dBASE™ II

Système de gestion de base
de données relationnelle
pour

AMSTRAD

CPC 6128 et PCW 8256

Manuel d'utilisation et disquette

La Commande Electronique

ASHTON·TATE

dBASE™ II

Système de gestion de base
de données relationnelle
pour

AMSTRAD

CPC 6128 et PCW 8256

Manuel d'utilisation et disquette

dBASE II

Pour

AMSTRAD

CPC 6128 et PCW 8256

MANUEL D'UTILISATION

Éditeur : La Commande Electronique
7, rue des Prias
27920 St-Pierre-de-Bailleul

Composition et Mise en page : Imprimerie Durand.
Imprimé en France par DM Impression
Dépôt légal, 4e trimestre 1985 ISBN 2-905350-05-9

SOMMAIRE

Première partie

GUIDE DE FORMATION

TABLE DES MATIERES

GUIDE DE FORMATION

CHAPITRE I : INTRODUCTION

CHAPITRE II : L'ESSENTIEL DE dBASE II

CHAPITRE III : INTRODUCTION A LA PROGRAMMATION

CHAPITRE IV : DEVELOPPEMENT DES TECHNIQUES dBASE II STANDARD

CHAPITRE I

INTRODUCTION

QU'EST-CE QUE dBASE II ?

La meilleure façon de comprendre dBASE II est de songer à ce que représente un traitement de texte pour des informations. Un traitement de texte vous permet de manipuler des caractères, des mots, des phrases et des pages pour créer un document qui correspondra à ce que vous attendez. dBASE II vous permet de travailler avec des rubriques, des enregistrements et des fichiers pour gérer vos informations de la façon que vous voulez.

Le traitement de texte est la cause de nombreuses installations de micro-ordinateurs dans les bureaux partout dans le monde. Pour beaucoup de gens, c'est un concept facile à comprendre et la nécessité d'améliorer considérablement la productivité dans les bureaux justifie l'achat d'un ordinateur.

A partir du moment où les gens seront familiarisés avec la sauvegarde des documents et le classement d'informations dans leur ordinateur, ils éprouveront bientôt le besoin de gérer leurs données d'une façon plus organisée. La démarche logique est de rechercher un moyen de faire du traitement de l'information.

Pour rendre cela aussi simple que possible, et en fournissant le maximum de possibilités, dBASE II possède en premier lieu un grand nombre de facilités pour la gestion des données. Parmi celles-ci, un langage d'interrogation interactif (ce qui veut dire que vous pouvez lui « parler »), un éditeur de texte pour créer des documents au format, et un langage de programmation très puissant permettant à une personne qui veut apprendre d'adapter dBASE II aux besoins de ceux qui sont moins familiarisés avec les ordinateurs. Beaucoup d'autres caractéristiques font de dBASE II l'instrument idéal pour résoudre des applications professionnelles.

LA RAISON DE CET OUVRAGE

Cet ouvrage a été écrit pour compléter le manuel qui accompagne dBASE II et non pour le remplacer. Il était souhaitable que la matière traitée dans cet ouvrage fournisse un choix supplémentaire de commandes et de techniques. Vous pourrez alors lire le manuel pour trouver les explications plus détaillées de toutes les commandes.

dBASE II est un vaste programme avec de nombreuses possibilités. Il est donc nécessaire que le manuel explique chaque commande avec toutes les variantes possibles. Ce livre n'a pas le même objectif, ainsi beaucoup de commandes ont été complètement laissées de côté. A certains moments, il vous sera demandé de vous y référer pour des détails particuliers sur les touches du clavier et la syntaxe. Ceci nous laisse beaucoup plus de place pour étudier les méthodes de conception et de réalisation d'une application dBASE II.

NIVEAU REQUIS DE CONNAISSANCE INFORMATIQUE ?

En un mot, la réponse est « aucun ». L'ouvrage vous fournira toutes les connaissances nécessaires. La démarche adoptée dans ce livre sera de commencer avec des sujets qui ne nécessitent aucune connaissance antérieure, et de construire progressivement des exemples plus complexes. Les lecteurs débutants plus avertis pourront atteindre à leur rythme les chapitres plus « pointus ».

Vous remarquerez que cette méthode donne satisfaction si les exemples sont suivis dans l'ordre. Si vous ne comprenez pas à la première lecture, la seconde vous paraîtra sans doute plus évidente.

Il se passe la même chose lorsque vous êtes débutant et que vous parcourez un magazine spécialisé en micro-informatique; vous ne comprenez rien. Au bout d'un certain temps, vous pouvez lire tous les articles.

QUI PEUT LIRE CE LIVRE ?

dBASE II est utilisé actuellement par des personnes de niveaux différents. A partir du moment où cet ouvrage construit des exemples sur des notions qui ont été vues précédemment, tout utilisateur de dBASE II aura intérêt à lire ce livre. La meilleure démarche est de commencer à la première page et de continuer jusqu'à ce que cela devienne trop technique. Les personnes familières avec dBASE II peuvent également lire les premiers chapitres.

CHAPITRE II

L'ESSENTIEL DE dBASE II

DIALOGUER AVEC dBASE

Il y a deux façons élémentaires de communiquer avec des ordinateurs : soit à travers des menus, soit par l'intermédiaire de langages de programmation. La plupart des programmes utilisés habituellement sont « dirigés par des menus », dans lesquels est affichée une liste d'options. Un exemple typique serait une comptabilité qui afficherait ce qui suit :

```
                    COMPTABILITE XYZ

1.  Comptabilité clients

2.  Comptabilité fournisseurs

3.  Grand livre

4.  Paie

 CHOISISSEZ UNE DE CES OPTIONS :
```

Si c'est le numéro du programme de paie qui est choisi, un second menu vous permettra de sélectionner des fonctions différentes dans la paie. Quelqu'un qui ne connaît pas le programme est assisté dans chacune des opérations.

Le premier inconvénient est le temps mis pour afficher les menus et attendre les réponses de l'utilisateur. Cinq à six menus peuvent être affichés avant que la fonction désirée soit atteinte. Il est également possible que l'on soit obligé de retourner au menu principal avant de commencer toute nouvelle opération. Les programmes dirigés par des menus sont utiles quand ils sont utilisés pour des utilisateurs ne connaissant pas le programme, mais avec un peu d'habitude, on peut souhaiter obtenir l'exécution d'une tâche particulière de façon plus directe.

dBASE II utilise le système de direction par « commandes », dans lequel une tâche est exécutée et décrite avec un langage spécial. Ces commandes ont des significations particulières et obligent de se conformer strictement à une syntaxe propre à dBASE II. Un utilisateur motivé peut rapidement exécuter n'importe quelle tâche sans être obligé de travailler à partir de menus.

Le problème est donc la motivation. Les personnes achètent un ordinateur et il pourrait, éventuellement, faire un travail à leur place. La plupart du temps, ces personnes ne sont pas préparées à passer du temps à apprendre comment se servir d'un ordinateur. Après tout, n'importe qui sait regarder la télé ou utiliser le téléphone sans aucune pratique. Alors pourquoi est-il nécessaire d'apprendre un nouveau langage pour être à même d'utiliser un ordinateur ? La réponse est qu'un ordinateur est une machine électronique destinée à toutes sortes d'usages et qui a un potentiel illimité, les nouveaux jeux de commandes doivent être acquis pour chaque application différente.

dBASE II vous fournit le langage pour convertir votre ordinateur en machine de base de données. Si on utilise par exemple une application de tenue de stock, une liste des articles qui coûtent plus de 1.000 Frs peut être obtenue avec les commandes suivantes :

```
. USE STOCK
. LIST ALL ARTICLES, FOURNIS FOR PRIX > 1000.00
```

Un programme supervisé par des menus ne pourrait pas donner ce résultat aussi facilement.

