l'autre sans regarder si vous avez ou non demandé un affichage.

## RECHERCHE DANS UN FICHER

### Trouve

La première commande de recherche, la plus simple, est **trouve**. A partir du début d'un fichier, elle recherche la première occurrence<sup>2</sup> d'une chaîne de caractères dans n'importe quel champ. Par exemple :

`trouve "sénégal"`

Quand vous presserez **(ENTREE)**, il y aura un court instant d'attente, puis le premier enregistrement contenant le mot "sénégal" s'affichera. Notez que cette recherche est indépendante du fait que le texte soit en majuscules ou en minuscules, et trouvera donc "sénégal", "SENEGAL" ou "Sénégal".

---

<sup>2</sup> De l'anglais "occurent" : apparition d'une séquence de lettres dans un énoncé.

## Continue

Si le premier enregistrement trouvé n'est pas celui que vous désirez, vous trouverez l'occurrence suivante en tapant :

```
continue
```

La commande **continue** recherchera une nouvelle occurrence de la chaîne à partir de l'enregistrement suivant.

Il se peut que vous ayez à recommencer plusieurs fois une recherche avant de trouver l'enregistrement voulu. Pressez **(F5)** et **ARCHIVE** rappellera la commande précédente dans la ligne de commande. Pressez **(ENTREE)** et la commande sera exécutée.

## Cherche

Une autre méthode pour localiser un enregistrement particulier est l'utilisation de la commande **cherche**. Elle permet de localiser un enregistrement en fonction du contenu d'un ou de plusieurs de ses champs. Par exemple :

```
cherche continent$="EUROPE" et langage$="FRANCAIS"
```

Cette séquence trouvera le premier enregistrement dans le fichier qui réunit les deux conditions. Vous devez entrer au clavier le nom complet des champs.

A la différence de **trouve**, la commande **cherche** ne teste que les champs spécifiés, et tient compte des majuscules et des minuscules. Utilisez les fonctions **majus( )** et **minus( )** pour que la recherche soit indépendante de la calligraphie, par exemple :

```
cherche minus(continent$)="europe"
```

A nouveau, la commande **continue** peut être utilisée pour trouver l'occurrence suivante.

## Isole

Vous pouvez avoir envie quelquefois de regarder une partie d'un enregistrement d'un fichier. Supposez, par exemple, que seuls certains détails concernant les pays d'Europe vous intéressent. **Isole** extrait du fichier tous les enregistrements qui satisfont à une certaine condition. Le fichier se comportera alors comme si seuls, ces enregistrements sont présents. Essayez cette commande sur le fichier PAYS pour voir comment elle marche.

Pour commencer, tapez :

```
écris nombenr( )
```

qui indique combien il y a d'enregistrements dans le fichier.

Puis, tapez :

```
isole continent$="EUROPE"  
écris nombenr( )
```

et vous verrez combien d'enregistrements ont été sélectionnés. Les enregistrements extraits du

fichier sont contenus en mémoire centrale, et vous pouvez les réintégrer au fichier, n'importe quand, en utilisant l'ordre **restaure**.

Tapez :

```
Restaure
```

et demandez de nouveau la valeur de **nombenr( )**, pour vérifier que le fichier a été remis dans son état original.

Quand vous utilisez la commande **écris**, toute fiche visible à l'écran s'efface. Ceci, parce qu'en général **affiche** et **écris** utilisent des zones de l'écran qui se chevauchent. Après l'usage de **écris**, vous devez retaper **affiche** pour restituer l'image à l'écran.

### TRI D'UN FICHER

Les enregistrements du fichier peuvent ne pas être dans l'ordre de vos besoins. Vous pouvez trier le fichier de manière numérique ou alphanumérique, selon le contenu des champs.

**Seuls les huit premiers caractères des champs alphanumériques sont pris en compte lors des tris.**

Supposez, par exemple, que vous vouliez trier les enregistrements de "PAYS" dans l'ordre alphabétique des capitales. Vous devrez utiliser la commande **trie** comme suit :

```
trie capital$c
```

Le '**c**' qui suit le point-virgule précise qu'il s'agit d'un tri croissant. Remplacez '**c**' par '**d**', si vous désirez un tri décroissant. Le champ **capital\$** devient la clef de tri pour le fichier. Il est possible de spécifier une clef de tri comportant jusqu'à **quatre** champs, en donnant la liste des champs après la commande **trie**.

Pour chacune des clefs, vous devez préciser si le tri sera croissant ou décroissant. La commande suivante, par exemple, triera le fichier de manière décroissante pour la population, croissante pour la capitale :

```
trie pop;d,capital$c
```

Remarquez qu'un point-virgule sépare chaque nom de champ du 'c' ou 'd' qui spécifient ordre croissant ou décroissant, mais que chaque paire (nom de champ/lettre) est séparée par une virgule de la suivante.

Quand plus d'un champ est spécifié pour diverses options de tri, les enregistrements sont initialement triés selon le contenu du premier champ de la liste. Si deux enregistrements ou plus ont le même contenu pour ce champ, ils sont ordonnés selon le champ suivant dans la liste. S'il y a encore des enregistrements équivalents, ils sont classés selon le contenu du troisième champ, et ainsi de suite.

## SITUER

Quand un fichier a été trié, vous pouvez utiliser la commande **situe** pour faire d'un enregistrement quelconque l'enregistrement courant dans un fichier. Son rôle est de trouver le premier enregistrement dont le premier champ est plus grand ou égal à l'expression donnée.

Par exemple, si le fichier "PAYS" a été trié comme décrit ci-dessus, la commande :

```
situe 100
```

localise le premier pays dans le fichier trié dont la population est de 100 millions d'habitants. S'il n'y en a pas, **ARCHIVE** situera le premier pays dont la population est inférieure à 100 millions (rappelez-vous que le fichier a été trié dans l'ordre décroissant).

**Situe** est suivi par une expression qui peut aussi bien être numérique qu'alphanumérique, mais doit être du même type que le champ utilisé pour trier le fichier (voir le chapitre Références).

Vous pouvez localiser un enregistrement avec une clef de plus d'un champ, en utilisant **situe** avec des expressions multiples, séparées par des virgules.

Exemple :

```
que a=100  
que b$="D"  
situe a,b$
```

trouvera le premier pays dont la population égale 100 millions ou moins, et dont la capitale commence par un "D" ou une des lettres suivantes de l'alphabet. Ici **ARCHIVE** localisera le Bangladesh dont la population égale 76,1 millions et dont la capitale est Dacca.

La seule restriction sur le nombre d'expressions que vous pouvez utiliser avec **situe** est le nombre de champs de la clef de tri.

Vous ne pouvez utiliser **continue** après **situe**. Répéter **situe** avec les mêmes conditions sélectionnera toujours le même enregistrement.

**Situe** est le moyen le plus rapide de localiser un enregistrement dans un grand fichier trié. Vérifiez tout de même qu'il s'agit bien de l'enregistrement que vous cherchez.

## FERMER UN FICHER

Quand vous avez fini de lire un fichier, vous devez le signaler à **ARCHIVE** en tapant :

```
ferme
```

cela n'agira que sur les fichiers et préservera tout programme ou dessin d'écran. Pour fermer tous vos fichiers, effacer vos données et l'écran, tapez :

```
raz
```

cela restituera à **ARCHIVE** son état initial d'après chargement.

Ou bien, si vous avez fini d'utiliser **ARCHIVE** vous pouvez retourner au **SUPER BASIC** en tapant **quitte**. Cette commande ferme automatiquement tout fichier ouvert, avant de quitter **ARCHIVE**.

**Rappelez-vous que vous ne devez jamais retirer une cartouche d'un Microdrive, tant qu'elle contient des fichiers ouverts.**

# CHAPITRE CINQ

## MODIFICATIONS DE FICHIER

---

Avant d'entrer des exemples, tapez d'abord `raz` pour vous assurer qu'ARCHIVE est initialisé et prêt pour un nouveau départ.

La commande **ouvre** prépare un fichier aux opérations de lecture et d'écriture. Ouvrir un fichier avec la commande **ouvre** à la place de **lis**, permet d'écrire dedans pour changer son contenu, aussi bien que de le lire. Ce qui signifie que toute addition, suppression ou modification s'intègre au fichier, de manière permanente, à la fermeture.

Tapez :

```
ouvre "PAYS"
```

Si vous avez ouvert un fichier à la lecture avec **lis**, vous ne devez pas utiliser de commande qui tente de modifier les données. Sinon, **ARCHIVE** afficherait un message d'erreur. Les commandes décrites dans ce chapitre modifient les fichiers et ne doivent être utilisées qu'avec un fichier ouvert par **ouvre**

Affichez le premier enregistrement ainsi :

```
début affiche
```

## FERMETURE DU FICHIER

Quand vous aurez fini les modifications du fichier, vous devrez le fermer (en utilisant **ferme** ou `raz`) pour vous assurer que tous les changements sont enregistrés.

Si vous ne fermez pas correctement un fichier (par exemple, si vous vous contentez d'éteindre l'ordinateur quand vous avez fini), vous risquez de modifier le fichier involontairement, et vos dernières modifications ne seront pas enregistrées.

**Assurez-vous toujours qu'il n'y a pas de fichier ouvert sur la cartouche avant de la retirer du Microdrive. N'éteignez pas l'ordinateur avant d'avoir fermé tous les fichiers ouverts et enlevé les cartouches des Microdrives.**

## INSERTIONS

La commande **insère** s'utilise pour rajouter un ou plusieurs enregistrements au fichier courant. Quand vous vous servirez de **insère**, vous serez amené à entrer le contenu de chaque champ du nouvel enregistrement.

Tapez :

```
insère
```

Dans la zone d'affichage, vous verrez alors

```
nom logique      : maître
country$         :
continent$       :
capital$         :
currency$        :
language$        :
pop              :
area             :
gdp              :
```

Vous pouvez à présent taper le contenu de chaque champ. Pour passer d'un champ au suivant, pressez **(ENTREE)** ou **(TAB)** ; et d'un champ au précédent, maintenez la touche **(SHIFT)** tout en appuyant sur **(TAB)**. Vous pouvez faire autant de modifications que vous voulez dans les champs jusqu'à ce que vous soyez satisfait. Le nouvel enregistrement s'insère dans le fichier lors de la pression de **(F5)**. Appuyez sur **(F4)** pour quitter **insère**.

Essayez :

```
ECOSSE           (TAB)
EUROPE           (TAB)
EDIMBURGH        (TAB)
LIVRE STERLING  (TAB)
ANGLAIS          (TAB)
10               (TAB)
30               (TAB)
50.              (TAB)
```

Vous verrez sur la zone d'affichage :

```
nom logique      : maître
country$         : ECOSSE
continent$       : EUROPE
capital$         : EDIMBURGH
currency$        : LIVRE STERLING
language$        : ANGLAIS
pop              : 10
area             : 30
gdp              : 50
```

Quand vous avez tapé correctement les nouvelles informations **(F5)** insère le nouvel enregistrement dans le fichier. Pressez **(F4)** quand vous terminez l'insertion.

Vous pouvez aussi conclure l'entrée de chaque champ et passer au suivant en pressant **(ENTREE)**. Le nouvel enregistrement est automatiquement ajouté au fichier quand vous appuyez sur **(ENTREE)** après la dernière valeur. Si le fichier a été trié, le nouvel enregistrement est inséré à sa place correcte.

## SUPPRESSION

La commande **efface** sert à supprimer du fichier l'enregistrement courant (montré par **affiche**). Tout ce que vous avez à faire, pour ôter un enregistrement particulier, est de l'afficher et, après vous être assuré que c'est le bon, tapez :

```
Efface
```

## MODIFICATION D'UN ENREGISTREMENT

Il est aussi simple de modifier le contenu d'un champ que de tous, dans un enregistrement existant. Deux méthodes sont possibles :

### Modifie

Sélectionnez l'enregistrement que vous voulez modifier (utilisez **affiche** et **trouve**) puis tapez **modifie**. **Modifie** travaille comme **insère** sauf que l'on voit l'ancien contenu de chaque champ. Vous pouvez sauter les champs que vous ne voulez pas transformer (en utilisant **(TAB)** ou **(ENTREE)**). Entrez une nouvelle valeur ou servez-vous des touches de déplacement du curseur pour en modifier une ancienne. Pressez **(F5)** pour remettre l'enregistrement en place.

Comme avec **insère** l'enregistrement est automatiquement remis en place en pressant **(ENTREE)** après le dernier champ de l'enregistrement.

### Change

Sélectionnez l'enregistrement que vous voulez traiter, puis changez le contenu des champs jusqu'à ce que l'enregistrement affiché soit comme désiré. Tapez **change**. Par exemple, supposons que vous décidiez que l'Islande devrait être en Europe plutôt que dans l'Arctique. Trouvez l'enregistrement en tapant :

```
trouve "ICELAND"  
affiche
```

Utilisez **que** pour changer le contenu du champ de **continent\$**.

```
que continent$="EUROPE"
```

Pour finir, mettez cette modification dans l'enregistrement en tapant **change**.

Dans les deux méthodes ci-dessus, le nouvel enregistrement sera inséré à sa position correcte dans le fichier préalablement trié.

La commande **modifie** est plus simple à utiliser, mais elle affecte toujours l'enregistrement courant. La commande **change** s'emploie quand vous vous servez de beaucoup de fichiers.

Souvenez-vous que vous devez fermer le fichier avec les commandes **ferme**, **raz** ou **quitte**, avant d'éteindre l'ordinateur.

# CHAPITRE SIX

## CREATION DE FICHIER

---

Si vous avez suivi les exemples jusqu'ici, vous n'avez utilisé **ARCHIVE** que pour regarder les fichiers fournis. Dans ce chapitre, vous verrez comment créer votre propre fichier.

Si nécessaire, tapez **raz** pour effacer la mémoire de l'ordinateur et fermer tous les fichiers encore ouverts. Assurez-vous que le Microdrive 2 contient bien une cartouche formatée sur laquelle sera créé le fichier.

Supposons que vous vouliez utiliser **ARCHIVE** pour faire un catalogue de vos livres. Vous devez donc créer un nouveau fichier nommé "LIVRES". La première chose à faire est de décider ce que va contenir le fichier, c'est à dire quels champs vous allez utiliser dans chaque enregistrement. Dans notre cas, vous aurez besoin évidemment de préciser le nom de l'auteur, le titre et le sujet du livre ; vous pouvez aussi inclure d'autres détails tels que le genre (Roman ou autre), le numéro ISBN (International Standard Book Number), sa place dans votre bibliothèque, un résumé du texte, etc...

Dans cet exemple, nous ne prendrons que trois champs alphanumériques, un pour l'auteur, un pour le titre, un pour le sujet, et un champ numérique pour le numéro ISBN.

### CREATION

Vous créez un fichier avec la commande **crée**. Vous devez préciser le nom du fichier à créer et le nom des champs à utiliser dans chaque enregistrement. Le signe '\$' indique que le champ contient une donnée alphanumérique. Quand vous avez fini de définir les champs d'un enregistrement, vous en terminez avec la commande **crée** par **fcrée**. Vous pouvez créer un fichier catalogue simple en tapant la séquence suivante :

```
crée "LIVRES"  
auteur$  
titre$  
sujet$  
isbn  
fcrée
```

Il est à noter que la commande **fcrée** n'est pas nécessaire. Vous pouvez l'utiliser si vous le voulez. Mais il suffit, pour clore la création d'un fichier simple, de presser (**ENTREE**) sur une ligne d'entrée vierge. Vous devez cependant inclure **fcrée** si vous vous servez de **crée** dans un programme **ARCHIVE**.



## AJOUT D'ENREGISTREMENT

Quand vous avez créé un fichier, il est ouvert en même temps à la lecture et à l'écriture, mais il ne contient pas d'enregistrement. Ceux-ci peuvent être rajoutés en utilisant **insère**. Tapez

Insère

et la zone d'affichage montrera :

```
nom logique      : maître
auteur$          :
titre$           :
sujet$           :
isbn             :
```

Tout ce que vous avez à faire est d'entrer au clavier le contenu de chaque champ. Tapez, par exemple :

```
COHEN J.M.                (TAB)
Traité culinaire sur la soupe aux choux (TAB)
Recette de Grand Mère     (TAB)
123456789                 (TAB)
```

Vous verrez apparaître sur la zone d'affichage :

```
nom logique      : maître
auteur$          : COHEN JM
titre$           : Traité culinaire sur la soupe aux
choux
sujet$           : Recettes de Grand Mère
isbn             : 123456789
```

Insérez l'enregistrement dans le fichier "**LIVRES**" en pressant **(F5)**. A ce moment-là, une fiche vierge se présente à l'écran pour un nouvel enregistrement.

Souvenez-vous que vous pouvez aussi clore l'entrée pour chaque champ et aller au suivant en pressant **(ENTRÉE)**. Après le dernier champ, **(ENTREE)** ajoute l'enregistrement au fichier.

Quand vous avez fini, pressez **(F1)**, et n'oubliez pas d'utiliser **ferme ou quitte** pour sauvegarder le fichier.

# CHAPITRE SEPT

## ASPECT DES ECRANS

---

Quand vous utilisez la commande **affiche** sur le fichier que vous avez créé, les enregistrements sont montrés selon le format d'écran standard d'ARCHIVE

### DEFINITION D'UN DESSIN D'ECRAN

Vous pouvez définir votre propre dessin d'écran, mieux adapté à l'information de vos fichiers. Ouvrez un fichier déjà existant par **ouvre** et tapez :

```
affiche
```

La commande **éditem** vous place en mode d'édition écran, tapez :

```
Editem
```

Le dessin d'écran standard, créé automatiquement par **ARCHIVE** apparaît dans la zone d'affichage. S'il n'y a pas de dessin d'écran dans la mémoire de l'ordinateur, la zone d'affichage restera vierge.

Quel que soit le fichier, vous verrez que le contenu des champs n'est pas inclus. Des rangées de points marquent les emplacements où sont normalement montrés ces contenus. On peut comparer un dessin d'écran à un arrière-plan sur lequel les contenus de plusieurs variables sont montrés à des emplacements spécifiques. **ARCHIVE** présente un dessin d'écran en deux étapes - d'abord, il dessine l'arrière-plan, puis il affiche les contenus des variables aux positions définies.

Au commencement, vous êtes au niveau principal de **éditem** et vous avez trois options :

```
Définir l'arrière-plan sur l'écran  
Presser (ESC) pour quitter éditem  
Presser (F3) pour utiliser une commande d'édition d'écran
```

Pour former un dessin d'écran, pressez **(F3)** puis **'C'** pour effacer l'écran et prendre un nouveau départ. Pressez **(ENTER)** pour confirmer votre choix ; toute autre touche vous ramènera au niveau principal de **éditem**.

Choisissez les couleurs du papier et de l'encre en pressant **'P'** ou **'E'** et toute autre touche pour faire défiler les quatre couleurs disponibles. **(ESC)** ramène au niveau principal, pour la définition de l'arrière-plan.

L'arrière-plan peut être un texte explicatif comme :

```
Répertoire Géographique Mondial de Jean Dupont.
```

Ou il peut consister en un nouveau nom de champ dans votre fichier :

### **Population (en millions d'hbts) :**

Le curseur est mobile dans la zone d'affichage, grâce aux quatre touches de déplacement. Tout ce que vous taperez apparaîtra immédiatement dans la zone d'affichage à la position du curseur, et deviendra partie intégrante de l'arrière-plan du dessin. La seule exception est quand le curseur est positionné dans une zone de l'écran réservée à l'affichage d'une variable. **ARCHIVE** montre le nom de la variable dans la zone de travail, au bas de l'écran. Vous ne pouvez positionner de texte d'arrière-plan dans cette zone, à moins de l'avoir préalablement libérée, comme nous le verrons plus loin.

Les quatre commandes d'édition vous permettent de produire des dessins d'écran attrayants et colorés pour afficher vos données. L'effacement d'écran a déjà été expliqué. Vous aurez sans doute besoin de faire des expériences pour dominer complètement les trois commandes restantes. Assurez-vous que vous utilisez une copie du fichier qui peut être sacrifiée.

## **COMMANDES D'EDITION D'ECRAN**

### **Marquage de Variable (V)**

Supposons que vous vouliez montrer la valeur de la variable **country\$** à une position particulière de l'écran. Amenez le curseur à ce point et pressez **(F3)** puis la touche **'V'**. **ARCHIVE** demande d'entrer le nom de la variable, et vous tapez :

country\$

Notez que ce nom n'apparaît pas à l'écran - vous marquez seulement l'endroit où la valeur devra être affichée. Quand vous pressez **(ENTREE)**, **ARCHIVE** attend que vous lui indiquiez la longueur du champ où la variable sera affichée. Pressez n'importe quelle autre touche pour remplir d'une rangée de points la zone désirée. Pour en effacer, employez **(CTRL)** et la touche de déplacement à gauche du **curseur**. **(ENTREE)** vous ramène au niveau principal de **éditem**.

Si vous bougez le curseur dans l'une des zones marquées (par des points), **ARCHIVE** montre dans la zone de travail, le nom de la variable pour laquelle l'espace est réservé.

Si un nouveau marquage de variable en chevauche un autre, la possibilité d'annuler le plus ancien vous est offerte. Vous pouvez alors redéfinir un nouveau champ pour une nouvelle variable.

### **Encre (E)**

Pour changer la couleur de l'encre, positionnez le curseur à l'endroit désiré et pressez **(F3)** puis **'E'**. **ARCHIVE** vous indique les quatre couleurs disponibles dans la zone de contrôle. Celle que vous sélectionnez est mise en évidence. Pressez n'importe quelle touche pour changer de couleur, puis **(ENTREE)** pour valider votre choix. Tout ce que vous allez taper ensuite apparaîtra dans cette nouvelle couleur jusqu'à ce que vous utilisiez à nouveau la commande **encre**.

## Papier (P)

Changer la couleur du papier s'effectue de la même manière, sauf qu'il faut presser **(F3)** puis la touche **'P'**.

Pour qu'un changement de couleur n'affecte qu'une partie de ligne, amenez le curseur au début de la zone concernée, puis sélectionnez les couleurs du papier et de l'encre. Placez le curseur à la fin de la zone, et faites une deuxième sélection des couleurs, en leur restituant leur valeur d'origine.

## ACTIVATION D'UN FORMAT D'ECRAN

Quand un format d'écran a été défini, il reste activé même après avoir quitté **éditem**. Ce qui signifie que les valeurs de toutes les variables, indiquées dans le dessin d'écran, s'affichent automatiquement chaque fois **qu'ARCHIVE** achève l'exécution d'une commande ou d'un programme. Si, par exemple, vous tapez **suiv**, **ARCHIVE** pointe sur l'enregistrement suivant du fichier courant, et ne présente que les champs contenus dans l'écran actif. Tout écran actif est désactivé par la commande **éponge**.

Si un dessin d'écran est désactivé, la commande **écran** le réactive. Elle affiche le dessin d'écran, mais pas les valeurs courantes des variables.

## SAUVEGARDE ET CHARGEMENT D'ECRANS

Vous pouvez sauvegarder vos dessins d'écran sur une cartouche de Microdrive avec la commande **sauvem** :

```
sauvem "Nom du Fichier"
```

Où "Nom de fichier" est celui de votre choix. Le dessin d'écran est sauvegardé tel qu'il apparaît. Vous pouvez recharger le dessin d'écran en entrant la commande :

```
chargem "Nom du Fichier"
```

Quand un dessin d'écran se charge, il s'affiche automatiquement et s'active.

**ARCHIVE** ne met pas automatiquement à jour un dessin d'écran actif à partir d'un programme. Supposez que vous vouliez montrer tous les enregistrements du fichier courant, un par un, et que vous tentiez de le faire au moyen du programme ci-dessous :

```
début:que x=0: tant que x<nombrenr ( ):suiv:que x=x+1:ftantque
```

Les commandes **tantque** et **ftantque** font s'exécuter répétitivement la section du programme qu'elles entourent, tant que la condition qui suit **tantque** est vraie. Pour que ce soit correct, à chaque **tantque** doit correspondre un **ftantque**.

Ce programme ne fait pas ce que vous voulez puisque **ARCHIVE** ne met à jour les contenus du dessin d'écran qu'à la fin.

## LA COMMANDE MONTRE

Vous pouvez cependant forcer l'affichage des variables dans un écran actif à partir d'un programme, avec la commande **montre**. Le petit programme suivant affichera tous les enregistrements comme voulu.

```
début:que x=0:tantque x<nombre( ) :montre:suiv:que x=x+1:ftantque
```

S'il n'y a pas d'écran actif, **montre n'a** aucun effet.

## LA COMMANDE AFFICHE

Rappelez-vous que la commande **affiche** utilise la disposition standard. Elle remplacera tout dessin d'écran par sa propre liste, simplifiée, des champs de l'enregistrement courant. En conséquence, vous devez sauvegarder par **sauvem** votre écran, avant de réutiliser **affiche**. Sinon, votre écran sera remplacé et vous ne pourrez pas le récupérer. Vous devrez le redessiner avec **éditem**.

# CHAPITRE HUIT

## LES PROCEDURES

---

Avant de commencer les exemples de ce chapitre, tapez **raz** pour réinitialiser l'ordinateur, puis **lis "PAYS"** pour ouvrir le fichier de démonstration sur la cartouche qui doit être dans le Microdrive 2.

Les commandes et les fonctions d'**ARCHIVE** forment un véritable langage de programmation, dont vous pouvez vous servir pour gérer vos fichiers. Vous allez voir comment les programmes d'**ARCHIVE** sont faciles à écrire.

Un programme **ARCHIVE** est fait d'une ou de plusieurs sections séparées. Chacune d'elles est une procédure, qui est tout simplement un tronçon de programme doté d'un nom. Écrire une procédure, c'est réellement ajouter une nouvelle commande à **ARCHIVE**. Elle s'exécute en tapant son nom au clavier.

Aucune procédure ne peut contenir plus de 255 lignes de 160 caractères.

### CREATION D'UNE PROCEDURE

Utilisez l'éditeur de programme chaque fois que vous désirez écrire ou modifier une procédure. L'éditeur de programme est décrit en détail au chapitre 9, mais ici nous allons voir brièvement quelques-unes de ses possibilités, juste de quoi écrire quelques procédures. Auparavant, vous aurez vérifié qu'il n'y a pas de procédure en mémoire centrale. Tapez :

édite

pour entrer dans l'éditeur de programme. La zone de contrôle indique alors que vous devez donner le nom de la procédure. L'éditeur permet toujours de créer une nouvelle procédure s'il n'y en a pas qui soit définie ou chargée.

La première chose à faire est de décider ce que doit effectuer la nouvelle procédure. Commençons par une tâche toute simple : faciliter les choses en attribuant à **affiche** le nom abrégé **'a'**. Donnez comme nom de procédure

a

La partie gauche de la zone d'affichage montre à présent le nom, et la partie droite une liste des instructions de la procédure. Comme pour l'instant celle-ci ne contient pas de commande, **proc** et **fproc** qui marquent le début et la fin de toute procédure, sont automatiquement écrits par **ARCHIVE**.

Le corps de la procédure, qui est une suite d'actions à effectuer, doit être rédigé. La zone de contrôle indique que vous pouvez ajouter des lignes de texte à la nouvelle procédure. Dans notre exemple, ce texte est la commande **affiche**. Tapez :

```
affiche
```

et **ARCHIVE** insèrera le nouveau texte dans la procédure, sous la ligne mise en évidence. A présent, la procédure est complète, et vous pouvez quitter l'édition en pressant **(ESC)** deux fois. Cette nouvelle procédure aura la même fonction que **affiche**.

## LISTE ET IMPRESSION DE PROCEDURES

La commande **édite** fait apparaître la liste des noms de toutes les procédures définies, présentes en mémoire centrale.

Pour visualiser le contenu de l'une d'elles, faites défiler la liste vers le bas avec **(TAB)**, ou vers le haut avec **(SHIFT)** et **(TAB)** en même temps, jusqu'à ce que le nom de la procédure recherchée soit mis en évidence. L'ensemble de ces instructions apparaîtra automatiquement à droite de l'écran. Si c'est trop long pour être contenu dans la zone d'affichage, seul le début est montré. Et vous pouvez alors vous déplacer vers le haut ou vers le bas, à travers la procédure, par l'usage des touches de déplacement du curseur. A la fin, vous quittez l'édition en pressant **(ESC)**.

Si vous voulez une liste imprimée de vos procédures, servez-vous de la commande **liste**.

Tapez :

```
liste
```

ce qui envoie vers l'imprimante toutes les procédures présentes en mémoire centrale.

**Attention n'utilisez cette commande que si l'imprimante est connectée, sinon le programme restera 'en suspens' indéfiniment.**

## SAUVEGARDE ET CHARGEMENT DE PROCEDURES

Pour conserver les procédures que vous avez définies, utilisez la commande **sauve** qui les sauvegarde sur la cartouche du Microdrive sous forme d'un fichier doté d'un nom. Vous pouvez donner le nom de votre choix à ce fichier, par exemple "Mesproc" :

```
sauve "Mesproc"
```

Pour les recharger en mémoire centrale, tapez :

```
charge "Mesproc"
```

Lors du chargement, la commande **charge** efface toutes les anciennes procédures. Pour les conserver, utilisez la commande **unis**, par exemple :

```
unis "Mesproc"
```

Elle fonctionne comme **charge** mais n'efface pas les anciennes procédures. Si une nouvelle procédure porte le même nom qu'une ancienne, elle prendra sa place.

## EXAMEN D'ENREGISTREMENTS DE FICHIER

Redonner des noms d'une seule lettre aux procédures les plus utilisées est l'un des moyens de se simplifier la vie. Un autre consiste à écrire une procédure plus longue qui remplace plusieurs commandes par une seule pression de touche. Essayez avec la commande **édite** de définir une procédure qui vous permet d'ouvrir et d'examiner n'importe lequel de vos fichiers, dans la mesure où celui-ci n'est pas encore en mémoire.

Si vous avez déjà défini une procédure, taper :

```
édite
```

ne vous donne pas automatiquement la possibilité de créer une nouvelle procédure. Pour ce faire, vous devez presser **(F3)** et **'N'**.

Ne vous affolez pas si vous faites quelques erreurs, vous apprendrez à les corriger dans le prochain chapitre.

```
proc vufiche
  éponge
  saisis "Quel fichier ?";fichier$
  lis fichier$
  affiche
  que touche$="z"
  tant que touche$ < >"q"
    montre
    que touche$=minus(clavier( ))
    si touche$="d":début:fsi
    si touche$="f":fin:fsi
    si touche$="s":suiv:fsi
    si touche$="p":préc:fsi
  f tantque
  ferme
fproc
```

Souvenez-vous que vous quittez le mode édition en pressant deux fois **(ESC)**.

Vous pouvez utiliser la procédure en tapant :

```
vufiche
```

La zone d'affichage sera tout d'abord effacée, puis un nom de fichier vous sera demandé. Entrez par exemple "PAYS". Si cependant "PAYS" est déjà chargé, un message d'erreur apparaîtra. Pour recommencer, tapez **raz** puis chargez et exécutez la procédure à nouveau. Quand vous avez entré le nom de l'un de vos fichiers de données, la procédure ouvrira ce fichier en mode lecture seule et affichera son premier enregistrement. Elle attendra alors une pression sur la touche **'d'**, **'f'**, **'s'**, **'p'** ou **'q'**. Les quatre premières lettres entraîneront l'affichage approprié (**début**, **fin**, **suiv** et **préc**) et la pression de **'q'** (**quitte**) fermera le fichier et arrêtera la procédure.