Si vous pensez qu'apprendre de nombreuses commandes vous prendra plus de temps que vous n'en disposez, n'ayez crainte. Seul un petit nombre de commandes essentielles exécuteront la plupart des tâches classiques. Si le système des menus vous paraît plus commode, soit. Un programme peut être écrit en langage dBASE II en utilisant des menus. Vous pourrez donc utiliser les deux approches. Essayons maintenant le système des commandes.

LA SYNTAXE DE dBASE II

Tous les langages ont un jeu de règles grammaticales pour la construction des phrases. On appelle cela la « syntaxe » du langage. La syntaxe des langages d'ordinateurs est habituellement rigide, la ponctuation et l'ordre des commandes sont compliqués. Souvent, le placement correct de points et de virgules est particulièrement long à apprendre. Le langage peut avoir des règles de syntaxe suffisamment concises pour que les utilisateurs puissent aisément les combiner pour créer de nouvelles commandes.

Construit pour des non-programmeurs, celui-ci simule généralement la syntaxe de l'anglais. Pour accélérer le processus d'apprentissage, le programme peut donner des explications lorsque l'utilisateur fait une faute de syntaxe. Les messages d'erreur seront alors aussi proches de la langue naturelle que possible. Si l'ordinateur répond à une faute de frappe par « SN ERROR », cela ne nous est pas d'un très grand secours.

La plupart des commandes dBASE II utilisent le format suivant :

VERBE LIMITATION NOM CONDITION

Les verbes décrivent l'action qui doit être exécutée, et sont habituellement significatifs.. Les verbes usuels sont : LIST, USE et DISPLAY. La limitation réduit le champ d'action de l'utilisation du verbe. Un verbe peut s'appliquer à l'ensemble d'un fichier ou aux cinq enregistrements suivants, dans chaque cas, la limitation pourrait être ALL pour tout le fichier, et NEXT 5 pour les cinq enregistrements suivants, respectivement. Le nom est l'objet sur lequel agit le verbe, cela peut être un fichier, une rubrique ou une variable. La condition sélectionne les noms qui remplissent cette condition.

Les questions de syntaxe peuvent être pénibles, aussi ne vous concentrez pas trop sur les règles de grammaire. La syntaxe de dBASE II n'est pas plus rigide que la plupart des langages, mais elle est très proche de l'anglais pour permettre de les apprendre à un rythme naturel. Si vous faites une erreur, dBASE II essaiera de commenter la faute et suggérera une correction.

Voici quelques exemples de commandes :

```
VERBE LIMITATION NOM CONDITION

DELETE ALL FOR PAIEMENT = 0

DISPLAY NEXT 10 NOM

LIST NOM, VILLE FOR VILLE = 'PARIS'
```

Comme vous pouvez le voir, il n'est pas toujours nécessaire d'inclure tous les éléments d'une phrase dBASE II. Il arrive que la limitation et le nom soient implicites. Une condition est ajoutée seulement si le verbe doit agir sur un groupe de noms particulier. N'essayez pas d'apprendre par cœur les commandes qui sont utilisées dans les exemples. Lorsque vous commencerez à travailler avec votre programme, les règles de syntaxe deviendront rapidement naturelles.

LES FICHIERS DISQUE

Un fichier disque est simplement une série de caractères que l'ordinateur écrit sur la surface magnétique d'une disquette ou d'un disque dur. Cela ressemble à l'enregistrement des sons sur un magnétophone à cassette. L'ordinateur peut stocker plusieurs milliers de noms et d'adresses et les retrouver avec une extrême rapidité.

La sauvegarde d'informations sur disque n'est pas simplement utilisée pour des noms et des adresses. Une série de commandes usuelles peuvent être sauvegardées sur un fichier disque, et utilisées à n'importe quel moment sans avoir à les retaper. Les limitations de capacité mémoire de l'ordinateur peuvent être repoussées par l'utilisation du disque comme un bloc-notes pour un stockage temporaire ou permanent de l'information.

UTILISONS DES EXEMPLES EXTRAITS DE SEQUENCES dBASE II

Nous allons nous amuser un peu, et essayer quelques-unes des commandes de base. Nous construirons une petite liste de mailing, et nous verrons ce que nous pouvons en faire. Il est donc nécessaire que vous tapiez ces exemples à l'ordinateur, puisque les commandes qui seront utilisées ici seront nécessaires pour les prochains exemples.

Lorsque dBASE II est prêt à recevoir une commande, il vous affiche un point, « . », au début de la ligne. Ce symbole signifie que le programme vous attend. Cette simplicité d'expression peut être un peu déconcertante au début.

Les exemples dans ce livre sont distingués du texte par des tirets. Dans tous les exemples, les mots en caractères gras doivent être entrés par l'utilisateur. Les mots en caractères classiques sont les réponses. Toutes les commandes ont été mises en majuscules pour plus de clarté, mais les commandes en caractères minuscules fonctionnent de la même façon. Les commandes dBASE II se trouvent en majuscules à la fois dans le texte et dans les exemples. Le symbole utilisé pour retour chariot ou la touche Return dans les exemples est <CR>.

Pour simplifier la frappe, seules les quatre premières lettres d'une commande sont nécessaires pour dBASE II. Pour plus de clarté, toutes les commandes utilisées dans les exemples seront tapées en entier.

UTILISATION DE PLUS D'UN LECTEUR DE DISQUE

dBASE II sous-entend que tous les fichiers se trouvent sur le même disque que lui-même. Les fichiers sur des lecteurs différents doivent être nommés avec la lettre du lecteur suivie par le nom du fichier. Par exemple, un fichier appelé MAIL sur le lecteur B: sera appelé B:MAIL. Si la plupart de vos fichiers sont sur un lecteur différent, un nouveau lecteur par défaut devra être désigné. Vous pourrez faire cette opération par la commande « SET DEFAULT TO B: ». dBASE II supposera alors que tous les fichiers sont sur le drive B:, à moins qu'un nouveau lecteur soit désigné.

CREER UN FICHIER DE DONNEES

Avant de sauvegarder l'information dans un fichier disque, vous devez fournir la description de vos données. Chaque rubrique, dans un nouveau fichier, doit comporter son nom, son type et sa longueur. L'ensemble des rubriques constitue un enregistrement, et son format est appelé la structure de fichier. Chaque enregistrement a le même format, même s'ils contiennent des informations différentes. Un fichier de données ressemble à un tableau d'informations, où chaque ligne est un enregistrement et chaque colonne est une rubrique.

Chaque rubrique doit avoir un nom de dix caractères au maximum, et il est suggéré d'utiliser un nom qui donne une idée de son contenu. Tous les enregistrements dans MAIL.DBF contiendront un mélange de chiffres et de lettres, ainsi ils ont été déclarés par la lettre « C », pour des caractères. Si vous devez effectuer des calculs arithmétiques sur une rubrique, vous devez déclarer cette rubrique sous le type « N », pour du numérique. Une rubrique logique est soit vraie ou fausse, elle sera définie comme de type « L ».

La taille d'une rubrique détermine la place prise sur le disque. Tous les enregistrements sont stockés avec leur taille maximum, et tous les espaces inutilisés sont remplis avec des blancs.

Réserver un espace trop important pour une rubrique gâchera considérablement la capacité disponible sur disque. D'un autre côté, laisser trop peu de place vous créera des problèmes lorsque vous aurez un nom particulièrement long à placer dans votre liste de clients. Ne vous tourmentez pas, la taille des rubriques peut être modifiée par la suite.