Comme c'est le premier programme d'une certaine longueur que nous proposons, quelques commentaires sont utiles. Notez d'abord que ce programme est indenté pour avoir un aspect plus clair. Vous n'avez pas besoin de le taper ainsi, l'indentation est rajoutée automatiquement quand vous écrivez, listez ou imprimez la procédure.

La partie principale de la procédure (attente d'une pression de touche en vue d'une action appropriée) est entourée par les commandes **tantque** et **ftantque**. La boucle répétitive ne cessera que lorsque la condition qui suit **tantque** est fausse, dans ce cas, une pression de la touche **'q'**.



La commande **si**, utilisée plusieurs fois dans cette boucle, implique l'emploi de **fsi**, pour marquer la fin des séquences d'instructions à exécuter si la condition est vraie. **si** et **fsi** sont des commandes séparées et peuvent être mises sur des lignes différentes. Par exemple, on aurait pu écrire :

```
si touche$="d"  
  début  
fsi
```

Vous pouvez inclure plusieurs lignes d'ordres entre le **si** et le **fsi**. Ils seront tous exécutés si la condition qui suit **si** est vraie. Dans la procédure **vufiche** ces ordres sont suffisamment courts, pour que chacun d'eux puisse être écrit sur une seule ligne, séparé du suivant par deux points.

Comme vous pouvez le voir, une commande **montre** est utilisée à partir de la boucle principale de cette procédure, pour s'assurer que chaque nouvel enregistrement sera affiché à l'écran. Souvenez-vous que, bien que **affiche (début, fin, etc...)** pointe sur l'enregistrement correct, les données dans la zone d'affichage ne sont pas automatiquement changées avant la fin de la procédure. Si nous n'avions pas inclus la commande **montre**, aucune information n'aurait été montrée sur l'écran, tant que vous n'auriez pas pressé '**q**' pour quitter la procédure. Dans ce cas, vous n'auriez pu voir que le résultat de la dernière séquence de pressions de touche.

# CHAPITRE NEUF

## EDITION

---

Ce chapitre décrit l'éditeur de programmes. Nous y incluons quelques exercices simples, mais le meilleur moyen d'apprendre reste la pratique. Commencez en tapant **raz** pour effacer la mémoire centrale.

Quand vous aurez lu ce chapitre, vous pourrez essayer d'écrire quelques programmes simples, ou bien de modifier les procédures que vous avez déjà créées. Si vous voulez utiliser des exemples plus longs, servez-vous de l'éditeur pour entrer les programmes des chapitres suivants.

### L'EDITEUR DE PROGRAMMES

**Edite** vous place au niveau principal de l'éditeur de programmes. Comme exemple, nous allons créer une procédure et lui ajouter quelques ordres. A partir du niveau principal, pressez **(F3)** et **'C'** ; répondez **test** à la demande du nom de procédure.

Pressez **(ESC)** deux fois, pour quitter l'éditeur sans ajouter d'ordre. Puis réutilisez la commande **édite**. S'il n'y a pas d'autre procédure chargée, vous verrez à l'écran :

```
test      proc test
          fproc
```

Si les procédures que vous avez créées dans le dernier chapitre, sont encore en mémoire, **test** est mis en évidence à gauche parmi elles. Pressez **(F4)** pour insérer des lignes de texte. La ligne contenant **proc** sera mise en évidence. Tapez :

```
écri "Ceci est un test"      (ENTREE)
écri "Il y a deux ordres"    (ENTREE) (ENTREE)
```

Le fait de presser **(ENTREE)** deux fois vous fait quitter l'insertion. Quand vous aurez fini, l'écran affichera :

```
test      proc test
          écris "Ceci est un test"
          écris "Il y a deux ordres"
          fproc
```

La ligne contenant le second ordre d'impression est mise en évidence. Rappelez-vous que jusqu'à ce que vous ayez pressé **(ENTREE)**, vous pouvez utiliser l'éditeur de lignes pour corriger une partie du texte que vous avez saisi. Toutefois, **(ENTREE)** insère la ligne centrale dans la procédure. Pour l'éditer à nouveau, pressez **(F5)**. **(ENTREE)** remplacera l'ancienne ligne par la nouvelle.

Vous n'êtes pas autorisé à éditer les ordres **proc** et **fproc**. Mais vous pouvez éditer le reste des contenus de cette ligne. Il est toujours possible de réattribuer un nom à une procédure grâce à l'éditeur de ligne, pour effacer l'ancien nom et le remplacer par un nouveau. La liste des procédures à gauche de l'écran est automatiquement classée par ordre alphabétique.

### **Commandes d'édition**

Vous avez sans doute remarqué les quatre commandes d'édition quand vous avez créé une nouvelle procédure. Vous pouvez en sélectionner une en pressant **(F3)** puis en tapant la première lettre de son nom.

### **Création d'une procédure (C)**

Entrez le nom de la procédure que vous voulez créer. Si ce nom est déjà attribué, vous ne pourrez en créer une seconde, mais vous aurez la possibilité d'éditer l'ancienne. Après le nom, **(ENTREE)** fait de la nouvelle procédure la procédure courante, listée à droite sur l'écran. Vous êtes en présence d'une procédure vide, c'est à dire qui ne contient que **proc** et **fproc**.

### **Abandon de procédure (A)**

Cette commande efface la procédure courante de votre programme. Vous devez tout d'abord sélectionner la procédure que vous voulez effacer avec **(SHIFT)** et **(TABULATE)**, comme décrit plus haut, pour en faire la procédure courante. Sélectionnez ensuite la commande par **(F3)** puis **'A'**.

Pressez **(ENTREE)** pour confirmer l'effacement. Si vous changez d'avis à ce moment-là, pressez une touche pour revenir à **édite** sans effacer la procédure.

**Faites attention quand vous vous servez de cette commande. Il est impossible de récupérer une procédure effacée. Il faut la retaper.**

### **Extraction de lignes (E)**

Cette commande extrait une ou plusieurs lignes de texte de la procédure courante. Le texte retiré peut être réinséré à une autre position ou dans une autre procédure au moyen de la commande **d'Injection de lignes**.

Utilisez les touches de déplacement vertical du curseur pour que la ligne courante devienne la première ou la dernière de la partie de texte que vous voulez ôter, puis pressez **(F3)** et **'E'**. Les touches de déplacement de curseur vous permettent alors de choisir l'autre extrémité de la zone que vous voulez extraire. Pressez **(ENTREE)**, et les lignes seront retirées de la procédure.

### **Injection de lignes (I)**

Cette commande insère dans la procédure courante, juste en dessous de la ligne courante, le texte retiré par le dernier usage de la commande **'E'**. Avant de vous servir de **'I'** vous pouvez, à l'aide de **(SHIFT)** et **(TAB)** choisir la procédure, et, à l'aide des touches de curseur, l'endroit où vous voulez insérer le texte. Il n'est alors plus possible d'insérer le texte ailleurs sans réutiliser **'E'**.

# CHAPITRE DIX

## PROGRAMMATION DANS ARCHIVE

---

Dans ce chapitre, nous allons développer une application réelle. Chaque nouvelle technique sera expliquée au moment de son emploi.

Supposons que vous vous occupez d'un club ou d'une association qui gère des inscriptions et qui édite un bulletin. Vous devez donc en envoyer un exemplaire, à chaque parution, à tous les membres qui ont réglé leur cotisation. Vous devez aussi envoyer un rappel à tous ceux qui ne l'ont pas encore réglée.

Nous allons établir une liste d'adresses à imprimer sur étiquettes à la demande. Admettons qu'il y ait six parutions de votre bulletin par an, et que les adhérents doivent renouveler leur cotisation à la réception du sixième. Le rappel de paiement est alors indiqué sur l'étiquette. Il est facile d'adapter cet exercice à toute autre situation analogue.

### LISTE D'ADRESSES

Dans cet exemple, nous emploierons autant que possible les commandes existantes et nous en aborderons de nouvelles. Si vous vous sentez perdu parce que vous n'avez pas encore rencontré, ou que vous ne comprenez pas toutes les subtilités d'une commande ou d'une option, consultez le chapitre REFERENCES ou utilisez la fonction **AIDE** en pressant la touche (F1). Nous nous servirons de **insère** et de **change** pour toutes les modifications du fichier. Nous devrons toutefois écrire des procédures spéciales pour l'édition des étiquettes.

Nous allons avoir les besoins suivants :

- Ajouter un nouvel enregistrement au fichier
- Effacer un enregistrement
- Modifier un enregistrement
- Enregistrer le paiement des cotisations
- Editer des étiquettes
- Abandonner le programme

Nous allons écrire les procédures qui effectueront ces tâches et nous les lierons ensemble par l'intermédiaire d'une autre qui nous permettra de choisir l'une de ces options.

Dans cette application, il est facile de déterminer de quels champs un enregistrement sera composé. Le nom et l'adresse sont essentiels et il faudra ajouter un champ pour le nombre de bulletins que l'adhérent aura reçu. Nous pouvons immédiatement créer le fichier nécessaire ainsi :

```
crée "courrier"  
  mr_mme$  
  nom$  
  prénom$  
  rue$  
  ville$  
  département$  
  code_postal$  
  bulletin  
  fcrée
```

Nous avons utilisé trois champs alphanumériques pour l'état civil de la personne : la qualité (Mr,Mme, Dr, etc...), le nom et le prénom. Nous aurions pu n'employer qu'un champ.

Il y a quatre champs alphanumériques pour l'adresse : la rue, la ville, le département et le code postal. Vous pouvez ne pas les utiliser à cet effet et les traiter comme quatre champs quelconques.

Il y a seulement un champ numérique qui contiendra le nombre de bulletins qu'il reste à envoyer.

Maintenant que nous avons le fichier, il ne nous reste plus qu'à l'utiliser pour tester les différentes procédures que nous allons écrire pour lui. C'est une bonne chose que de tester chaque procédure, aussi loin que possible, au fur et à mesure de son élaboration. Vous pouvez alors localiser chaque erreur quand elle apparaît et la corriger immédiatement. Si vous gardez les tests pour la fin, cela risque de se compliquer avec la présence de plusieurs erreurs simultanées. Maintenez les choses aussi simples que possible quand vous testez vos procédures. Assurez-vous que chacune d'elles tourne parfaitement avant de passer à la suivante. De cette manière, votre programme fonctionnera dès que la dernière procédure sera écrite.

### Insertion

Il n'est pas nécessaire d'écrire une procédure pour ajouter des enregistrements. Il suffit **d'insérer**. Souvenez-vous que vous devez utiliser **montre** pour forcer l'affichage des contenus d'un enregistrement à partir d'une procédure. Vous pouvez utiliser **insère** immédiatement pour ajouter quelques enregistrements au fichier afin de faire vos tests sur un fichier réel.

### Effacement

A certains moments, il faudra ôter les fiches des membres qui n'auront pas renouvelé leur inscription. Nous allons écrire une procédure, **nettoie**, qui cherchera dans le fichier tous les enregistrements de ces personnes et qui demandera s'il faut les effacer.

Nous utiliserons pour cela le champ **bulletin** qui contient le nombre de bulletins que la personne doit encore recevoir. Tous les enregistrements où ce champ est à zéro, représentent des candidats à l'effacement.

```
proc nettoie
note ***** efface membres qui ne cotisent plus *****
éponge
affiche
isole bulletin = 0
partout
montre
écris au 10,0; "Efface (o/n) ?
que oui$=majus(clavier( ))
écris oui$
si oui$= "0"
    efface
    écris "Effacé"; tab 15
sinon
    écris tab 15
fsi
fpartout
restaure
fproc
```

Comme un enregistrement effacé ne peut être retrouvé, le contenu total de l'enregistrement est affiché. A la demande, confirmez si vous désirez vraiment l'effacer. Nous utilisons la fonction **clavier( )** qui attend une pression de touche et qui retourne son code ASCII. Notez que **majus()** convertit ce code en majuscule, ce qui autorise une entrée en majuscule et en minuscule.

Une fois que vous êtes sûr que la procédure a été correctement entrée, vous pouvez l'essayer sur votre propre fichier, si bien sûr vous y avez mis un jeu d'essai. Quittez d'abord **édite** en pressant (ESC) (deux fois si besoin est) et sauvegardez votre procédure dans un fichier nommé **"gestion"**. Tapez :

```
sauve "gestion"
```

La procédure **nettoie** est maintenant stockée et peut être appelée chaque fois que **"gestion"** est chargé.

Après l'entrée de chacune des procédures suivantes, répétez l'opération en la sauvegardant dans **"gestion"**.

### Cotisations

Pour mettre à jour les cotisations, vous devez trouver l'enregistrement de la personne concernée. La manière la plus rapide est d'écrire une procédure **trouvenr** qui localise un enregistrement et qui l'envoie à la procédure **payé**.

La procédure **trouvenr** attend une chaîne de caractères (n\$). Si vous pressez simplement **(ENTREE)**, aucune recherche n'est faite, et cela indique que vous avez fini votre mise à jour.

Du niveau **édite** pressez **(F3)** et **'C'** pour créer **trouvenr**.

```
proc trouvenr
note ***** Trouve un enregistrement particulier *****
éponge
que oui$ ="n"
saisis "Qui ?";n$
si n$< >"
trouve n$
tantque oui$<>"0" et vu ( )
    écris mr_mme$;" ";nom$(1);" ";prénom$
    écris rue$
    écris "OK(o/n) ?";
    que oui$=majus(clavier( ))
    éponge
    si oui$<>"0"
        continue
    fsi
ftantque
si non vu( )
    écris n$;"non trouvé"
fsi
fsi
fproc
```

La commande **trouve** recherche le texte dans n'importe quel champ. Mais vous pouvez aussi identifier l'enregistrement par le nom et l'adresse. Bien sûr, le premier enregistrement trouvé ne sera peut-être pas le bon, il faudra alors continuer la recherche. C'est ce que fait la boucle **tantque ftantque**. Elle imprime le nom et la première ligne de l'adresse pour identifier l'enregistrement. Si vous ne répondez pas en pressant '0', elle continue la recherche. La boucle cesse si vous répondez par l'affirmative ou si aucune occurrence n'est trouvée dans les enregistrements restants. Remarquez que la fonction **vu( )** retourne une valeur vraie (différente de zéro) si la recherche est fructueuse.

Comme oui\$ peut encore contenir un '0' (d'une recherche précédente) nous devons lui donner une autre valeur avant d'entrer dans la boucle. Ce qui assure qu'elle sera effectuée au moins une fois.

Nous pouvons maintenant écrire la procédure **payé**.

```
proc payé
  note ***** mise à jour des cotisations *****
  éponge
  que n$="x"
  tant que n$< >"
    trouvenr
    si oui$="0"
      que bulletin=bulletin+6
      change
    fsi
  ftantque
fproc
```

La boucle dans cette procédure continue jusqu'à ce que n\$ soit vide. Ceci vous permet d'enregistrer plusieurs règlements sans avoir à choisir l'option **payé** pour chacun. Quand vous avez fini, pressez seulement (**ENTREE**) à la question "Qui ?". Si la valeur de oui\$ est '0' après l'appel à **trouvenr**, alors **bulletin** est initialisé à 6 pour marquer le règlement.

A nouveau nous devons remplir n\$ avec une valeur quelconque différente de la chaîne vide pour être sûr que la procédure ne sera pas affectée par une précédente opération.

### Mise à jour

La procédure qui vous permet de changer le contenu d'un enregistrement est maintenant très simple. Il s'agit encore de trouver un enregistrement particulier à modifier. La structure générale est semblable à celle de **payé**.

```
proc metajour
  note ***** modifie un enregistrement *****
  que n$="x"
  éponge
  tant que n$<>"
    trouvenr
    si oui$="0"
      modifie
      éponge
    fsi
  ftantque
fproc
```

## PARAMETRES

Nous allons maintenant faire une petite pause dans le développement du programme pour décrire l'emploi de paramètres avec les procédures. Vous pouvez utiliser un paramètre pour passer une valeur à une procédure plutôt que d'utiliser une variable. Nous allons vous montrer dans quelques exemples, la manière de les employer. Vous n'avez pas besoin de les sauvegarder dans "**gestion**" et vous pouvez les effacer avant de poursuivre avec la partie du programme qui traitera les étiquettes.

Essayez les courts programmes suivants. Avec l'éditeur de ligne ajoutez le paramètre à la ligne contenant le nom de la procédure.

```
proc test; a
  écris 5*a
fproc
```

Ceci définit une procédure nommée test qui a besoin d'un paramètre "**a**". Notez que le paramètre est séparé du nom de la procédure par un point-virgule. Chaque fois que vous appellerez cette procédure, vous devrez fournir un paramètre. Par exemple, vous pouvez faire :

```
test; 3
```

ce qui imprimera la valeur 15 - 1- nombre (3) est passé à la procédure qui l'affecte à la variable a. Vous pouvez définir une procédure avec autant de paramètres que vous le désirez si vous les séparez par des virgules. Par exemple :

```
proc essai; a,b,c
  écris a*b*c
fproc
```

que vous pouvez tester par :

```
essai; 3,4,5
```

Les valeurs peuvent être des variables comme montré ci-dessous.

```
que x=2
que y=5
que z=7
essai; x,y,z
```

Notez que les noms des variables n'ont pas à être identiques à ceux utilisés dans la procédure. Vous pouvez faire la différence entre les paramètres formels (ici a, b et c) de la définition de la procédure, et les paramètres véritables qui sont les valeurs réelles.

Vous pouvez aussi passer des expressions :

```
essai; x*2,z/y,(z-y)*x
```

Il n'y a pas de restriction dans l'utilisation de variables numériques, mais vous pouvez aussi passer des chaînes de caractères (ou des expressions alphanumériques) comme paramètres, du moment que vous avez spécifié le type correct de variable dans la définition de la procédure.



Par exemple :

```
proc tente; a$
    écris a$
fproc

que t$="Message"
tente; t$
```

La seule nécessité est que le nombre et le type des paramètres fournis respectent ceux de la définition formelle de la procédure.

### Étiquettes

La raison de ce bref interlude sur les paramètres est qu'ils donnent une assez bonne idée de la façon d'écrire la procédure d'édition des étiquettes. Dans l'objectif de tests, nous allons d'abord écrire la procédure qui montre l'adresse à l'écran, puis nous la convertirons plus tard pour que la sortie se fasse sur l'imprimante. Nous supposons que les étiquettes occupent huit lignes. Si cela ne convient pas à votre imprimante et/ou à vos désirs, vous devez changer le nombre de lignes de blancs dans la procédure pour qu'il corresponde à vos besoins. Rappelez-vous de sauvegarder vos procédures dans "**gestion**".

Nous allons d'abord écrire une procédure qui affiche une simple ligne, dont le contenu sera passé comme paramètre.

```
proc impligne; x$
    écris x$
fproc
```

Nous pouvons utiliser cette procédure pour afficher les huit lignes de texte de l'étiquette.

```
proc impétiq
note ***** Ecris les étiquettes *****
si bulletin
    si bulletin =1
        impligne;"Rappel de cotisation"
    sinon
        impligne;" "
    fsi
impligne; " "
impligne; mr_mme$+" "+non$(1)+" "+prénom$
impligne; rue
impligne; ville$
impligne; département$
impligne; code_postal$
impligne; " "
que bulletin=bulletin-I
change
fsi
fproc
```

La procédure génère le rappel de cotisation sur l'étiquette s'il s'agit du dernier bulletin. A chaque fois qu'une étiquette est imprimée, le nombre de bulletins est décrémenté. Si ce nombre arrive à zéro, alors l'étiquette n'est pas imprimée.

Vous pouvez vous rendre compte combien les paramètres sont pratiques, sans eux, cette procédure aurait été bien plus Longue. Regardez comme est facile d'accoler la qualité, le nom et le prénom sur la première ligne de l'adresse.

Vous vous demandez peut-être pourquoi nous n'avons pas utilisé **écri**s dans la procédure **impétiq**. La raison est que dans sa forme actuelle, il est très facile de vectoriser la sortie vers un autre périphérique que l'écran en changeant dans **impligne**:

```
proc impligne; x$ imprime x$ fproc
```

Pour terminer, écrivons la procédure d'édition des étiquettes

```
proc toutétiq
  éponge
  partout
  impétiq
  fpartout
fproc
```

### Abandon du programme

L'abandon du programme est la dernière option. La procédure correspondante peut être très simple, tout ce qu'elle a à faire est de vérifier que le fichier est bien fermé avant de rendre le contrôle au clavier. Nous y avons ajouté un petit message qui confirmera que le programme s'est bien terminé.

```
proc salut ferme
  écris "Au revoir"
  stoppe
fproc
```

### ERREURS

Il est sûr que, tôt ou tard, vous ferez une erreur en utilisant le programme. Vous risquez, par exemple, de presser la touche **(ESC)** accidentellement ou de taper une chaîne, alors qu'un nombre est attendu. Ce type d'erreur est détecté par **ARCHIVE** qui affiche le message approprié et qui rend la main au clavier.

Vous pouvez utiliser la commande **erreur** pour signaler la procédure à effectuer dès qu'une erreur est rencontrée.

La gestion des erreurs par **ARCHIVE** est inactivée pour la procédure contenant la commande **erreur** et il vous est laissé le soin de les traiter de manière sélective. Vous obtiendrez le numéro de l'erreur en lisant **errnum( )**. Vous pouvez le relire plusieurs fois puisqu'il n'est remis à zéro que par une nouvelle utilisation de **erreur**. Si aucune erreur ne s'est produite depuis le début du programme, ou depuis la dernière exécution de **erreur**, **errnum( )** est égal à zéro.

Cette méthode, même si elle est n'est pas aisée à comprendre au premier abord, vous donne un contrôle souple et puissant de la gestion des erreurs. L'exemple suivant montre la manière type d'utiliser **erreur** dans l'entrée de nombres.

```

proc numtest
  saisis x
fproc

proc test
  que n=1
  tantque n
    erreur numtest
    que n=errnum( )
    si n
      écris "Erreur numérique";n;",recommencez"
    fsi
  ftantque
fproc

```

La première procédure attend simplement l'entrée d'un nombre dans la variable x. La seconde gère toute erreur ayant pu être générée dans la première. Si une erreur est apparue dans **numtest**, un retour à **test** sera fait avec **errnum( )** mis à jour. Si **errnum( )** est différent de zéro, un message d'erreur est affiché (il peut être n'importe lequel). Comme ces ordres sont inclus dans une boucle **tantque ... ftantque**, toute erreur relancera la procédure. Le numéro d'erreur est effacé par **erreur**, pour l'essai suivant. Vous ne pouvez quitter **test** qu'avec l'entrée d'un nombre valide.

Cet exemple affiche le numéro de l'erreur détectée. Dans de nombreux cas, celui-ci est peu important, car il est surtout intéressant de savoir s'il y a eu erreur ou non. Une liste de numéros d'erreur et de leur signification est donnée dans le chapitre des Références.

Nous pouvons maintenant aborder la procédure qui permet de choisir l'une des six options sur une simple pression de touche. Elle est suffisamment simple pour qu'aucune explication ne soit nécessaire.

```

proc choisir
  note ***** Choix d'une option *****
  éponge
  écris
  écris "Ajout Impres. Cotis. Modif. Effac. Quit.,":;
  écris " ? ";
  que choix$=minus(clavier( ))
  écris choix$
  si choix$="a": insère: f si
  si choix$="i": impétiq: fsi
  si choix$="c": payé: fsi
  si choix$="m": modif: fsi
  si choix$="e": efface: fsi
  si choix$="q": salut: fsi
fproc

```

Tout ce qui reste à faire pour terminer notre programme, est d'écrire une procédure de démarrage qui ouvre le fichier et appelle **choisir**. Nous devons inclure **choisir** dans une boucle pour que les options vous soient proposées chaque fois que l'option précédente est terminée.

Vous verrez que la boucle **tantque ... ftantque** dans la procédure suivante ne se termine jamais. Comme une telle boucle ne s'achève que si l'expression qui suit le **tantque** est à zéro, il suffit de maintenir celle-ci à 1 par exemple. La seule manière de quitter sera alors le choix de l'option **Quit**. La commande **stoppe** dans **salut** rend immédiatement la main au clavier.

```

proc commence
note ***** procédure de lancement *****
éponge
ouvre "courrier.dbf"
tantque 1
  erreur choisir
  que n=errnum( )
  si n
    écris "Erreur, pressez une touche pour continuer"
    que m$=clavier( )
  fsi
ftantque
fproc

```

Dans cette boucle, une séquence d'ordres gère les erreurs. S'il y en a une, le programme attend une pression de touche avant de poursuivre pour vous laisser voir votre dernière manipulation et la corriger.

## LA COMMANDE EXEC

La procédure principale de notre programme est appelée **commence**. C'est à dire que vous pouvez utiliser la commande **exec** lorsque vous lancez ce programme. Nous avons déjà utilisé cette commande quand nous avons vu le programme de '**chargement**' du fichier "PAYS".

Sauvegardez la procédure finale dans "**gestion**". Quand vous voulez exécuter votre programme vous aurez besoin de charger les procédures en mémoire centrale, puis de lancer la procédure principale.

Exemple :

```

charge "gestion"
commence

```

La commande **exec** charge le programme nommé et exécute automatiquement la procédure **commence** si elle existe. Vous pouvez donc lancer le programme de la même manière en faisant :

```

exec "gestion"

```

Les deux dernières sections de ce chapitre contiennent des procédures d'ordre général que vous trouverez sans doute pratiques.

## VARIABLES LOCALES

La plupart des variables d'un programme sont des variables globales. Elles peuvent être utilisées ou modifiées dans n'importe quelle procédure, et pas seulement dans celle où elles ont été déclarées.

Les variables utilisées comme des paramètres formels dans la procédure sont des variables locales et elles ne sont pas reconnues en dehors de la procédure où elles ont été déclarées.

Les exemples suivants clarifieront la situation. Avant de continuer, tapez **raz** pour nettoyer la mémoire de l'ordinateur. D'abord, créons une procédure qui englobe deux variables locales a et b\$ ainsi que deux variables globales u et v\$.

```

proc démo; a,b$
  écris a,b$
  que u=3
  que v$="texte"
  écris u;v$
fproc

```

Utilisons démo :

```

démo 5;"mots"

```

Les quatre valeurs sont imprimées car reconnues dans **démo**. Tapez :

```

écris u;v$

```

montre que ces deux variables sont reconnues même hors de la procédure. Cependant :

```

écris a,b$

```

causera une erreur car elles ne sont pas reconnues en dehors de **démo**. Tous les paramètres formels sont des variables locales, mais vous pouvez aussi en déclarer d'autres.

```

proc extra
  écris "dans extra"
  écris p;q;r
fproc

proc vide
  local q,r
  que p=2
  que q=3
  que r=4
  écris "dans vide"
  écris piger
  extra
fproc

```

Si vous exécutez **vide** vous trouverez que les valeurs de p, q et r sont toutes reconnues (et imprimées) dans **vide**, mais **extra** ne connaît pas les valeurs de q et de r qui sont locales à **vide**.

Les valeurs des variables locales ne sont pas définies sauf dans la procédure où elles sont déclarées - pas même dans celles appelées de la procédure déclarative. La variable p est globale, elle est reconnue partout.

Vous devez vous demander pourquoi les variables locales sont nécessaires. Pour illustrer leur utilité, supposez que vous écrivez un programme contenant plusieurs procédures qui ont été écrites pour d'autres programmes. Il est fort probable que plusieurs d'entre elles auront des variables de même nom. Si ces variables sont globales, vous risquez d'avoir des erreurs et de reprendre toutes les procédures pour modifier le nom des variables incriminées.

Si les variables sont locales, tout ceci n'a aucune importance. Par exemple, la procédure **choisir**, dans le paragraphe sur les erreurs, utilise ce type de variable. Ce qui veut dire qu'il n'est pas besoin de vérifier si **choix\$** existe dans une autre procédure. Les procédures appelées ne peuvent modifier le contenu de **choix\$**.

## LIBELLES

Afficher un libellé et attendre une pression de touche sont les actions les plus courantes. Il est intéressant d'en faire une procédure. Elle doit être capable d'afficher un grand nombre de messages. Une manière simple consiste à les passer sous la forme d'un paramètre.

```
proc libellé; m$
  écris m$+": ";
  que x$=majus(clavier( ))
  écris x$
fproc
```

Le message à afficher est passé à la procédure comme un paramètre dans la variable locale **m\$**. La fonction **clavier( )** attend qu'une touche soit pressée et retourne son code ASCII. Dans cette procédure, ce code est converti en majuscule par **majus( )** de manière à ce que le résultat soit indépendant du mode majuscule ou minuscule. Puis il est assigné à la variable globale **x\$** qui est disponible pour toutes les autres procédures du programme.

Une procédure pratique est **pause**. Elle utilise **libellé** pour imprimer un message et attend simplement une pression de touche. **Pause** emploie la variable locale **y\$** pour préserver le contenu antérieur de **x\$**.

```
proc pause
  note ***** Attend une pression de touche *****
  local y$
  que y$=x$
  écris
  libellé; "Pressez une touche pour continuer"
  que x$=y$
fproc
```

## SAISIE DE DONNEES

### Texte

L'entrée de texte au clavier est très facile. Toute série de caractères est une chaîne valide (même si elle n'a aucun sens). Elle ne causera jamais d'erreur. Vous n'avez donc pas à prendre de précautions spéciales. Il suffit d'utiliser une ligne comme la suivante

```
saisis "Tapez votre nom S.V.P. ";nom$
```

Notez qu'un espace est inclus à la dernière place du texte affiché. Il fera beaucoup de différence dans l'aspect de vos messages.

Vous pouvez saisir plusieurs variables en un seul ordre d'entrée. Tout ce que vous devez faire est d'inclure les libellés et les noms de variables en les séparant par des points-virgules.

```
saisis "Votre nom ? ";nom$; "Votre prénom ? ";prénom$;
```

La ligne se termine par un point-virgule qui empêche le curseur de passer à la ligne après l'entrée.

## Nombres

Quand vous utilisez la commande **saisis** pour assigner un texte à une variable alphanumérique, l'ordinateur accepte tout ce que vous tapez. Si, d'aventure, avec une variable numérique, vous tentez d'entrer autre chose qu'un chiffre valide, vous obtiendrez un message d'erreur. En supposant que vous ne voulez pas que votre programme s'interrompe chaque fois que vos doigts glissent quand vous entrez un nombre, mieux vaut vous assurer que vous gérez bien ce type d'erreur.