Chaque enregistrement peut contenir jusqu'à 32 rubriques. Chaque rubrique caractère peut avoir une longueur maximum de 254 caractères, et les rubriques numériques jusqu'à 10 positions. Les rubriques logiques sont toujours d'une longueur de 1 caractère. La taille totale de toutes les rubriques dans un enregistrement peut atteindre 1 000 caractères. Un maximum de 65 535 enregistrements peut être stocké dans n'importe quel fichier de données.

```
. CREATE

DONNEZ LE NOM DU FICHIER : MAIL

DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
 CHAMP    NOM,TYP,DIM,DECIMAL(S)
 001      PRENOM,C,10
 002      NOM,C,10
 003      ADRESSE,C,20
 004      VILLE,C,10
 005      DEPARTMT,C,2
 006      CODEPOST,C,5
 007      <CR> (touche Return ou Enter)

VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ? N
```

Un fichier nommé MAIL.DBF a maintenant été créé sur le disque. L'extension DBF est ajoutée automatiquement, ainsi MAIL peut être reconnu comme un fichier de données lorsque l'on lit le répertoire du disque. La place disponible a été attribuée pour le prénom, le nom, l'adresse, la ville, la région et le code postal.

UTILISER (USE) UN FICHIER DE DONNEES

De façon à garder un style naturel, vous sélectionnez un fichier de travail en utilisant la commande USE. Lorsque la commande USE est fournie, dBASE II stoppe toute utilisation d'un fichier de données précédent. Lorsque vous avez utilisé la commande USE, dBASE II supposera que toutes les commandes se réfèrent à ce fichier. Un peu plus tard, nous verrons comment utiliser plus d'un fichier à la fois.

```
. USE MAIL
```

Vous pouvez maintenant travailler avec le fichier appelé MAIL.DBF. En termes informatiques, cela s'appelle l'ouverture d'un fichier. Si vous êtes un programmeur BASIC, vous serez sans doute étonné de savoir que USE remplace toutes les instructions OPEN et FIELD du BASIC. Ce simple détail peut justifier de se mettre à dBASE II.

AJOUTER (APPEND) DES ENREGISTREMENTS A dBASE II

Maintenant que MAIL.DBF a été créé, l'information peut être ajoutée au fichier avec la commande APPEND. APPEND vous permet de déplacer le curseur sur n'importe quelle rubrique, et de saisir ou de modifier l'information.

Lorsque vous en avez fini avec cet enregistrement, simplement dépassez la dernière rubrique. dBASE II vous présentera alors un nouveau enregistrement. Pour dire à dBASE II que vous ne désirez pas saisir de noms supplémentaires, appuyez simplement sur Return sur le premier champ du nouvel enregistrement.

L'information sera alors sauvegardée sur le fichier disque qui a été sélectionné par USE.

```
. APPEND

RECORD # 00001
PRENOM      : FRANCIS
NOM         : PRADEL
ADRESSE     : 100 RUE FEDERALE
VILLE       : BELLEY
DEPARTMT    : 01
CODEPOST    : 01284

RECORD # 00002
PRENOM      : SAMUEL
NOM         : BEN
ADRESSE     : 30 OUEST RUE ALEMEDA
VILLE       : ST. LOUIS
DEPARTMT    : 97
CODEPOST    : 97821

RECORD # 00003
PRENOM      : MAXIME
NOM         : JUNIOR
ADRESSE     : 1 RUE DU CENTRE
VILLE       : CHARLY
DEPARTMT    : 02
CODEPOST    : 02174

RECORD # 00004
PRENOM      : <CR>
```

Lorsque l'on saisit l'information, vous devez trouver avantage à utiliser l'éditeur plein écran. En appuyant sur des touches spéciales, vous déplacez le curseur de rubrique en rubrique, ce qui permet de modifier l'information qui vient d'être saisie. L'appendice du manuel dBASE II vous donne la liste des touches à utiliser pour profiter de cet éditeur d'écran.

Si les touches curseur ne fonctionnent pas correctement, relisez la partie installation de votre manuel ou contactez votre distributeur.

LISTER (LIST) LES CONTENUS D'UN FICHIER DE DONNEES

Maintenant que nos noms sont sauvegardés sur le disque, ils peuvent être affichés sur l'écran avec la commande LIST.

```
. LIST

00001  FRANCIS   PRADEL      100 RUE FEDERALE       BELLEY      01 01284
00002  SAMUEL    BEN          30 OUEST RUE ALEMEDA  ST. LOUIS   97 97281
00003  MAXIME    JUNIOR        1 RUE DU CENTRE      CHARLY      02 02174

. LIST NOM

00001  PRADEL
00002  BEN
00003  JUNIOR

. LIST NOM,VILLE

00001  PRADEL     BELLEY
00002  BEN        ST. LOUIS
00003  JUNIOR     CHARLY

. LIST FOR VILLE = "BELLEY"

00001  FRANCIS    PRADEL    100 RUE FEDERALE      BELLEY     01 01284
```

Consulter un fichier de données est une chose facile. On peut lister sur l'écran tout un fichier ou uniquement certaines rubriques. Si nécessaire, vous pouvez lister uniquement les enregistrements qui obéissent à certains critères. Vous devez juste connaître le nom des rubriques.

Le nombre affiché avant la donnée est appelé le numéro d'enregistrement, il est affecté à chaque enregistrement lorsqu'il est saisi (APPEND). Ce nombre sera utilisé pour repérer sa position physique parmi les enregistrements du fichier de données.

UTILISER UNE IMPRIMANTE

Tout ce qui apparaît sur l'écran peut également être envoyé à une imprimante en tapant un controle-P. Il faut tenir enfoncée la touche marquée « CTRL », tandis que vous appuyez sur la touche « P ». Le premier controle-P dit à dBASE II de démarrer l'envoi de tout ce qui se trouve sur l'écran à l'imprimante, et le second l'éteint. Lorsque l'imprimante est active, toutes les commandes dBASE II demeurent inchangées.

TRIER (SORT) UN FICHIER DE DONNEES

Il est souvent utile, lorsque l'on a un fichier mailing, de vouloir imprimer les noms dans un ordre alphabétique, par le nom ou le code postal. Ceci peut être effectué par un tri (SORT) ou l'indexation (INDEX) du fichier.

On trie un fichier sur une rubrique particulière en déclarant son nom à la commande SORT, et le nom du nouveau fichier de données. Le fichier original sera lu et une version par ordre alphabétique sera sauvegardée sur le nouveau nom du fichier.

Le nom du nouveau fichier trié devra signifier son origine, mais n'importe quel nom ferait l'affaire.

```
. LIST NOM,CODEPOST

00001   PRADEL          01284
00002   BEN             97821
00003   JUNIOR          02174

. SORT ON CODEPOST TO MAILCP

*** TRI TERMINE ***

 . USE MAILCP

  . LIST NOM,CODEPOST

00001   PRADEL          01284
00002   JUNIOR          02174
00003   BEN             97821
```

Un nouveau fichier de données, appelé MAILCP.DBF, a été créé sur le disque. Comme vous pouvez le voir sur les numéros d'enregistrements, l'ordre physique des enregistrements se trouve dans l'ordre croissant des codes postaux. Le fichier original MAIL.DBF est gardé sur le disque dans son ordre de création. Une limitation évidente du tri est l'espace requis par le fichier nouvellement trié. Par conséquent, tout nouvel enregistrement ajouté à la fin d'un fichier trié ne sera plus dans un ordre trié.

L'inconvénient majeur du tri d'un fichier est le temps pris par cette opération. dBASE II est habituellement un programme très rapide d'exécution, mais le tri est la commande la plus lente du langage. On peut arriver jusqu'à des temps de plus d'une heure pour trier 1 500 enregistrements sur un disque souple.

INDEXER (INDEX) UN FICHIER DE DONNEES

Plutôt que trier un fichier, on peut créer un index à partir de n'importe quelle rubrique. La rubrique utilisée pour l'index est appelée la rubrique-clé. Un nouveau fichier est créé contenant un emplacement pour chaque enregistrement dans le fichier de données. Cet emplacement contient la clé et le numéro d'enregistrement correspondant. Cela ressemble un peu à l'index d'un livre, qui contient une liste des mots-clés et les pages sur lesquelles on peut les retrouver.

```
. USE MAIL

. INDEX ON NOM TO MAILNOM

00003 ENREGISTREMENTS INDEXES

. USE MAIL INDEX MAILNOM

. LIST NOM

00002  BEN
00003  JUNIOR
00001  PRADEL
```

Le fichier index de la base MAIL.DBF a été sauvegardé sur le disque avec le nom MAILNOM.NDX, et il est construit à partir de la rubrique appelée NOM. Les fichiers

index sont stockés sous une forme illisible, mais si on les traduisait, ils apparaîtraient comme suit :

```
BEN      2
JUNIOR   3
PRADEL   1
```

Si un fichier de données est utilisé avec un fichier index (USE INDEX), toutes les commandes dBASE II, telles que LIST, agiront comme si le fichier est dans l'ordre de l'index. Les numéros d'enregistrement montrent que les enregistrements sont encore actuellement dans l'ordre physique de création.

Puisque l'indexation d'un fichier ne nécessite pas la copie totale de celui-ci, on réalise donc une économie de place sur le disque comparée à la technique du tri (SORT). Si vous utilisez une base que vous indexez (USE INDEX), tout ajout d'informations dans le fichier mettra à jour automatiquement l'index.

D'autre part, l'indexation est à peu près trois ou quatre fois plus rapide que le tri. On peut créer plusieurs index sur un même fichier, et jusqu'à sept maximum peuvent être utilisés en même temps. Malheureusement, la gestion de nombreux index mis à jour diminuera la vitesse d'exécution de dBASE II. En pratique, trois fichiers d'index pourront être utilisés à la fois.

Un avantage supplémentaire de l'indexation d'un fichier est la possibilité d'utiliser la commande FIND. Les contenus des rubriques indexées sont fournis à dBASE II qui va chercher l'enregistrement correspondant. Si la recherche est infructueuse, un message d'erreur sera affiché sur l'écran. La commande FIND est extrêmement rapide et prend habituellement moins de deux secondes pour trouver n'importe quel enregistrement.

```
. FIND JUNIOR

. DISPLAY

00003  MAXIME     JUNIOR     1 RUE DU CENTRE        CHARLY   02 02174

. FIND FORD

ENREGISTREMENT NON TROUVE
```

Si dBASE II ne peut pas trouver (FIND) la clé recherchée, un message *** ENREGISTREMENT NON TROUVE *** sera affiché.

TRIER OU INDEXER, QUE CHOISIR ?

Il n'est pas prouvé que l'indexation soit plus profitable que le tri. Il y a des situations où il est préférable de trier un fichier (SORT). Lorsqu'il est nécessaire de consulter chaque enregistrement un par un, pour sortir des étiquettes sur imprimante, il est plus rapide d'utiliser un fichier qui a été trié suivant un ordre déterminé. Ceci vient du fait que tous les enregistrements sont physiquement dans le bon ordre, tandis que les fichiers indexés ont seulement l'apparence d'être en ordre.

dBASE II n'a que très peu d'opérations disque pour retrouver les enregistrements triés. L'exemple ci-dessous vous montre le type d'étude que seriez amené à faire.

Créez le fichier en ordre :	Indexation	5 minutes.
	Tri	20 minutes.
Temps supplémentaire pour trier : 15 minutes.		
Imprimer des étiquettes :	Trier le fichier	5 minutes.
	Indexer le fichier	10 minutes
Temps supplémentaire pour imprimer le fichier indexé : 5 minutes.		

Si ce jeu d'étiquettes est imprimé trois fois, à partir du même fichier, il est certainement plus rapide de trier le fichier. Lorsque tous les enregistrements d'un fichier doivent être traités dans l'ordre plusieurs fois, un fichier trié (SORT) est préférable à un fichier indexé (INDEX).

ECRIRE DES RAPPORTS

La commande LIST est un moyen d'examen du contenu du fichier de données. Lorsque l'on a besoin de sortir un listing de toutes les informations d'un fichier, de façon plus présentable, on utilise la commande REPORT. Cette commande permet la construction d'un rapport sous forme de tableau qui est alors affiché à l'écran ou envoyé à l'imprimante.

Construire un rapport (REPORT) ressemble à la création (CREATE) d'un fichier de données. Les contenus de chaque colonne doivent être décrits en fournissant la taille et le nom de la rubrique de données qui doit être affichée. Chaque colonne doit également comporter un en-tête qui sera imprimé en haut de chaque page. A côté de l'affichage des contenus d'une rubrique déterminée, une colonne peut comporter le résultat d'une opération arithmétique à partir d'une combinaison de rubrique. Ainsi, si prix est une rubrique de données, la taxe peut être imprimée à une colonne particulière comme étant égale à prix * 0.05.

Le rapport peut inclure des totaux de rubriques numériques et des interruptions pour les sous-totaux lorsque les contenus d'une rubrique particulière changent. C'est le cas lorsque l'on a des rapports financiers à éditer. Un titre peut être ajouté en haut du rapport, il sera imprimé au début de chaque page avec la date et le numéro de page.

```
. USE MAIL

. REPORT FORM MAILLIST
ENTREZ LES OPTIONS, M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR DE PAGE
 DESIREZ-VOUS UN EN-TETE (Y/N) ? Y
DONNEZ L'EN-TETE : LISTE MAILING
DESIREZ-VOUS UN DOUBLE INTERLIGNE (N/Y) ? N
DESIREZ-VOUS DES TOTAUX (N/Y) ? N
COL     COLONNE LONGUEUR, CONTENU
001     10,PRENOM
ENTREZ LE TITRE : PRENOM
002     15,NOM
ENTREZ LE TITRE : NOM
003     15,VILLE
ENTREZ LE TITRE : VILLE
004     <CR>
```

Les paramètres de ce rapport sont sauvegardés sur le disque sous le nom MAILLIST.FRM. Ils peuvent être imprimés à tout moment en répétant la commande REPORT. Si l'on précise une condition, le rapport imprimera uniquement les enregistrements remplissant la condition.

```
. REPORT FORM MAILLIST

     PAGE NO. 00001
     01/12/85
                                        LISTE MAILING

     PRENOM          NOM                    VILLE

     FRANCIS         PRADEL                 BELLEY
     SAMUEL          BEN                    ST. LOUIS
     MAXIME          JUNIOR                 CHARLY

. REPORT FORM MAILLIST FOR VILLE = 'BELLEY'

     PAGE NO. 00001
     01/12/85
                                        LISTE MAILING

     PRENOM          NOM                    VILLE

     FRANCIS         PRADEL                 BELLEY
```

Si l'on veut modifier le rapport, on peut utiliser n'importe quel système de traitement de texte pour modifier le fichier MAILLIST.FRM. Lorsqu'il est modifié, il suffit simplement de le sauvegarder sur le disque et dBASE II pourra de nouveau l'utiliser un peu plus tard.

MODIFICATIONS ET CORRECTIONS DES DONNEES (EDIT, REPLACE)

Une très grande part du processus d'entrée de données est l'isolement et la correction des erreurs. Lorsque de nouveaux enregistrements viennent d'être ajoutés (APPEND), ils doivent être listés sur l'imprimante et vérifiés. Lorsque l'on trouve une erreur, il y a plusieurs façons de la corriger. La plus simple est celle qui utilise la commande EDIT. Le numéro d'enregistrement doit suivre cette commande.