Pour ce faire, la manière la plus pratique est la commande **erreur** décrite plus tôt. La procédure suivante accepte n'importe quel nombre valide dans un intervalle donné. Elle autorise aussi l'affichage d'un message.

```
proc entnum; m$,min,max
  note ***** Accepte nombre dans intervalle *****
  local mauvais
  que mauvais=1
  tant que mauvais
    écris m$;" ?";
    erreur litnum
    que mauvais=errnum( )
    si non mauvais
      si nombre< min ou nombre> max
        que mauvais=1
        écris "L'intervalle est de ";min;" à ";max
      fsi
    fsi
  si mauvais
    écris "Essayez encore"
  fsi
ftant que
fproc
```

Comme **erreur** doit être suivi d'un nom de procédure, nous définissons **litnum** qui lit la variable **nombre**.

```
proc litnum
  saisiss nombre
fproc
```

Supposons que vous vouliez une procédure qui vérifie qu'un nombre est dans l'intervalle de 1 à 10. Vous pouvez utiliser **entnum** de la façon suivante :

```
proc interv
  entnum; "Valeur numérique ",1,10
fproc
```

# CHAPITRE ONZE

## FICHIERS MULTIPLES

---

### NOM LOGIQUE DE FICHIER

Nous allons étendre ici nos explications et montrer comment travailler avec deux fichiers ouverts (ou plus). Quand vous avez plus d'un fichier ouvert en même temps, vous devez être en mesure de savoir lequel employer pour une opération particulière. Il faut donner à chacun un **nom logique** unique quand vous l'ouvrez, ou quand vous le créez. C'est par ce nom que vous y ferez référence dans toutes les commandes qui le manipulent.

Quand il s'agit d'un fichier unique, **ARCHIVE** fournit automatiquement, à l'ouverture, le **nom logique "maître"**. Il est dit **logique** pour le différencier de son nom **physique** sous lequel vous l'avez sauvegardé.

Grâce aux noms logiques, un programme peut travailler avec des fichiers multiples. Vous ne pouvez ouvrir plusieurs fichiers qu'en utilisant ensemble leur nom physique et leur nom logique. Notez que le nom logique n'est pas sauvegardé avec le fichier à sa fermeture et doit être précisé à chaque ouverture.

Plusieurs fichiers peuvent contenir des champs de même nom. Quand c'est le cas, l'identification du fichier, auquel appartient le champ, se fait grâce à son nom logique. Par exemple, si le champ **country\$** apparaît dans deux fichiers dont les noms logiques sont "**maître**" et "**b**", vous pouvez sélectionner chacun des champs respectivement par "**maître.country\$**" et "**b.country\$**".

Ce premier exemple indique comment ajouter, effacer ou renommer des champs dans un fichier existant.

Supposez que vous vouliez effectuer quelques modifications dans le fichier "**PAYS**", pour créer un nouveau fichier ne comprenant que les pays d'Europe. Le champ "**continent\$**" devient inutile. D'autre part, nous changerons le champ "**pop**" en "**population**".

La meilleure manière de modifier le fichier est d'en créer un second contenant les champs désirés, puis d'y recopier les enregistrements utiles de l'ancien fichier. Appelons le nouveau fichier "**Europe**". La procédure suivante fera le reste du travail.

```
proc commence
note ***** Crée le fichier Europe
crée "Europe" logique "e"
country$
capital$
language$
currency$
population
dgp
area
```



```

fcréé
lis "PAYS" logique "g"
  isole continent$="EUROPE"
  partout
    écris au 0,0;g.country$;tab 30
    que e.country$=g.country$
    que e.capital$=g.capital$
    que e.language$=g.language$
    que e.currency$=g.currency$
    que e.population=g.pop
    que e.gdp=g.gdp
    que e.area=g.area
    ajoute "e"
  fpartout
  ferme "e"
  ferme "g"
  écris
  écris "Terminé"
fproc

```

## LE FICHIER COURANT

Vous voyez dans cet exemple qu'il est possible d'avoir les mêmes noms de champs dans deux fichiers et qu'ils peuvent être distingués en leur ajoutant le nom logique. Si vous ne précisez pas de cette manière le fichier d'appartenance, il est supposé qu'il s'agit du fichier courant. Le fichier courant est le dernier fichier ouvert, ici, **"PAYS"** (avec comme nom logique **"g"**). Pour éviter toute confusion, il est plus sûr d'inclure le nom logique.

Cependant, au moyen de la commande **active**, vous pouvez à tout moment définir le fichier courant. Dans l'exemple précédent, avec :

```
active "e"
```

**"Europe"** serait devenu le fichier courant jusqu'à un nouveau changement, à l'ouverture d'un autre fichier ou à l'utilisation de **active**.

## GESTION DE STOCK

Maintenant, voyons un exemple plus complexe. Dans un système de gestion de stock nous devons :

- Trouver une information sur un produit précis.**
- Obtenir la quantité en stock de tous les produits.**
- Enregistrer les ventes et modifier les stocks.**
- Faire les demandes de réapprovisionnement du stock.**
- Enregistrer les approvisionnements.**

Vous avez obligatoirement besoin d'un fichier contenant les détails des produits que vous avez en stock et il est intéressant de posséder un fichier de tous vos fournisseurs. Vous devez être capable d'accéder à un fichier à partir de l'autre - si vous désirez obtenir la liste de tous vos fournisseurs d'un produit donné, ou savoir quels sont les produits que vous fournit telle société.

Contrairement aux précédentes applications, nous n'allons pas utiliser de menu, pour rester aussi simples que possible. Nous allons écrire une série de commandes qui, comme les commandes usuelles, s'emploieront en tapant leur nom.

Les procédures dépendent fortement de la structure des fichiers utilisés, il est donc primordial de les voir d'abord.

### Le fichier stock

Le fichier **stock** doit contenir la position du stock pour chaque produit. La liste suivante donne tous les champs que nous utiliserons.

Nom du champ	Utilisation	Exemple
<b>stockno\$</b>	<b>Code interne au stock</b>	<b>A101</b>
<b>description\$</b>	<b>Description du produit</b>	<b>Stylo bille</b>
<b>qté</b>	<b>Nombre en stock</b>	<b>500</b>
<b>prixvte</b>	<b>Prix de vente</b>	<b>1.25</b>
<b>nivappro</b>	<b>Seuil minimum de réapprovisionnement</b>	<b>200</b>
<b>achatqté</b>	<b>Quantité à acheter</b>	<b>400</b>

Nous pouvons créer le fichier avec :

```

crée "stock" logique "sto"
  stockno$
  description$
  qté
  nivappro
  prixvte
  achatqté
fcrée

```

### Le fichier Fournisseurs

Ce fichier contient les noms, adresses et numéros de téléphone des sociétés qui vous fournissent les marchandises que vous vendez. Il est utile d'y ajouter le nom de votre intermédiaire dans ces sociétés. Pour accéder à ces informations efficacement, nous inclurons un code pour chacune des sociétés. Voici la structure du fichier :

Nom du champ	Utilisation	Exemple
<b>nomsté\$</b>	<b>Nom de la société</b>	<b>Durand et Fils</b>
<b>rue\$</b>	<b>1ère ligne d'adresse</b>	<b>27 rue Paname</b>
<b>ville\$</b>	<b>2ème ligne d'adresse</b>	<b>PARIS</b>
<b>département\$</b>	<b>3ème ligne d'adresse</b>	<b>Ile de France</b>
<b>code_postal\$</b>	<b>4ème ligne d'adresse</b>	<b>75018</b>
<b>contact\$</b>	<b>Nom du contact</b>	<b>André Martin</b>
<b>tel\$</b>	<b>Numéro de téléphone</b>	<b>16 1 842 17 89</b>
<b>code\$</b>	<b>Votre code pour la société</b>	<b>a</b>

Créons le fichier :

```
crée "fournis" logique "frn"  
  nomsté$  
  rue$  
  ville$  
  département$  
  code_postal$  
  contact$  
  tel$  
  code$  
fcrée
```

### Fichier d'approvisionnements

Ce fichier forme le lien entre les deux précédents. Voici la liste de ses champs :

Nom du champ	Utilisation	Exemple
<b>stonum\$</b>	<b>Votre code stock</b>	<b>A101</b>
<b>code\$</b>	<b>Code fournisseur</b>	<b>a</b>
<b>fcode\$</b>	<b>Code du produit chez le fournisseur</b>	<b>123-456</b>
<b>prix</b>	<b>Prix d'achat</b>	<b>0,87</b>
<b>délai</b>	<b>Délai de livraison en jours</b>	<b>28</b>

Chaque enregistrement de ce fichier est lié avec un enregistrement du fichier "**stock**" et un du fichier "**fournis**". L'exemple ci-dessus montre que Durand et Fils (fournisseur code "a") vous fournit en stylos bille (code stock Aloi). Nous avons encore inclus des détails sur le prix, le délai de livraison et le code du produit chez le fournisseur. Ces informations sont utiles quand vous commandez des produits.

L'utilisation de ce fichier vous permet de vous approvisionner dans les cas où un fournisseur propose plusieurs produits en stock (**code\$** égaux et **stockno\$** différents) et quand un produit est disponible chez plusieurs revendeurs (**stockno\$** égaux et **code\$** différents).

Le fichier est :

```
crée "appros" logique "apr"  
  stockno$  
  code$  
  fcode$  
  prix  
  délai  
fcrée
```

Après la création de ces fichiers, il vous faut des procédures pour en manipuler les informations. Vous allez voir que l'option la plus demandée est de trouver des renseignements sur un produit précis pour répondre à un client, le plus rapidement possible, même à partir d'une demande incomplète. Nous utiliserons la commande **trouve** qui cherche toutes les occurrences de la demande.

La procédure pourra demander une confirmation dans la sélection d'un enregistrement. Nous déléguerons cette tâche à une autre procédure qui nous servira dans d'autres situations.

```
proc confirme
  écris:écris "Confirmez (o/n)"; que oui=minus(clavier( ))="o"
  éponge
fproc
```

Elle assigne à la variable **oui** la valeur **1** si la touche "**o**" est pressée, sinon la valeur zéro. Notez que le signe = sert pour une assignation et pour le test logique (vrai ou faux).

```
proc demande
  note ***** Demande le nom du produit *****
  écris
  saisis "Produit ?";nom$
  active "sto"
  trouve nom$
  que oui=0
  tantque vu( ) et non oui
    affiche
    montre
    confirme
    si non oui
      continue
    fsi
  ftantque
  si non vu ( )
    écris
    écris nome n'existe pas"
  fsi
fproc
```

La procédure trouve seulement l'enregistrement correct.

Une autre procédure utile est **question** :

```
proc question
  demande
  lstcom
fproc
```

Elle utilise une procédure **lstcom** qui attend une pression de touche, efface l'écran et imprime la liste des commandes disponibles. Nous l'oublierons jusqu'à ce que nous ayons écrit les procédures dont elle devra lister les noms. Quittez **édite** de temps en temps pour sauvegarder ces procédures.

## Etat du stock

Nous allons écrire une procédure simple qui produit un état général du stock.

```
proc état
note ***** état du stock *****
éponge
écris tab 2;"Produit";tab 11;"Code";
écris tab 20;"Quantité";tab 31;"Prix"
écris tab 40;"Valeur du Stock";
écris
que total=0
active "sto"
partout
  écris description$(jusqu'à 10);tab 11;sto.stockno$;
  tab 20;qté;
  écris tab 31,"Frs";prixvte;tab 40;"Frs";prixcte*qté
  que total=total+prixvte*qté
fpartout
écris
écris "Valorisation du Stock";total;"Frs"
Pstcom
Fproc
```

## Mise à jour des ventes

Pour enregistrer une vente, il suffit de soustraire du stock la quantité de produit vendue. Il est conseillé de demander une confirmation pour s'assurer qu'il s'agit bien du bon stock et qu'il reste assez du produit pour satisfaire la demande.

```
proc quantité
note ***** imprime produits en stock *****
demande
ecris
saisis "Combien ?";nombre
écris
éponge
écris nombre;"*";sto.stockno$;" (";sto.description$;")"
fproc

proc vente
note ***** traite vente *****
quantité
si nombre < sto.qté
  écris "Montant",nombre*sto.prixvte
  confirme
si oui
  que sto.qté=sto.qté-nombre
  change
  montre: note *** montre l'enregistrement modifié***
fsi
sinon
  écris "Pas assez de stock"
fsi
lstcom
fproc
```

## Mise à jour des approvisionnements

La procédure suivante permet d'enregistrer une entrée en stock. Elle demandera aussi une confirmation avant la mise à jour.

```
proc rentrée
note ***** entrée en stock *****
quantité
confirme
écris
si oui
  écris "Accepté"
  que sto.qté=sto.qté+nombre
  change
  montre
sinon
  écris "Entrée annulée"
fsi
lstcom
fproc
```

Jusqu'ici, nos procédures ne se sont référées qu'au fichier **stock**. Quand nous désirons réapprovisionner notre stock nous devons faire appel aux fichiers **fournis** et **appros** pour obtenir les noms et adresses des sociétés, les prix, etc...

Après avoir localisé le produit du stock (avec **demande**), nous sélectionnons dans les autres fichiers, à l'aide du code, toutes les sociétés qui peuvent livrer le produit. Comme l'enregistrement contient aussi le prix et le délai de livraison pour chaque fournisseur, nous pouvons choisir entre le produit le moins cher et la livraison la plus rapide.

Nous utilisons **situe** comme moyen le plus rapide de localiser l'enregistrement d'un fournisseur donné. Ce qui implique que **fournis** doit être trié (en fonction du code fournisseur, **code\$**) avant d'exécuter **faittri**.

```
proc faittri
note ***** trie le nouveau stock *****
demande
active "apr"
isole sto.stockno5=apr.stockno$
écris
écris "Rapidité ou Meilleur prix (r/m)";
si minus (clavier( ))="r"
  rapide
sinon: mscher
fsi
que vcode$=fcode$
restaure
active "frn"
situe comp$
faitfac
écris
écris "Délai de livraison";del;" jours"
lstcom
fproc
```

La procédure **mscher** trouve le fournisseur au meilleur prix et **rapide** celui dont les délais sont les plus courts. Elles fonctionnent de la même manière.

```
proc mscher
note ***** Trouve le moins cher
active "apr"
que pri=prix
que comp$=code$
que del=délai
partout
si prix<pri
que prix=prix
que comp$=code$
que del=délai
fsi
fpartout
fproc

proc rapide
note ***** Trouve livraison la plus rapide
active "apr"
que del=délai
que comp$=code$
que pri=prix
partout
si délai<del
que del=délai
que comp$=code$
que pri=prix
fsi
fpartout
fproc
```

La procédure **faitfac** produit le formulaire de commande. Vous pouvez le modifier selon vos besoins. Nous utiliserons une version simple qui affiche les détails à l'écran.

```
proc faitfac
note ***** Edite la commande *****
éponge
écris
écris frn.nomsté$
écris frn.rue$
écris frn.département$
écris codepostal$
écris
écris "Commande de";sto.achatqté;
écris " * produit n° ";vcode$
écris "(";sto.description$;") ";
écris "au prix unitaire de ";pri;" Frs" écris
écris "Total ";sto.achatqté*prie Frs"
fproc
```

La dernière commande sera celle qui fermera tous les fichiers quand nous aurons fini de les utiliser.

```
proc salut
  confirme
  si oui
    éponge
    écris:écris "Au revoir"
    ferme "sto"
    ferme "apr"
    ferme "frn"
    éponge
  fsi
fproc
```

Ecrivons maintenant une courte procédure qui lance l'application. Elle doit ouvrir les trois fichiers avec les noms logiques corrects, effacer l'affichage et vous montrer les commandes qui sont disponibles. Dans un usage normal, le fichier stock est le seul dont les données peuvent être modifiées. Les deux autres fichiers n'ont besoin d'être ouverts qu'à la lecture. Elle trie aussi le fichier fournisseurs pour que nous puissions utiliser **situe** pour trouver une société avec son code de référence.

```
proc début
  éponge
  écris au 5,5;"Démonstration de Gestion de Stock"
  écris
  ouvre "stock" logique "sto"
  lis "fournis" logique "frn"
  lis "appro" logique "apr"
  active "fn'"
  trie code$c
  lstcom
fproc
```

Finalement, écrivons lstcom qui efface simplement l'écran et liste les noms des commandes disponibles.

```
proc lstcom
  note ***** Efface l'écran et liste commandes *****
  local x$
  écris
  écris "Presser une touche pour continuer"
  que x$=clavier( )
  éponge
  écris
  écris "Question Etat Rentrée Faittri Vente Salut"
  écris:écris "Tapez votre choix"
fproc
```



# CHAPITRE DOUZE

## REFERENCES

---

### VARIABLES

Les noms de variables comportent jusqu'à treize caractères, et ne doivent pas commencer par un chiffre (0 à 9). Ils peuvent être un mélange de majuscules, de minuscules et de chiffres. Les autres caractères ne sont pas admis, sauf le dollar \$ et le point . qui ont une signification particulière.