De la même façon que la commande APPEND, EDIT vous permet d'opérer en mode plein écran. Cela veut dire que les contenus d'enregistrements sont affichés entièrement, et le curseur peut être déplacé sur n'importe quelle rubrique. Nous allons maintenant modifier le deuxième enregistrement pour que Samuel soit changé en Suzy (EDIT). Nous

pouvons dire à dBASE II d'arrêter la modification (EDIT) en appuyant sur la touche controle-W.

```
. LIST

00001  FRANCIS    PRADEL    100 RUE FEDERALE        BELLEY     01 01284
00002  SAMUEL     BEN        30 OUEST ALEMEDA       ST. LOUIS  97 97821
00003  MAXIME     JUNIOR      1 RUE DU CENTRE       CHARLY     02 02174

. EDIT 2

ENREGISTREMENT 00002

PRENOM    : SUZY
NOM       : BEN
ADRESSE   : 30 OUEST RUE ALEMEDA
VILLE     : ST. LOUIS
DEPARTMT  : 97
CODEPOST  : 97821

.LIST

00001  FRANCIS     PRADEL   100 RUE FEDERALE        BELLEY     01 01284
00002  SUZY        BEN       30 OUEST RUE ALEMEDA   ST. LOUIS  97 97821
00003  MAXIME      JUNIOR     1 RUE DU CENTRE       CHARLY     02 02174
```

La commande EDIT vous permet donc de changer le nom et de mettre Suzy à la place, mais on peut se servir d'une commande similaire (REPLACE). Le numéro d'enregistrement est d'abord tapé, suivi par l'information qui devra être remplacée (REPLACE) dans le nom de la rubrique. Si la limitation est définie comme ALL, chaque enregistrement devra être modifié.

Bien entendu, dBASE II ne vous alerte pas si le changement est incohérent. Puisqu'il n'existe pas de commande permettant d'arrêter le remplacement, il est conseillé de faire une copie du fichier de données avant de faire des modifications importantes. Vous utiliserez dans ce cas la commande COPY.

```
. 2

. REPLACE PRENOM WITH "SAMUEL"

00001 REMPLACEMENT(S)

. DISPLAY

00002   SAMUEL    BEN      30 OUEST RUE ALEMEDA    ST. LOUIS  97 97821

. COPY TO TEST

. USE TEST

. REPLACE ALL PRENOM WITH "FRANCIS"

00003 REMPLACEMENT(S)

. LIST

00001  FRANCIS    PRADEL  100 RUE FEDERAL        BELLEY     01 01284
00002  FRANCIS    BEN      30 OUEST RUE ALEMEDA  ST. LOUIS  97 97821
00003  FRANCIS    JUNIOR     1 RUE DU CENTRE     CHARLY     02 02174
```

Vous venez de voir un exemple excessif de la commande REPLACE. On ne pourrait pas opérer de cette façon sur une véritable application de stock.

```
. USE STOCK

. REPLACE ALL PRIX WITH PRIX * 1.10
```

Le prix de tous les articles du stock hypothétique viennent d'être augmenté de 10 %. Ce sont des choses qui arrivent !

SUPPRESSION D'ENREGISTREMENTS D'UN FICHIER DE DONNEES (DELETE)

La plupart des programmes de base de données ont un processus de suppression d'enregistrement très complexe. Avec dBASE II, il n'y a pas de problème. Vous entrez simplement le numéro d'enregistrement avec lequel vous voulez travailler, suivi par la commande DELETE.

```
. USE MAIL

. LIST

00001  FRANCIS    PRADEL   100 RUE FEDERALE       BELLEY     01 01284
00002  SAMUEL     BEN       30 OUEST RUE ALEMEDA  ST. LOUIS  97 97821
00003  MAXIME     JUNIOR     1 RUE DU CENTRE      CHARLY     02 02174

. 1

. DELETE

00001  ENREGISTREMENT(S) EFFACE(S)

. DISPLAY

00001  *FRANCIS     PRADEL    100 RUE FEDERALE      BELLEY     MA 01284
```

L'enregistrement est toujours dans le fichier, il a été marqué d'un astérisque pour une suppression future. dBASE II a choisi la méthode sûre dans les suppressions d'enregistrements. Ils sont tout d'abord marqués avec la commande DELETE et ensuite supprimés avec la commande PACK. De cette façon, vous avez la chance de pouvoir changer d'avis et de rappeler les enregistrements par RECALL, avant de les supprimer (PACK). PACK est irréversible, aussi il sera peut-être nécessaire de faire une copie de l'ancien fichier avant d'utiliser la commande PACK. Vous aurez donc une copie de sécurité.

```
. RECALL

00001 ENREGISTREMENT(S) RESTITUE(S)

. DISPLAY

00001  FRANCIS     PRADEL       100 RUE FEDERALE     BELLEY    MA 01284

. COPY TO TEST2

. USE TEST2

. 1

. DELETE

00001 ENREGISTREMENT(S) EFFACE(S)

. PACK

COMPACTAGE TERMINE, 00002 ENREGISTREMENT(S) TRANSFERE(S)

. LIST

00001  SAMUEL     BEN      30 OUEST RUE ALEMEDA    ST. LOUIS  97 97821
00002  MAXIME     JUNIOR    1 RUE DU CENTRE        CHARLY     02 02174
```

Le nouveau fichier, TEST2.DBF, ne contient plus que deux enregistrements. Comme vous pouvez le voir, les numéros d'enregistrements ont été ajustés. Les anciennes données sont toujours dans le fichier MAIL.DBF, et nous continuerons à l'utiliser pour nos exemples.

UTILISATION DE LA MEMOIRE DE L'ORDINATEUR SOUS dBASE II

Parallèlement au stockage de l'information sur le disque, vous pouvez aussi utiliser la mémoire de l'ordinateur pour y ranger des données usuelles. Cela est très important lorsque vous commencerez à programmer. Soixante-quatre emplacements de stockage sont disponibles pour les variables mémoire, et ces variables peuvent contenir jusqu'à 254 caractères, un nombre à 10 positions, ou une valeur vraie ou fausse.

Il n'est pas nécessaire de déclarer à dBASE II la taille et le type d'une variable mémoire, puisqu'elle sera attribuée d'après son contenu. L'information est sauvegardée dans une variable mémoire avec la commande STORE. Toutes les variables mémoire peuvent être listées à tout moment avec la commande LIST MEMORY.

```
. STORE "FRED" TO NOM

FRED

. LIST MEMORY

NOM          (C)  FRED
** TOTAL **      01 VARIABLES USED  00004 BYTES USED

. STORE 100 TO MONTANT

100

. LIST MEMORY

NOM          (C)  FRED
MONTANT      (N)    100
** TOTAL **      02 VARIABLES USED  00010 BYTES USED
```

Le contenu d'une variable mémoire particulière peut être affiché avec la commande ?. On peut la traduire par « Qu'est-ce que » ou « Affiche ».

```
. ? NOM

FRED

. ? 10 * MONTANT

1000
```

Toutes les variables mémoire peuvent être sauvegardées dans un fichier disque avec la commande SAVE TO. Le résultat est la création d'un fichier avec l'extension MEM. Les variables sont rappelées dans la mémoire avec la commande RESTORE FROM. Si vous voulez supprimer une variable, vous utilisez simplement la commande RELEASE.

```
. LIST MEMORY

NOM         (C)  FRED
MONTANT     (N)    100
** TOTAL **      02 VARIABLES USED  00010 BYTES USED

. SAVE TO TEST

. RELEASE NOM

. LIST MEMORY
MONTANT     (N)    100
** TOTAL **      01 VARIABLES USED  00006 BYTES USED

. RELEASE ALL

. LIST MEMORY

** TOTAL **      00 VARIABLES USED  00000 BYTES USED

. RESTORE FROM TEST

. LIST MEMORY

NOM         (C)  FRED
MONTANT     (N)    100
** TOTAL **      02 VARIABLES USED  00010 BYTES USED
```

Le fichier créé dans les exemples précédents s'appelait TEST.MEM, il est fondamentalement différent de TEST.DBF créé précédemment.

POINTEUR D'ENREGISTREMENT

Lorsqu'on utilise un fichier de données, dBASE II garde toujours la trace de l'enregistrement courant sur lequel on travaille. La raison en est que le numéro d'enregistrement courant est sauvegardé sous le nom « # » dans la mémoire de l'ordinateur. Cette variable mémoire est identifiée comme le pointeur d'enregistrement.

Un pointeur est un nombre qui se rapporte à une adresse. Exactement comme le code postal se rapporte à l'adresse de votre maison, le pointeur d'enregistrement se rapporte à l'enregistrement courant.

Pour découvrir le numéro d'enregistrement courant, nous demandons la valeur du pointeur d'enregistrement.

Pour examiner l'enregistrement, nous utilisons DISPLAY. La première fois que nous utilisons un fichier (USE), le premier enregistrement est l'enregistrement courant. dBASE II l'appelle le haut du fichier (TOP). Le dernier enregistrement est appelé BOTTOM. Pour se déplacer vers l'enregistrement suivant, nous sautons (SKIP). Si nous ajoutons un numéro d'enregistrement, nous obtiendrons le nouvel enregistrement courant.

```
. USE MAIL

. GOTO TOP

. ? #

1

. DISPLAY

00001   FRANCIS     PRADEL       100 RUE FEDERALE      BELLEY     02 01284

. 1

. DISPLAY

00001   FRANCIS     PRADEL       100 RUE FEDERALE      BELLEY     02 01284
```

Si on utilise un index, le pointeur d'enregistrement travaille dans l'ordre de l'index.

Lorsque l'on utilise la commande FIND, dBASE II sauvegarde le numéro d'enregistrement trouvé dans le pointeur d'enregistrement. S'il ne trouve pas l'enregistrement, dBASE II sauvegarde la valeur 0 dans le pointeur d'enregistrement.

```
. USE MAIL INDEX MAILNOM
. FIND MAXIME
. ? #
3
. DISPLAY
00003  MAXIME     JUNIOR      1 RUE DU CENTRE        CHARLY  02 02174
. FIND FORD
*** ENREGISTREMENT NON TROUVE ***
. ? #
0
. GOTO TOP
. DISPLAY
00002  SAMUEL     BEN      30 OUEST RUE ALEMEDA     ST. LOUIS  97 97821
```

Le pointeur d'enregistrement doit toujours être pris en compte lors de l'exécution d'une fonction. Lorsque l'on a bien compris l'effet de la fonction sur le pointeur d'enregistrement, on obtient toute la puissance de la fonction.

VARIABLES SYSTEME

Le pointeur d'enregistrement est un exemple de variables système. Ces variables sont mises à jour automatiquement par dBASE II.

Deux autres variables système sont DATE() et EOF. La date est enregistrée en mémoire, en début de programme, lorsque dBASE II vous la demande. EOF surveille la fin d'un

fichier de données pour tester dans des programmes si le dernier enregistrement a été atteint. Lorsque le pointeur d'enregistrement est positionné au-delà du dernier enregistrement, EOF prend la valeur vraie (true).

```
. ? DATE()

01/12/85

. LIST

00002  SAMUEL     BEN        30 OUEST RUE ALEMEDA   ST. LOUIS  97 97821
00003  MAXIME     JUNIOR      1 RUE DU CENTRE       CHARLY     02 02174
00001  FRANCIS    PRADEL    100 RUE FEDERALE        BELLEY     01 01284

. GOTO TOP

. DISPLAY

00002  SAMUEL     BEN        30 OUEST RUE ALEMEDA   ST. LOUIS  97 97821

. ? EOF

.F.

. GOTO BOTTOM

. SKIP

*** ENREGISTREMENT: 00001

. ? EOF

.T.
```

On s'aperçoit que le butoir se fait sur le dernier enregistrement. On peut donner à la commande SKIP une valeur pour sauter plusieurs enregistrements.

Exemple : SKIP 4 pour sauter 4 enregistrements ou
SKIP — 2 pour revenir à 2 enregistrements plus haut.

UTILISER (USE) PLUS D'UN FICHIER DE DONNEES

dBASE II se réserve deux emplacements mémoire pour les fichiers de données. Jusqu'à maintenant, nous n'avons utilisé que le premier emplacement mémoire. Pour utiliser un autre fichier de données en même temps, vous devez taper la commande SELECT SECONDARY, ce qui aura pour conséquence l'ouverture de la deuxième zone de mémoire. La commande USE dans la zone SECONDARY n'affectera pas le fichier qui est utilisé dans la zone PRIMARY. SELECT PRIMARY revient à la première zone (PRIMARY).

```
. SELECT PRIMARY

. USE MAIL

. LIST

00001  FRANCIS   PRADEL   100 RUE FEDERALE        BELLEY     01 01284
00002  SAMUEL    BEN       30 OUEST RUE ALEMEDA   ST. LOUIS  97 97821
00003  MAXIME    JUNIOR     1 RUE DU CENTRE       CHARLY     02 02174

. SELECT SECONDARY

. USE MAILCP

. LIST NOM

00001  PRADEL
00002  JUNIOR
00003  BEN

. SELECT PRIMARY
```

Dans cet exemple, MAIL.DBF et MAILCP.DBF sont maintenant utilisés à tour de rôle.

LES MACROS

Il est souvent nécessaire de taper la même commande plusieurs fois. L'effort de frappe peut être réduit en sauvegardant la commande dans une variable mémoire (STORE). Cette variable peut alors être utilisée à la place d'une commande suffisamment longue.

Si un & est placé devant une variable, dBASE II se comporte comme si le contenu de la variable était tapé au clavier. L'utilisation d'une variable mémoire pour remplacer une commande entière est appelée une macro. Les programmeurs utilisent souvent les macros pour raccourcir l'écriture avec un langage.

```
. USE MAIL

. STORE "LIST" TO L

LIST

. &L

00001  FRANCIS   PRADEL   100 RUE FEDERALE        BELLEY     01 01284
00002  SAMUEL    BEN       30 OUEST RUE ALEMEDA   ST. LOUIS  97 97821
00003  MAXIME    JUNIOR     1 RUE DU CENTRE       CHARLY     02 02174
```

STRUCTURE INTERNE D'UN FICHIER DE DONNEES

Le nom, le type et la taille des rubriques constituent la structure du fichier. Cette information est stockée en tête de chaque fichier de données, dans un bloc de caractères appelé l'en-tête de fichier. dBASE II peut déterminer la structure de n'importe quel fichier de données en utilisation (USE) simplement par la lecture de cet en-tête. La commande LIST STRUCTURE affiche cette information.

```
. LIST STRUCTURE

STRUCTURE DU FICHIER            : MAIL.DBF
NOMBRE D'ENREGISTREMENTS        : 00003
DATE DE LA DERNIERE MISE A JOUR : 01/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM       TYP     DIM     DECIMALE(S)
001       PRENOM     C      010
002       NOM        C      010
003       ADRESSE    C      020
004       VILLE      C      010
005       DEPARTMT   C      002
006       CODEPOST   C      005
** TOTAL **              00058
```

A côté de l'information sur la structure du fichier, l'en-tête gère aussi le nombre d'enregistrements dans le fichier et la date de la dernière mise à jour (ajouts ou modifications) du fichier.

La structure de MAIL.DBF nous dit que chaque enregistrement prend 58 caractères sur le disque. Une fois que le nombre total d'enregistrements placés dans le fichier est connu, l'espace de stockage nécessaire peut être calculé. Il est égal au nombre d'enregistrements multiplié par la longueur de chaque enregistrement. Avec en plus 521 caractères pour l'en-tête.

En examinant la structure du fichier, n'importe qui peut connaître les éléments ou rubriques d'un fichier créé par d'autres. De nombreuses personnes ou programmes peuvent alors partager aisément les mêmes fichiers de données.

MODIFIER LA STRUCTURE (MODIFY STRUCTURE) D'UN FICHIER DE DONNEES

Dans toute application informatique, une chose est à prendre en compte, c'est le changement. Dans le milieu informatique, il est dit souvent que « toute application qui tourne est déjà périmée ».

Il y a plusieurs raisons à ces inévitables changements. Les utilisateurs n'ont que très rarement une bonne compréhension de leur manuel de référence; ce qui amène une difficulté dans l'implémentation des systèmes automatisés. D'autre part, l'introduction de tout système informatique provoquera le changement de point de vue des possibilités de l'informatique. On entend parfois, « s'il peut le faire, peut-il faire aussi... ? ». Ces petites choses supplémentaires peuvent rallonger considérablement un projet.