Si un nom de variable se termine par un \$, c'est une variable alphanumérique. Les chaînes admettent jusqu'à 255 caractères en longueur. Si le nom ne se termine pas par \$, la variable est numérique. Un nom de variable peut se référer au contenu d'un enregistrement, pour devenir une variable de champ. Les variables de champ se réfèrent normalement au fichier courant, ou à un autre si son nom est préfixé par le nom logique du fichier suivi d'un point. Une variable de champ s'écrit donc :

nom logique fichier.nom champ

par exemple **maître.continent\$**. Si un nom de variable contient un point, il doit se rapporter à un champ de fichier ouvert. S'il n'y a pas de point, le nom de la variable est recherché dans l'ordre suivant :

1. un champ du fichier courant
2. une variable locale (un paramètre de la procédure courante s'il existe)
3. une variable globale

Un message d'erreur est affiché, si elle n'est pas trouvée.

### SYNTAXE

Le terme **syntaxe** représente la structure exacte d'une commande ou d'une fonction. La syntaxe d'une commande spécifie les paramètres dont elle a besoin, l'ordre dans lequel ils doivent être fournis et les symboles (si nécessaire) qui les séparent.

### EXPRESSIONS

Ce paragraphe décrit la notation utilisée pour exprimer la syntaxe du langage de programmation **d'ARCHIVE**.

Une **expression** est une combinaison de valeurs littérales, de variables, de fonctions et d'opérateurs qui résultent en une valeur numérique. Une **expression numérique** donne un résultat numérique et une **expression alphanumérique** donne une chaîne de caractères. Exemples :

$3 * y * \sin(x) + \text{long}(a\$)$  (numérique)  
"abc" + a\$ + rept(" - ", 5) (chaîne)

Une expression peut, comme dans les exemples précédents, être composée de sous-expressions. Auquel cas, vous ne devez pas mélanger différents types.

### Conventions de syntaxe

La notation est identique à celle utilisée pour le **SUPERBASIC**

<b>Symbole en italique</b>	<b>Signification</b>
( )	montre une unité syntaxique
( )	entourent un paramètre optionnel
* *	entourent des paramètres répétés
ou [ ]	ou bien
{ } ou ( )	commentaires

### Unités syntaxiques

<b>c.car</b>	chaîne de caractères
<b>c.exp</b>	expression alphanumérique
<b>n.exp</b>	expression numérique
<b>exp</b>	expression (numérique ou alphanumérique)
<b>écr</b>	paramètre d'affichage
<b>var</b>	nom de variable, chaîne ou nombre
<b>nlf</b>	nom logique de fichier
<b>npf</b>	nom physique de fichier (8 caractères max)
<b>ndp</b>	nom de procédure.

Une chaîne de caractères est du texte entouré de guillemets, par exemple 'texte' ou "texte".

Une expression alphanumérique est une chaîne de caractères, une combinaison de chaînes, une variable alphanumérique ou une fonction dont le résultat est de type alphanumérique. Exemple :

```
"jean"+a$+chr$(72)
```

Une expression numérique est un nombre, une combinaison de nombres, de variables numériques et d'opérateurs (+, -, \*, /, etc...) qui aboutissent à un résultat numérique. Exemple :

```
(3+x)/sin(y)
```

Un paramètre d'affichage est l'un des quatre mots-clés suivants : **au**, **tab**, **encre**, **papier**. Voici la description complète de leur syntaxe :

```
écris_paramètre:=[au n.exp, n.exp]
                  [tab n.exp]
                  [encre n.exp]
                  [papier n.exp]
```

Les noms logiques de fichier et les noms de procédures ont les mêmes restrictions que les noms de variables. Les noms physiques de fichier ne doivent pas excéder 8 caractères.

Comme exemple de définition de syntaxe, voyons l'ordre **trie**. Ce qui donne :

```
trie spec * (,trie spec ) * trie spec:-- var; ci d
```

**Trié** doit être suivi d'au moins une spécification de tri qui est elle-même composée d'une variable séparée par une virgule d'un **c** ou d'un **d**. De plus, vous pouvez ajouter 3 autres spécifications supplémentaires en les séparant par des virgules.

La notation syntaxique ne donne pas la signification ou l'utilisation des symboles. Donc lisez la description des commandes. La syntaxe ne vous donne que le nombre et le type de paramètres à fournir pour obtenir une commande valide. Le nombre de répétitions n'est pas non plus indiqué dans notre notation. **Trie** accepte jusqu'à 4 paires variables/lettres.

## LES FICHIERS DE DONNEES D'ARCHIVE

### Un champ

Un champ est une place réservée pour une chaîne ou un nombre.

Dans **ARCHIVE**, chacun est identifié par le nom d'une variable de champ. Le type du champ dépend du nom de la variable - alphanumérique si le nom se termine par \$. Un champ de ce type admet jusqu'à 255 caractères de long. Un champ numérique possède un nom ne se terminant pas par \$. Tous les nombres occupent la même place en mémoire, indépendamment de leur valeur.

### Un enregistrement

Un enregistrement est un ensemble de champs dont les contenus sont en relation d'une manière ou d'une autre. Les champs peuvent, par exemple, contenir le nom, l'adresse et le numéro de téléphone d'une personne. Dans **ARCHIVE** les enregistrements sont de longueur variable pour n'occuper que la mémoire nécessaire. Un enregistrement peut comporter jusqu'à 255 champs.

### Un fichier

Un fichier est composé d'enregistrements liés. Dans notre exemple, un fichier de données serait composé d'enregistrements concernant plusieurs personnes. **ARCHIVE** gère des fichiers d'au plus 15000 enregistrements. En pratique, vous n'êtes limité que par la place disponible sur une cartouche de Microdrive, qui ne peut contenir que 1000 enregistrements de 100 caractères. Le fichier est l'entité de base que vous pouvez charger ou sauvegarder d'un Microdrive. Chaque fichier est identifié par son nom. Dans **ARCHIVE**, vous donnez un nom physique au fichier lors de la création, mais vous pouvez changer le nom logique à tout moment.

### Ouverture et Fermeture de Fichiers

Pour lire ou écrire des données sur un fichier il faut l'**ouvrir**. Sans entrer dans des détails, l'ouverture transfère une copie du fichier de la cartouche du Microdrive vers la mémoire centrale. Si le fichier est très volumineux, seule une partie est copiée.

Par la commande **lis**, vous n'ouvrez un fichier qu'à la lecture et il n'est pas possible de modifier son contenu. Pour y lire et y écrire à la fois, il faut utiliser **ouvre**. Chaque fois que vous ouvrez un fichier, **ARCHIVE** réserve de la place pour les variables de champ d'un enregistrement.

Les variables de champ contiennent toujours les valeurs de l'enregistrement courant. Quand vous fermez un fichier avec **ferme** ou **quitte**, toutes les modifications effectuées sur le fichier sont reportées sur la cartouche du Microdrive. La copie en mémoire est effacée. Seule la fermeture d'un fichier vous assure que la cartouche contient la dernière version. Comme un fichier ouvert occupe de la place en mémoire, il est préférable de le fermer quand vous ne l'utilisez plus.

Quand vous abandonnez **ARCHIVE** avec **quitte**, tous les fichiers encore ouverts sont fermés.

**Ne débranchez pas votre ordinateur et ne retirez pas la cartouche d'un Microdrive tant que des fichiers sont encore ouverts.**

## Noms logiques de Fichier

Chaque fichier ouvert possède un **nom logique associé** qui lui a été donné lors de l'ouverture. Si vous ne précisez pas de nom logique, il lui est automatiquement donné celui de "**maître**". Le nom logique identifie un fichier particulier lors de l'utilisation de fichiers multiples.

## PROCEDURES

Une procédure est un groupe d'instructions doté d'un nom. Elle débute par une déclaration de la forme :

```
proc ndp [ ; var * [,var] *
```

et se termine par :

```
fproc
```

Elle est appelée par son nom à partir d'un programme, d'une autre procédure ou encore d'elle-même. Cela fonctionne comme si elle se trouvait à l'endroit d'où elle est appelée.

Dans **ARCHIVE**, **proc** et **fproc** ne peuvent être entrés au clavier. Ils sont ajoutés automatiquement lors de la création d'une procédure.

## L'EDITEUR DE PROGRAMME

On accède à l'éditeur par la commande **édite**.

Si aucune procédure n'est présente en mémoire, la création vous est immédiatement proposée. Sinon, la liste de toutes les procédures sera affichée sur la gauche de l'écran. La première d'entre elles sera mise en évidence et la liste de ses instructions apparaîtra à droite. La première ligne de la procédure est signalée comme ligne courante.

Une fois dans **édite** vous avez quatre options :

### Choisir une procédure :

Pressez (**TAB**) pour vous déplacer vers le bas dans la liste des procédures ; pressez (**SHIFT**) et (**TAB**) pour vous déplacer vers le haut. La liste d'instructions sur la droite est toujours celle de la procédure courante.

### Choisir une ligne

Pressez (**F3**) pour obtenir le menu des commandes d'édition. Il y a quatre commandes qui sont choisies par leur initiale tapée au clavier.

- |                 |  |
|-----------------|--|
| <b>Efface</b>   | Presser ( <b>ENTREE</b> ) pour effacer la procédure courante mise en évidence sur la gauche de l'écran, ou une autre touche pour annuler l'effacement.   |
| <b>Création</b> | Entrez le nom de la procédure et pressez (ENTREE). Si une procédure de ce nom existait déjà, vous aurez la possibilité de l'éditer.  |
| <b>Extraire</b> | Extrait du texte d'un endroit pour le mettre dans un tampon de travail. Les touches de déplacement vertical du curseur vous permettent de choisir le début, puis la fin de la zone à extraire. Pressez (ENTREE) pour effectuer l'extraction. |
| <b>Inclure</b>  | Copie sous la ligne courante le contenu du tampon d'extraction puis l'efface.  |

### Insérer du texte

Pressez (**F4**) pour insérer une ou plusieurs lignes au-dessous de la ligne courante de la procédure courante. Tapez le texte et pressez (**ENTREE**). (**ENTREE**) vous gardera en mode insertion si aucun texte ne le précède.

### Editer du texte

Pressez (**F5**) pour éditer la ligne courante de la procédure courante. La ligne est copiée sur le bas de l'écran et peut être modifiée avec l'éditeur de ligne. (**ENTREE**) remplacera l'ancienne ligne par la nouvelle.

## L'EDITEUR D'ECRAN

On accède à l'éditeur d'écran par la commande **éditem**. Il vous permet de définir une nouvelle présentation des données ou d'en modifier une ancienne. Ceci fait, vous pouvez la sauvegarder, grâce à **sauvem**, ou en charger une, grâce à **chargem**.

Une image écran est composée de deux parties. L'arrière-plan fixe et les valeurs des variables qui y seront affichées. La commande **écran** montre l'arrière-plan et la commande **montre** affiche le contenu des variables.

**éditem** admet deux options :

- l'entrée de texte d'arrière-plan
- l'utilisation d'une commande d'édition écran (F3)

Quatre commandes d'édition écran sont disponibles :

- **E** Efface l'écran
- **M** Marque la zone d'affichage d'une variable
- **E** Fixe la couleur de l'encre
- **P** Fixe la couleur du papier

Un dessin d'écran est activé par :

```
chargem  
écran
```

Quand un écran particulier est actif, ses variables sont affichées après **montre**, ou quand le contrôle est rendu au clavier, après l'exécution d'un programme ou d'une commande. Un écran est désactivé par la commande **éponge**. S'il n'y a pas d'écran actif, **montre** reste sans effet. Il ne peut résider en mémoire qu'un écran à la fois.

La commande **affiche** crée et affiche en priorité sa propre présentation d'écran.

## LES COMMANDES

Dans **ARCHIVE**, les commandes suivantes sont disponibles :

### PARTOUT

Permet la recherche dans les enregistrements présents dans le fichier, de la manière la plus rapide.

```
Syntaxe : partout [ nlf ] : ... : fpartout
```

Cette recherche n'est pas particulière. Le nom logique de fichier indique dans quel fichier ouvert elle est à faire. Si aucun nom logique n'est donné, elle s'effectuera dans le fichier courant.

La boucle **partout** est étudiée pour l'examen d'enregistrements plutôt que pour leur modification. N'utilisez pas **change** à l'intérieur d'une boucle **partout**, sauf si vous êtes sûr que la longueur de l'enregistrement n'est pas modifiée. Vous pouvez par exemple changer la valeur d'un nombre ou convertir un texte en majuscules. Dans le doute, utilisez une boucle **tantque** - en testant la valeur de **fdf( )** pour détecter la fin de fichier.

Exemple :

```
début
tantque non fdf( )
...
change
...
suiv
ftantque
```

## MODIFIE

Modifie la présentation d'écran courante pour l'affichage des variables.

Syntaxe : `modifie`

Vous pouvez modifier le contenu de n'importe quel champ du fichier courant, dont la valeur est affichée. Il n'est pas nécessaire que toutes les variables soient montrées, mais elles seules peuvent être changées. Si l'affichage est vide, **ARCHIVE** exécutera un **affiche** de l'enregistrement courant.

Choisissez le champ avec **(TAB)** ou **(ENTREE)** jusqu'à ce que le curseur se positionne devant le champ voulu (les variables qui n'appartiennent pas au fichier sont sautées). Vous pouvez alors entrer la nouvelle valeur ou utiliser l'éditeur de ligne pour modifier une valeur existante. **(TAB)** ou **(ENTREE)** vous amène au champ suivant.

**(SHIFT)** et **(TAB)** à la fois vous positionnent sur le champ précédent.

Après vos modifications, **(F5)** remplacera l'ancien enregistrement par le nouveau. L'enregistrement est automatiquement mis à jour si vous pressez **(ENTREE)**. Si le fichier est trié, la mise à jour conserve l'ordre.

## AJOUTE

Ajoute un enregistrement au fichier spécifié, ou au fichier courant si aucun nom logique n'est donné.

Syntaxe : `ajoute [nlf]`

Les champs du fichier prennent les valeurs des variables de champs courantes. Si le fichier est trié, l'ordre est conservé.

## **PREC**

Revient à l'enregistrement précédent du fichier spécifié ou du fichier courant si aucun nom logique n'est donné.

Syntaxe : `préc [nlf]`

## **SAUVEGARDE**

Effectue une copie d'un fichier spécifié. Vous pouvez copier tous vos fichiers pour vous protéger contre des effacements ou de mauvaises manipulations.

Syntaxe : `sauvegarde ancien_nom en nouveau_nom`

## **FERME**

Ferme le fichier spécifié ou le fichier courant si aucun nom logique n'est donné.

Syntaxe : `ferme [nlf]`

## **EPONGE**

Efface la zone d'affichage et désactive tout dessin d'écran.  
Voir **écran**, **écharge** et **monte**.

Syntaxe : `éponge`

## **CONTINUE**

Poursuit le **cherche** ou **trouve** précédent à partir de l'enregistrement courant du fichier courant.

Syntaxe : `continue`

## **CREE**

Crée un fichier doté d'un nom, dont les enregistrements contiennent des champs donnés dans une liste de variables spécifiées dans la commande. Si aucun nom logique n'est fourni, celui de "**maître**" lui est donné.