Pour être au point avec les nécessités de modification d'une base de données, il est nécessaire d'avoir une bonne compréhension des techniques impliquées dans le changement de structure d'une base de données. Nous allons étudier quelques-unes de ces techniques.

Toute méthode de modification de la structure d'un fichier de données implique la création d'un nouveau fichier de données avec la structure et les données souhaitées. Il faut toujours garder l'ancien fichier comme sauvegarde pour garantir la sécurité des informations.

Si la commande CREATE est utilisée pour définir le nouveau fichier de données, on peut utiliser une variante de la commande APPEND déjà vue précédemment. Le nouveau fichier de données est appelé (USE), et la donnée est ajoutée à partir de l'ancien fichier (APPEND FROM).

dBASE II lit les en-têtes de chaque fichier et examine les rubriques en contrôlant leur nom dans chaque fichier. Le transfert s'opère ensuite à partir des différentes rubriques, enregistrement par enregistrement.

Nous allons changer la structure de MAIL.DBF en utilisant cette méthode, en supprimant la rubrique ville, en ajoutant une rubrique numéro de téléphone et en raccourcissant la longueur de la rubrique nom.

```
. CREATE
DONNEZ LE NOM DU FICHIER : NEWMAIL
DONNEZ LA STRUCTURE DES ENREGISTREMENTS SELON LE FORMAT :
CHAMP   NOM,TYP,DIM,DECIMALE(S)
001     PRENOM,C,10
002     NOM,C,4
003     ADRESSE,C,20
004     DEPARTMT,C,2
005     CODEPOST,C,5
006     TEL,C,13
007     <CR>
DESIREZ-VOUS COMMENCER LA SAISIE (Y/N) ? N

. USE NEWMAIL

. APPEND FROM MAIL

00003 ENREGISTREMENT(S) AJOUTE(S)

. LIST

00001    FRANCIS    PRAD 100 RUE FEDERALE        01 01284
00002    SAMUEL     BEN   30 OUEST RUE ALEMEDA   97 97821
00003    MAXIME     JUNI   1 RUE DU CENTRE       02 02174
```

Le nouveau fichier NEWMAIL.DBF a maintenant la structure désirée, et les données ont été transférées automatiquement dans leurs rubriques respectives. Les rubriques de l'ancien fichier qui manquent sont ignorées, les rubriques correspondantes sont chargées avec les informations qu'elles peuvent contenir, et les nouvelles rubriques sont laissées vides. La nouvelle rubrique vide, TEL, pourra être saisie manuellement avec les commandes EDIT ou REPLACE.

La méthode qui nécessite le moins de frappe est l'utilisation de la commande COPY. Si l'information d'un fichier existant est nécessaire, l'ancien fichier est utilisé (USE) et les rubriques sélectionnées sont copiées vers le nouveau fichier.

```
. USE MAIL

. COPY FIELD PRENOM,NOM,DEPARTMT TO NEWMAIL2

00003 ENREGISTREMENT(S) COPIE(S)

. USE NEWMAIL2

. LIST STRUCTURE

STRUCTURE DU FICHIER            : NEWMAIL2.DBF
NOMBRE D'ENREGISTREMENT         : 00003
DATE DE LA DERNIERE MISE A JOUR : 01/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM        TYP   DIM     DECIMALE(S)
0001      PRENOM      C    010
0002      NOM         C    010
0003      DEPARTMT    C    002
** TOTAL **             00023

. LIST

00001    FRANCIS         PRADEL            01
00002    SAMUEL          BEN               97
00003    MAXIME          JUNIOR            02
```

S'il est nécessaire de faire des changements plus conséquents, on peut copier uniquement la structure de l'ancien fichier dans le nouveau fichier (COPY STRUCTURE). On utilise alors ce nouveau fichier (USE), et on modifie sa structure. MODIFY STRUCTURE détruit toute information dans le fichier, c'est la raison pour laquelle on doit créer d'abord le nouveau fichier. Après les modifications de structure, les données sont ajoutées à partir de l'ancien fichier par APPEND FROM.

Puisque la commande MODIFY STRUCTURE implique le mode d'édition plein écran de l'en-tête du fichier, nous observons simplement le résultat dans l'exemple ci-dessous. Dans l'exemple qui suit, nous allons modifier la structure du fichier NEWMAIL3 pour étendre la rubrique NOM, supprimer la rubrique ADRESSE et inverser le code postal et le département.

```
. USE MAIL

. COPY STRUCTURE TO NEWMAIL3

. USE NEWMAIL3

. LIST STRUCTURE

STRUCTURE DU FICHIER            : NEWMAIL3.DBF
NOMBRE D'ENREGISTREMENTS        : 00000
DATE DE LA DERNIERE MISE A JOUR : 01/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM         TYP   DIM     DECIMALE(S)
001       PRENOM       C    010
002       NOM          C    010
003       ADRESSE      C    020
004       VILLE        C    010
005       DEPARTMT     C    002
006       CODEPOST     C    005
** TOTAL **              00058

. MODIFY STRUCTURE

LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES... (Y/N) ? Y

. LIST STRUCTURE

STRUCTURE DU FICHIER            : NEWMAIL3.DBF
NOMBRE D'ENREGISTREMENTS        : 00000
DATE DE LA DERNIERE MISE A JOUR : 01/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM         TYP   DIM     DECIMALE(S)
001       PRENOM       C    010
002       NOM          C    020
003       VILLE        C    010
004       CODEPOST     C    005
005       DEPARTMT     C    002
** TOTAL **              00048
```

```
. APPEND FROM MAIL

00003 ENREGISTREMENT(S) AJOUTE(S)

. LIST
00001    FRANCIS      100 RUE FEDERALE          BELLEY      01284 01
00002    SAMUEL        30 OUEST RUE ALEMEDA     ST. LOUIS   97821 97
00003    MAXIME         1 RUE DU CENTRE         CHARLY      02174 02
```

Les noms des rubriques existantes ont été conservés. De cette façon, dBASE II est capable de dire quelles sont les rubriques qui ont été transférées. Une procédure plus complexe est nécessaire pour modifier les noms des rubriques.

L'ancien fichier de données doit être copié sur le disque (COPY) sans sa structure. Quand les données sont ajoutées au nouveau fichier, ce sont les positions relatives plutôt que les noms de rubriques qui sont utilisées pour placer les données dans les nouvelles rubriques. On utilise ici l'option SDF pour la copie.

```
. USE MAIL
. COPY TO MAILTEMP SDF

00003 ENREGISTREMENT(S) COPIE(S)
```

Un fichier nommé MAILTEMP.TXT est maintenant créé sur le disque. La commande SDF indique que les données ont été sauvegardées sans la structure. Il se présente comme suit, mais ce n'est pas listable

```
FRANCIS   PRADEL    100 RUE FEDERALE          BELLEY      0101284
SAMUEL    BEN        30 OUEST RUE ALEMEDA     ST. LOUIS   9797821
MAXIME    JUNIOR      1 RUE DU CENTRE         CHARLY      0202174
```

```
. COPY STRUCTURE TO NEWMAIL4

. USE NEWMAIL4

. MODIFY STRUCTURE

LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES (Y/N) ? Y
```

```
. LIST STRUCTURE

STRUCTURE DU FICHIER            : NEWMAIL4.DBF
NOMBRE D'ENREGISTREMENTS        : 00000
DATE DE LA DERNIERE MISE A JOUR : 01/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM         TYP   DIM     DECIMALE(S)
001       PRENOM       C    010
002       NOM          C    010
003       ADRESSE      C    020
004       VILLE        C    010
005       DEPARTMT     C    002
006       CODEPOST     C    005
** TOTAL **              00058

. APPEND FROM MAILTEMP SDF

00003 ENREGISTREMENT(S) TRANSFERE(S)

. LIST

00001    FRANCIS   PRADEL   100 RUE FEDERALE       BELLEY      01 01284
00002    SAMUEL    BEN        30 OUEST RUE ALEMEDA ST. LOUIS   97 97821
00003    MAXIME    JUNIOR      1 RUE DU CENTRE     CHARLY      02 02174
```