Syntaxe : `crée npf [logique:nfp] : var *[:var] *:fcrée`

## **EFFACE**

Efface l'enregistrement courant d'un fichier spécifié ou du fichier courant si aucun nom logique n'est donné.

Syntaxe : `efface [nfl]`

## CAT

Affiche la liste des fichiers d'une cartouche de Microdrive.

Syntaxe : `cat [ lecteur ]`

**Lecteur** est **mdv1\_** ou **mdv2** . Si rien n'est précisé, **ARCHIVE** choisira le Microdrive 2. Avant la liste des fichiers, le nom de la cartouche (celui donné au formatage) sera affiché.

## AFFICHE

Affiche le nom logique du fichier courant et la liste de ses champs accompagnés des valeurs des variables de champ de l'enregistrement courant. Si le fichier est trié, les clés de tri sont signalées ainsi que leur priorité.

Syntaxe : `affiche`

Cette commande remplace tout écran actif par cette liste qui reste active.

## ETAT

Envoie les champs spécifiés, d'enregistrements donnés du fichier courant, sous forme de tableau, vers la sortie série **ser1**. Si vous n'avez précisé aucune variable, tous les champs sont imprimés.

Syntaxe : `état [ ; var ] * [ ,var ]*`

Vous pouvez vectoriser la sortie vers un fichier Microdrive avec **via**.

## EDITE

Appelle l'éditeur de procédure pour en créer une nouvelle ou en éditer une ancienne.

Syntaxe : `édite`

**FPARTOUT** voir **partout**

**FCREE** voir **créé**

## ERREUR

Syntaxe : `erreur ndp [ ;exp * [,exp]* ]`

Marque la procédure de gestion d'erreur. Toute erreur qui survient durant l'exécution de la procédure contenant cette commande ou toute procédure qu'elle appelle, force un retour à la procédure marquée. Celle-ci obtient le numéro de l'erreur en consultant le contenu de **errnum( )**. Le numéro est remis à zéro chaque fois que **erreur** est exécuté.



## EXPORTE

Sauvegarde les champs spécifiés d'enregistrements donnés du fichier courant sur une cartouche de Microdrive, dans un format que **QL ABACUS** et **QL EASEL** peuvent **importer**.

Syntaxe : `exporte ndf [ ;var ] * [ ,var ] * [ QUILL ]`

Si vous ne spécifiez aucun champ, tous sont exportés. Le paramètre optionnel **QUILL** (séparé par un espace au moins du dernier nom de variable) exporte le fichier dans un format compatible à **QL QUILL**.

Le fichier est **exporté** sous le nom **ndf**, et si aucune extension de nom n'est précisée, **ARCHIVE** utilise **\_exp**.

Voir dans le chapitre INFORMATIONS tous les détails sur **importe** et **exporte**.

## TROUVE

Recherche à partir du début du fichier la première occurrence d'une chaîne spécifiée. La recherche ne dépend pas de la calligraphie.

Syntaxe : `trouve c.exp`

Vous pouvez poursuivre la recherche avec **continue** et déterminer si elle est fructueuse en examinant la valeur retournée par **vu ( )**.

## DEBUT

Trouve le premier enregistrement du fichier spécifié ou du fichier courant si aucun nom logique n'est donné.

Syntaxe : `début [ nlf ]`

## FORMATE

Formate la cartouche du Microdrive 2 (celui de droite) et lui donne le nom spécifié. Ce nom est affiché quand vous utilisez la commande **cat**.

Syntaxe : `formate "vos paramètres"`

## SI

Autorise le contrôle d'une séquence d'ordres à une condition.

Syntaxe : `si n.exp : ... [ sinon : ... ] : fsi`

### Sans sinon :

Si l'expression est différente de zéro, les ordres qui suivent sont exécutés. Si l'expression est égale à zéro, l'exécution est transférée après le **fsi**.

### Avec sinon :

Si l'expression est différente de zéro, les ordres entre le **si** et le **sinon** sont exécutés. Sinon, les ordres entre le **sinon** et le **fsi** sont exécutés. Dans les deux cas, le déroulement du programme se poursuit avec les instructions qui suivent le **fsi**.

## IMPORTE

Lit le fichier **nom1**, exporté de **QL ABACUS** ou **QL EASEL** et produit un fichier de données **ARCHIVE** nommé **nom2**. Comme avec **ouvre** et **lis**, vous pouvez spécifier un nom logique.

```
Syntaxe : importe nom 1 dans nom2 [ logique nlf ]
          où  nom1:=ndf
              nom2:=ndf
```

Voir le chapitre [INFORMATION](#) pour une description complète de **importe** et **exporte**.

## ENCRE

Fixe la couleur de l'encre spécifiée par la valeur de l'expression, pour tout le texte qui suit.

Syntaxe : **encre n.exp**

Les couleurs sont : 0 et 1 Noir  
2 et 3 Rouge  
4 et 5 Vert  
6 et 7 Blanc

Si l'expression est supérieure à 7, la valeur prise en compte sera le reste de la division par 8. Par exemple, **encre 41** est équivalent à **encre 1**, ce qui fixera dans les deux cas l'encre à noir. Si l'ordre **encre** est utilisé au sein d'une commande **écris**, le changement de couleur ne persistera que le temps de cette commande.

## SAISIS

Attend du clavier les valeurs à assigner aux variables spécifiées dans la commande. Chaque nom de variable peut être précédé d'un libellé qui sera affiché comme signal avant la saisie. Tous les paramètres de saisie doivent être séparés les uns des autres par des points-virgules. Si la liste se termine par un point-virgule, le curseur ne passera pas à la ligne suivante à la fin de l'entrée.

```
Syntaxe : saisis [var | c.lit | écr * [;var| c.lit| écr ]* [ ; ]
```

La liste des paramètres d'affichage peut contenir des ordres de positionnement du curseur.

```
au ligne,colonne
tab colonne
ou
ligne:=n.exp
colonne:=n.exp
```

**Au** positionne le curseur à la ligne et à la colonne spécifiées et **tab** à la colonne spécifiée si elle est la droite de la position d'écriture, sinon **tab** reste sans effet. **Au** et **tab** ne peuvent pas être employés en dehors des ordres **écris** et **saisis**.

## INSERE

Ajoute un nouvel enregistrement au fichier.

```
Syntaxe : insère
```

Voir **modifie**.

## TUE

Efface un fichier de la cartouche d'un Microdrive.

Syntaxe : `tue ndf`

**Attention** : Utilisez cette commande avec précautions, car un fichier effacé n'est plus récupérable.

## FIN

Trouve le dernier enregistrement du fichier courant ou d'un fichier spécifié si le nom logique est donné.

Syntaxe : `fin [ nlf ]`

## QUE

Assigne à une variable une valeur spécifiée (comme le LET du SUPER BASIC)

Syntaxe : `que var = exp`

## LISTE

Liste toutes les procédures en mémoire sur l'imprimante.

Syntaxe : `liste`

## CHARGE

Charge en mémoire à partir de la cartouche du Microdrive le fichier de procédures dont le nom est spécifié.

Syntaxe : `charge [ objet ] ndf`

L'option **objet** signale un fichier binaire au lieu d'un fichier ASCII.

## LOCAL

Dans une procédure, signale que les variables qui suivent sont locales. Ces variables ne sont reconnues que dans la procédure qui les a déclarées. Leurs valeurs sont perdues à la fin de la procédure.

Syntaxe : `local var * [ ,var ] *`

## SITUE

Trouve, dans un fichier trié, le premier enregistrement qui contient le contenu de l'expression (des expressions).

Syntaxe : `trouve exp * [ , exp ] *`

L'enregistrement est trouvé beaucoup plus rapidement qu'avec **trouve**, mais le fichier **doit être trié**. Chaque expression doit se rapporter au contenu d'un champ de tri. Si le champ est alphanumérique, l'occurrence dépend du cas.

Si vous avez trié le fichier avec plus d'une clé de tri, vous pouvez spécifier plusieurs expressions (une par clé). Les expressions doivent être séparées par des virgules et dans le même ordre que dans le précédent ordre **trie**. Exemples :

```
trie animal$; c , poids ; c
situe "Eléphant" , 2000
```

trouvera le premier enregistrement dont le champ **animal\$** contient le texte "Eléphant" et dont le champ **poids** contient ou excède 2000. Si aucune réponse exacte est trouvée, **trouve** pointera sur l'enregistrement dont les champs sont supérieurs aux valeurs spécifiées (dans le sens du tri, "d" est supérieur à "e" si le fichier est trié en ordre décroissant).

## LIS

Ouvre à la lecture seule le fichier dont le nom est spécifié. Si aucun nom logique n'est donné, **ARCHIVE** prendra "**maître**".

```
Syntaxe : lis ndf [ logique nlf ]
```

## IMPRIME

Affiche les valeurs de la liste qui suit sur l'imprimante, de la même manière que **liste**.

```
Syntaxe : imprime [ exp | écr * [ ;exp | écr ] * ] [ ; ]
```

## UNIS

Ajoute aux procédures en mémoire celles d'un fichier spécifié. Si les procédures ont le même nom, seule la nouvelle sera conservée.

```
Syntaxe : unis [ objet ] ndf
Objet indique un fichier binaire.
```

## MODE

Change la forme de l'affichage.

```
Syntaxe : mode var , var
```

La première variable est 0 ou 1. Zéro regroupe les zones d'affichage, de contrôle et de travail en une seule. La valeur 1 les sépare de nouveau en trois. La seconde variable correspond au nombre de caractères par ligne, 40, 60 ou 80 que l'on choisit avec 4, 6 ou 8.

Initialement, ces valeurs sont : mode 1, 8

## RAZ

(Remise à zéro). Efface toutes les données de la mémoire de l'ordinateur pour un nouveau départ. Tous les fichiers encore ouverts sont fermés. **RAZ** n'efface pas les fichiers sur Microdrive.

Syntaxe : `raz`

## SUIV

Pointe sur l'enregistrement suivant du fichier courant, ou d'un fichier dont le nom logique est spécifié.

Syntaxe : `suv [ nlf ]`

## OUVRE

Ouvre le fichier dont le nom est spécifié à la lecture et à l'écriture. Le nom logique "**maître**" lui est donné si aucun n'est précisé.

Syntaxe : `ouvre ndf [ logique nlf ]`

## TRIE

Trie les enregistrements d'un fichier en fonction du contenu de champs spécifiés.

Syntaxe : `trie clé_tri * [ ,clé_tri ] *`  
ou `clé_tri:=var; c | d`

Le premier champ spécifié dans la liste est la clé de tri primaire. Les enregistrements égaux sur cette clé sont ensuite triés sur la clé suivante si elle existe, etc... Pour chaque clé, un sens de tri peut être donné, croissant ou décroissant, noté **c** ou **d**. Le tri ne prend en compte que les huit premiers caractères des chaînes et seules **4** clés de tri sont autorisées.

## PAPIER

Fixe une couleur du papier en fonction de l'expression.

Syntaxe : `papier n.exp`

Voir **encre**.

## POSITION

Fixe l'enregistrement dont le numéro suit comme enregistrement courant.

Syntaxe : `position n.exp`

## ECRIS

Affiche à l'écran les valeurs des paramètres dont la liste suit. Ils doivent être séparés par un point-virgule. Si la liste se termine par un point-virgule, il n'y aura pas de saut de ligne à la fin de l'instruction.

Syntaxe : `écris [ exp | écr ] * [ ;exp | écr ] * [ ; ]`

## QUITTE

Ferme les fichiers ouverts et retourne au **SUPER BASIC**.

Syntaxe : `quitte`

## NOTE

Marque un commentaire dans une procédure. Tout le texte qui suit est ignoré lors de l'exécution.

Syntaxe : `note`

## RESTAURE

Restaure tous les enregistrements d'un fichier extraits avec **isole**. L'ordre d'un tri antérieur est perdu.

Syntaxe : `restaure`

## REVIENS

Force un retour à la procédure d'appel.

Syntaxe : `reviens`

## EXEC

Charge en mémoire, à partir d'un Microdrive, un fichier de procédures et exécute la procédure **commence** (si elle existe).

Syntaxe : `exec [ objet ] ndf`

Voir **sauve**.

## SAUVE

Sauvegarde toutes les procédures en mémoire sur une cartouche de Microdrive.

Syntaxe : `sauve [ objet ] ndf`

L'option **objet** signale que la sauvegarde s'effectue en binaire plutôt qu'en ASCII. Elle est plus rapide, car ne demande pas de conversion. Vous pouvez **charger**, **unir** ou **exécuter** un programme avec cette option. Ces programmes ont comme extension de nom **\_pro** au lieu de **\_prg**. Cette sauvegarde protège votre programme contre tout examen ou modification. L'option **protège** n'autorise pas la sauvegarde, le listage et l'édition. L'union d'un programme avec un programme protégé n'ôte pas la protection. Et le seul moyen d'annuler celle-ci est **raz**.

La sauvegarde d'une version protégée d'un programme n'affecte en rien la copie encore en mémoire. Copie que vous pouvez éditer, lister, etc...

## ECRAN

Affiche la présentation d'écran chargée auparavant par **chargem**. N'a pas d'effet si aucun écran n'est en mémoire.

Syntaxe : `écran`

## CHERCHE

Cherche dans le fichier courant, depuis le début, un enregistrement dont l'un des champs vérifie l'expression.

Syntaxe : `cherche n.exp`

## EDITEM

Appelle l'éditeur d'écran pour vous permettre de faire la présentation d'un fichier. Voir chapitre 7.

## ISOLE

Cherche dans tout le fichier les enregistrements où l'expression qui suit est vraie. Le fichier résultant ne contient que ces enregistrements. Voir **restaure**.

Syntaxe : `isole n.exp`

## SAISISM

Attend l'entrée des variables dont la liste suit. Toutes les variables doivent être affichées à ce moment par un écran actif.

Syntaxe : `saisism var * [ ,var ] *`

## CHARGEM

Charge en mémoire un écran de présentation précédemment sauvegardée, et l'affiche avec le contenu des variables qui en font partie.

Syntaxe : `chargem ndf`

Les valeurs des variables sont automatiquement mises à jour après une procédure, quand le contrôle revient au clavier.

## NORMAL

Dirige tous les **imprime**, **liste** et **état** vers l'imprimante. Ce qui annule les effets de **via**.

Syntaxe : `normal`

## VIA

Dirige tous les **imprime**, **liste** et **état** vers le fichier spécifié - ou l'écran - au lieu de l'imprimante.

```
Syntaxe : via ndf [ exporte | état ]  
ou  
via écran
```

Si vous dirigez la sortie vers un fichier, elle passera à travers le gestionnaire d'imprimante courant pour que tous les codes de contrôle d'impression soient générés.

Si vous utilisez **exporte**, **ARCHIVE** s'assurera que le fichier ne contient que des codes ASCII imprimables, des retours chariot et des sauts de ligne. Le résultat peut être importé par **QUILL**.

L'option **état** permet d'éviter de passer par le gestionnaire d'imprimante pour que tous les caractères (de contrôle y compris) soient transmis.

Si vous ne précisez pas de suffixe, **ARCHIVE** emploiera **\_lis**, (**\_exp** ou **\_dmp** si vous avez utilisé **exporte** ou **état**).

La dernière forme : **via écran** fixe la sortie vers l'écran.

## MONTRE

Force dans une procédure, l'affichage des champs de l'enregistrement courant.

```
Syntaxe : montre
```

Un écran doit être actif, sinon **montre** n'a aucun effet.

## SAUVEM

Sauvegarde sur un fichier Microdrive spécifié, la zone d'affichage actuelle en tant que présentation d'écran.

```
Syntaxe : sauvem ndf
```

Le texte d'arrière-plan et les variables affichées sont sauvegardés avec leur position.

## STOPPE

Arrête l'exécution de toutes les procédures et rend la main au clavier.

```
Syntaxe : stoppe
```

## TRACE

Commute le mode trace.

```
Syntaxe : trace
```

Tapez **trace** pour passer en mode trace. A partir de ce moment, toutes les lignes exécutées s'affichent sur la ligne de commande. Pressez la barre d'espace pour arrêter le déroulement d'un programme, qui continuera dès que vous relâcherez.



## CHANGE

Remplace l'enregistrement courant du fichier spécifié (ou du fichier courant si aucun nom logique n'est donné) par un enregistrement contenant les valeurs présentes des variables de champ.

Syntaxe : `change [ nlf ]`

## ACTIVE

Fixe comme fichier courant celui dont le nom est donné.

Syntaxe : `active nlf`

## TANTQUE

Répète l'exécution des instructions comprises entre **tantque** et **ftantque** tant que la valeur de l'expression est différente de zéro.

Syntaxe : `tantque n.exp : ... : ftantque`

## FONCTIONS

Une fonction est une sorte de récipient qui convertit un ou plusieurs paramètres, appelés **arguments de la fonction**, en une valeur finale que l'on dit **retournée** par la fonction.

Les fonctions **d'ARCHIVE** ont de 1 à 3 arguments. L'argument d'une fonction est écrit entre parenthèses, après le nom de la fonction sans laisser d'espace. Si plusieurs arguments sont nécessaires, il faut les séparer par des virgules. Les fonctions, même si elles n'ont pas d'argument, doivent être suivies de parenthèses. Ce qui permet de reconnaître les fonctions des commandes, même si leur nom est identique.

Les fonctions suivantes sont disponibles :

### ABS (n.exp)

Retourne la valeur absolue de l'argument. (Rend positif)

### ATG (n.exp)

Retourne l'angle, en radians, dont la tangente est **n.exp**.

### CAR(n.exp)

Retourne le caractère ASCII dont le code est **n.exp**. Tout code inférieur à 32 n'est envoyé qu'à l'imprimante.

Exemple :

```
imprime car(0)+car(13)
```

envoie vers l'imprimante le caractère ASCII du retour chariot. C'est intéressant si votre imprimante demande des séquences de codes de contrôle pour fonctionner. Voir le manuel de l'imprimante.

Pour écrire un 'A' sur l'écran, vous pouvez faire :

```
écris car(65)
```

**CODE(c.exp)**

Retourne le code ASCII du premier caractère de la chaîne spécifiée.

**COS(n.exp)**

Retourne le cosinus de l'angle donné en radians.

**NOMBENR ( [ nlf ] )**

Retourne le nombre d'enregistrements du fichier courant.

**DATE(n.exp)**

Retourne la date du jour sous la forme d'une chaîne :

n.exp	chaîne de la date
0	"AAAA/MM/JJ"
1	"JJ/MM/AAAA"
2	"MM/JJ/AAAA"

Vous devez mettre l'horloge à jour comme décrit dans LE GUIDE DES MOTS-CLEFS DU SUPER BASIC.

**JOURS (c.exp)**

Retourne le nombre de jours écoulés depuis le premier Janvier 1583 et une date spécifiée sous la forme d'une chaîne "AAAA/MM/JJ". La conversion présume de l'usage du calendrier Grégorien (moderne). La formule n'est bonne que pour des dates postérieures à 1583.

**Déci(valeur,dml,nbcar)**

**valeur:=(n.exp)**

**dml:=(n.exp)**

**nbcar :=(n.exp)**

Convertit la valeur numérique donnée en une chaîne de caractères équivalente, avec **dml** décimales, justifié à droite dans un champ de **nbcar** caractères de longueur.

déci(1.23e1,3,10) retourne " 12.300 "

**DEG(n.exp)**

Convertit un angle de radians en degrés.

**FDF ( [ nlf ] )**

Retourne une valeur de 1 si vous avez tenté de lire après la fin du fichier courant, ou d'un fichier dont le nom logique est spécifié. Sinon elle retourne une valeur de zéro.

**ERRNUM( )**

Retourne le numéro de la dernière erreur. Une erreur 0 indique que tout s'est bien passé. Ce nombre est le même **qu'ARCFIIVE** affiche quand il détecte une erreur.

**EXP(n.exp)**

Retourne la valeur de **e** (approximativement 2.718) à la puissance **(n.exp)**. Une erreur est générée si n.exp est supérieur à +88 car la valeur dépassera la capacité de calcul **d'ARCHIVE**.

**NOMZ (n.exp [,nlf ] )**

Retourne le nom d'un champ spécifié de l'enregistrement courant d'un fichier donné (ou du fichier courant si aucun nom logique n'est fourni).  
 nomz(0) retourne le nom du premier champ.

**TYPEZ(n.exp [,nlf 1)**

Retourne le type d'un champ spécifié de l'enregistrement courant d'un fichier donné (ou du fichier courant si aucun nom logique n'est fourni).  
 typez(0) retourne le type du premier champ.

**VALEURZ(n.exp [ ,nlf )**

Retourne la valeur d'un champ spécifié de l'enregistrement courant d'un fichier donné (ou du fichier courant si aucun nom logique n'est fourni).  
 valeurz(0) retourne la valeur du premier champ.

**VU( )**

Retourne 1 si un enregistrement est trouvé après l'utilisation de **cherche** ou de **trouve**, sinon retourne 0.

**Géné(valeur,nbcar)**

**valeur:=n.exp**

**nbcar: =n.exp**

Convertit une valeur numérique donnée en une chaîne de caractères. Le texte est justifié à droite dans un champ de **nbcar** caractères de long.

Exemple :

géné(1.23e1,10) retourne "12.3"

**CLAVIER( )**

Attend une pression de touche et retourne le caractère correspondant à la touche.

**TOUCHE( )**

Retourne le caractère dont la touche a été pressée au moment de l'exécution de la fonction (qui n'attend pas cette pression). Une chaîne vide est retournée si aucune touche n'a été pressée.

**SOUSCHN(chaine,sous)**

**chaîne:=c.exp**

**sous:=c.exp**

Trouve la première occurrence de **sous** dans **chaîne** et retourne la position du premier caractère de **sous** dans **chaîne**. Si aucune occurrence n'est trouvée, zéro est renvoyé. Le succès dépend des cas :

souschn("Janvier","Jan")      retourne 1  
 souschn("Janvier","an")      retourne 2  
 souschn("Janvier","AN")      retourne 0

**ENT(n.exp)**

Retourne la partie entière de l'expression.

ent(3.7)      retourne 3

ent(-4,8)      retourne -4

**LONG(c.exp)**

Retourne le nombre de caractères du texte spécifié.

**LN(n.exp)**

Retourne le logarithme Néperien (base e) de l'expression qui doit être supérieure à zéro, sinon une erreur est générée.

**MINUS(c.exp)**

Convertit le texte donné en minuscules.

**Mémoire( )**

Retourne le nombre d'octets de mémoire libres.

**MOIS(n.exp)**

Retourne sous forme de texte le nom du mois.

`mois(3)` retourne "Mars"

Si n.exp est supérieure à 12, le reste de la division par 12 sera pris en compte.

`mois(13)` et `mois(1)` donneront la même réponse : "Janvier"

Dépend de la version du QL

**ROND(valeur,nbcar)**

**valeur:=n.exp**

**nbcar:=n.exp**

Convertit une valeur numérique donnée en entier sous forme d'une chaîne justifiée à droite dans un champ de **nbcar** de long.

Exemple :

`rond(1.23e1,10)` retourne " 12"

**NOMBZ( [ nif ] )**

Retourne le nombre de champ composant un enregistrement du fichier spécifié (ou du fichier courant si aucun nom logique n'est donné)

**PI( )**

Retourne la valeur de la constante mathématique **PI**.

**RD(n.exp)**

Retourne une valeur en degrés convertie en radians.

**NUMENR( [ nif ] )**

Retourne le numéro (zéro pour le premier) de l'enregistrement courant du fichier spécifié (ou du fichier courant si aucun nom logique n'est donné)

**REPRO(c.exp,n.exp)**

Retourne une chaîne contenant **n.exp** fois les caractères contenus dans **c.exp**.

écrit `repro(" ",5)` retourne "\*\*\*\*\*"

écrit `repre("abc",3)` retourne "abcabcabc"

**SGN(n.exp)**

Retourne +1, -1 ou 0 selon que **n.exp** est positive, négative ou nulle.

**SIN(n.exp)**

Retourne le sinus de l'angle donné en radians.

**RAC(n.exp)**

Retourne la racine carrée de l'expression qui doit être positive.

**CHN(n,type,nbcar)**

**n:=n.exp**  
**type:=n.exp**  
**nbcar:=n.exp**

Le second paramètre **type** indique dans quelle forme se fera la conversion.

0     décimale (virgule flottante)  
 1     exponentielle ou scientifique  
 2     entière  
 3     générale

Le troisième paramètre, **nbcar** indique le nombre de décimales que vous désirez après conversion. Il doit toujours être spécifié, même s'il est inutile comme dans la conversion en entier.

Exemple :

que a\$,chn(12.3456,0,2)     retourne "12.35"  
 que a\$=chn(12.3456,1,4)     retourne "1.23456e1"

**TG(n.exp)**

Retourne la tangente de l'angle en radians spécifié.

**HEURE( )**

Retourne sous forme d'une chaîne, l'heure du jour, au format de "HH:MM:SS". Vous devez avoir mis l'horloge à l'heure comme décrit dans LE GUIDE DES MOTS-CLEFS DU SUPER BASIC.

**MAJUS(c.exp)**

Convertir en majuscule la chaîne spécifiée.

**VAL(c.exp)**

Convertit la chaîne de caractères donnée en une valeur numérique. L'opération s'arrête dès qu'un caractère non numérique est rencontré.

val("1.1ABC")     retourne 1.1  
 val("ABC")     retourne 0.0

**VALVAR(c.exp)**

Retourne la valeur de la variable dont le nom est spécifié.

Exemple

que a\$="long"  
 que longueur=15  
 écris valvar(a\$+"ueur")     retourne 15

Notez que valvar(nomz(y)) égale valeurz(y)

## ERREURS

Quand **ARCHIVE** détecte une erreur dans une commande, tapée au clavier, ou dans une procédure, il affiche un numéro d'erreur et un court message.

Exemples d'erreurs qui peuvent être détectées :

- tenter une division par zéro
- **Si** sans **fsi**
- Appeler une procédure avec un mauvais nombre de paramètres

Si l'erreur survient après une entrée au clavier, le texte de l'instruction reste visible dans la zone de travail. **(F5)** permet d'éditer cette ligne erronée pour la corriger.

Si l'erreur survient pendant l'exécution d'un programme, **ARCHIVE** indique le nom de la procédure et la ligne de celle-ci où s'est produite l'erreur. Vous pouvez alors utiliser l'éditeur de ligne pour les corrections.

Quand vous utilisez la commande **erreur** dans vos programmes, **ARCHIVE** ne fera aucun compte rendu d'erreur. Vous êtes libre de les gérer comme bon vous semble (même de les ignorer). Vous pouvez consulter **errnum()** pour détecter quelle erreur s'est produite.

La liste qui va suivre montre les correspondances entre les numéros et les messages d'erreur d'**ARCHIVE**. Un court exemple est parfois ajouté comme explication. Les messages d'erreur ne sont pas faits pour vous indiquer quelle erreur est commise exactement, mais plutôt pour vous indiquer son type.

Les messages d'erreur pour lesquels il n'y a pas d'exemple sont marqués d'un astérisque, ils seront expliqués plus loin.

N°	Message	Exemple
0	pas d'erreur	
1	commande incompréhensible	écrit
2	fin d'instruction attendue	que x=3 que y=4
3	nom de variable attendu	que 31=x
4	ordre d'impression / affichage invalide	écris crée
5	type de donnée incorrect	* (1)
6	expression numérique attendue	que x= "jean"
7	chaîne littérale attendue	que x\$=4
8	variable introuvable	que x=q (q non défini)
9	variable indéfinie	écris q
10	séparateur manquant	écris au 5
11	nom trop long	que cenomesttrèslong =10
12	nom en double	crée:n\$:n\$:fcrée
13	chaîne littérale attendue	* (2)
14	proc sans fproc	* (3)
15	proc invalide	* (3)
16	instruction prématurément terminée	crée "test":fcrée
17	erreur de structuration	* (4)

18	trop de nombres	* (5)
50	guillemets de fermeture manquants	que x\$="jean
51	omission de l'exposant après "E"	que x=1.2E
52	nombre trop grand	que x=1.2E100
53	symbole inconnu	que x=%
70	erreur de syntaxe	que x=3+
71	parenthèse manquante	que x=(3+5)/7)
73	valeurs incompatibles	que x="jean"+3
74	nombre incorrect d'arguments	que x\$=chn(1,2)
75	chaîne trop longue	que x\$=repro("...",256)
76	division par zéro	que a=0:que x=5/a
77	arguments invalides	que x=rac(-4)
78	indice de chaîne erroné	que x\$="jean"(jusqu'à 97)
80	mémoire insuffisante	* (6)
90	pas de place pour ouvrir un fichier	* (7)
91	E/S fichier inachevée	* (8)
93	hors-limites	écris au 100,100;37
94	fichier non ouvert	ajoute (avec fichier fermé)
100	ouverture fichier impossible	lis "xxx" avec fichier non existant
101	fichier protégé contre l'écriture	lis "nom":insère
103	type incorrect de fichier	chargem"nom" (fichier de données)
104	nom de fichier invalide	sauve "3test"
105	erreur de lecture sur fichier	* (9)

1. La cause la plus courante de l'erreur 5 - type de donnée incorrect - est la saisie de texte quand un nombre est attendu. Exemple **saisis x**.
2. L'erreur 13 peut survenir pendant l'importation d'un fichier que vous n'avez pas exporté. En général quand **ARCHIVE** trouve un nombre où il attendait un texte, ou vice versa, ce qui crée les erreurs 7 et 8.
3. Les erreurs 14 et 15 ne doivent jamais survenir en utilisation normale. Elles ne peuvent arriver que si vous créez un programme avec un autre éditeur que celui d'**ARCHIVE**.
4. L'erreur 17 indique une erreur de structure. Elle montre que un **partout**, un **si** ou un **tantque** n'est pas apparié avec un **fpartout**, un **fsi** ou un **ftantque** dans une procédure. Cette erreur peut arriver si vous ajoutez un **fproc** dans une autre structure, ou en utilisant **reviens** au clavier.
5. L'erreur 18 indique que vous essayez d'entrer plus de nombres qu'il n'est possible dans la mémoire réservée pour la saisie. Cette erreur peut être créée du clavier, d'une procédure ou pendant le chargement d'une procédure avec beaucoup de nombres sur l'une de ses lignes - une limite est de 15 à 20, ce qui est déjà suffisant.
6. L'erreur 80 n'apparaîtra que si vous utilisez un très gros programme. La taille d'un fichier de données n'est pas limitée par la mémoire centrale puisqu'il peut être chargé par tronçons. Si **ARCHIVE** signale cette erreur, il faut diminuer la taille du programme avant de poursuivre. Si nécessaire, divisez votre programme en plusieurs parties que vous sauvegardez séparément. Utilisez **unis** pour charger chacune des parties quand il le faut. Cette technique demande une grande dextérité de programmation.

7. L'erreur 90 survient quand la zone qu'ARCHIVE réserve pour stocker des informations internes sur le fichier, est pleine. Ceci peut arriver même si la valeur retournée par **mémoire( )** n'est pas près de zéro.
8. L'erreur 91 signale que pour certaines raisons, les échanges d'informations se sont mal effectués. Cela peut venir des données ou de la cartouche.
9. L'erreur 105 signale que le fichier en lecture n'est pas au bon format, ou dans le bon ordre, ou qu'il est défectueux. Cette erreur ne doit pas arriver si vous utilisez l'éditeur de programme d'ARCHIVE plutôt qu'un autre.