On peut également modifier les choses en créant un fichier ayant la même structure, mais contenant seulement des enregistrements particuliers. C'est encore plus simple, il suffit de déclarer la condition dans la commande APPEND FROM ou COPY TO.

```
. USE MAIL

. COPY STRUCTURE TO NEWMAIL5

. USE NEWMAIL5

. APPEND FROM MAIL FOR VILLE = 'BELLEY'

00001 ENREGISTREMENT(S) TRANSFERE(S)
```

```
. LIST

00001    FRANCIS   PRADEL    100 RUE FEDERALE    BELLEY      02 02184

. USE MAIL

. COPY TO NEWMAIL6 FOR VILLE = 'BELLEY'

00001 ENREGISTREMENT(S) TRANSFERE(S)

. USE NEWMAIL6

. LIST

00001    FRANCIS     PRADEL    100 RUE FEDERALE    BELLEY      01 02184
```

Toutes ces différentes techniques ont leur utilité, et leur choix est une affaire de goût personnel. Ce qu'il faut retenir, c'est la manière de déplacer les informations d'un fichier à l'autre.

SORTIR D'UNE SEQUENCE dBASE II

Maintenant que nous avons fini notre exemple de séquences dBASE II, il est logique d'arrêter le programme. **Ne terminez jamais** une séquence en appuyant sur le bouton de RESET, ou bien en éteignant l'ordinateur. Cela aurait pour résultat la perte des informations nouvellement ajoutées, ou bien des dégâts sur les fichiers de données existants. La seule façon correcte de sortir est de dire QUIT.

```
. QUIT

***  FIN DU TRAITEMENT dBASE II  Version L.C.E.  ***
```

CONCLUSION

Maintenant que vous venez de voir l'extraordinaire potentiel qui s'ouvre à vous, je vous souhaite la bienvenue dans le monde fantastique de dBASE II. Avec un peu de pratique, vous serez à même de créer, ajouter et modifier aussi bien qu'un professionnel.

CHAPITRE III

INTRODUCTION A LA PROGRAMMATION

POURQUOI ECRIRE DES PROGRAMMES dBASE II ?

Chaque fois que nous voulons exécuter une série de fonctions dBASE II, nous devons taper chaque commande séparément au clavier. Mais lorsque l'habitude est prise, les utilisateurs sont obligés de répéter fréquemment les jeux de commandes. Lorsque c'est le cas, chacun veut pouvoir entrer simplement une commande brève afin que dBASE II répète le processus complet à notre place.

D'une certaine façon, la nouvelle commande serait ajoutée au langage dBASE II. dBASE II s'occupe déjà de tout le travail de gestion des fichiers disque, lorsque vous dites USE. Ne serait-il pas souhaitable de taper le mot étiquettes si vous aviez une suite d'étiquettes à imprimer ? C'est justement l'objet de ce chapitre.

Les commandes que nous venons d'utiliser avec dBASE II fonctionnent de la même manière dans un programme. Ceci va donc faciliter la mise au point d'une idée que vous avez tapée d'une façon interactive avant d'écrire le programme, et donc accélérer considérablement son développement.

QUI PEUT APPRENDRE A ECRIRE UN PROGRAMME EN dBASE II ?

Les personnes qui ont des niveaux de connaissances informatiques variées utilisent maintenant dBASE II. Quelques-uns programment déjà depuis 15 ans en COBOL, tandis que d'autres débutent dans la programmation avec l'achat de dBASE II. D'autres encore n'auront jamais besoin d'écrire aucun programme mais bénéficieront d'une vue d'ensemble du langage dBASE II.

Alors que les gestionnaires peuvent être des experts sur les besoins d'une application, un manque de connaissance sur dBASE II peut leur créer des difficultés pour expliquer les points de détails à un programmeur. Après avoir lu ce chapitre, vous serez plus à même de comprendre les possibilités et les limites de n'importe quel système écrit en dBASE II. Cette compréhension enrichira le débat entre gestionnaires et programmeurs lors de la réalisation et de l'installation d'un système informatique.

Le débutant en programmation trouvera un certain plaisir à dBASE II s'il peut y consacrer un peu de temps pour l'étudier. Ce chapitre ne fait pas référence à une expérience de la programmation. En travaillant à partir d'exemples, de difficultés progressives et avec un peu de pratique, chaque lecteur pourra travailler à des programmes plus complexes. Lorsque cela vous semblera trop ardu, il pourra être profitable de revenir en arrière et réessayer quelques-uns des nombreux exemples de base. Au bout d'un certain temps, les solutions vous paraîtront naturelles et compréhensibles.

Les programmeurs plus avancés pourront lire ce chapitre introductif, puisque les chapitres suivants sont construits à partir de ces exemples. Même le programmeur dBASE II le plus averti y trouvera de nouvelles idées qui pourraient se révéler intéressantes par la suite.

Des parties de programmes standard ont été introduites pour fournir au programmeur dBASE II un choix de solides utilitaires. Ces structures de base seront utilisées fréquemment dans tous les exemples de programmation. Ces parties ou modules peuvent être utilisées plus tard par le programmeur pour construire de nouvelles applications. Cette approche modulaire simplifie la programmation et la mise au point.

CREATION D'UN PROGRAMME dBASE II

Un programme dBASE II revient à frapper simplement une série de commandes dans un fichier texte, de la même façon que vous pourriez écrire un texte avec un logiciel de traitement de texte. dBASE II possède un éditeur intégré pour la création de programmes. Nos exemples ont été réalisés de cette façon.

L'éditeur fourni par dBASE II n'est pas très complet et on peut lui adjoindre un traitement de texte plus performant.

Si on utilise un programme de traitement de texte, on doit suivre un certain nombre de règles. Le nom de fichier doit avoir l'extension CMD, de telle façon que dBASE II sache

que c'est un fichier programme. Le programme de traitement de texte doit également savoir que le fichier ne sera pas utilisé comme un document texte mais comme un programme source.

Cet ouvrage ne fournit pas de détails sur l'éditeur de dBASE II, comme c'est le cas dans le manuel dBASE II. Pour commencer à éditer un fichier CMD, on utilise l'instruction MODIFY COMMAND nom-de-fichier. L'extension CMD n'est pas nécessaire, elle sera ajoutée automatiquement. Si le fichier nommé n'existe pas, l'éditeur le créera. Dans le cas contraire, le fichier existant sera édité à l'écran. Vous trouverez les programmes d'exemples dans le texte après l'instruction MODIFY. Les listings des programmes d'exemples seront frappés en caractères gras. Lorsque vous avez fini d'entrer le texte du programme, appuyez sur contrôle-W, dBASE II sauvegardera alors le programme sur disque. Pour faire tourner le programme, il suffit de dire à dBASE II de le faire (DO).

Notre premier programme s'appelle EXEMPLE.CMD et il exécute quelques-unes des fonctions primaires de dBASE II.

```
. MODIFY COMMAND EXEMPLE

* EXEMPLE.CMD
USE MAIL
LIST
LIST NOM,VILLE
REPORT FORM MAILLIST FOR VILLE = "BELLEY"
```

La seule chose nouvelle est l'utilisation d'un commentaire dans la première ligne de programme. Cette ligne commence par un astérisque, ce qui veut dire que toute instruction sur le reste de la ligne est ignorée par dBASE II. L'utilisation de commentaires dans un programme est une chose nécessaire pour sa documentation, et également une habitude à acquérir qui a son importance. La documentation aide le lecteur du programme et plus tard le programmeur. Une pratique courante est d'inclure la date de la dernière modification, le nom des auteurs, ce qui permet de blâmer quelqu'un lorsque le programme ne marche pas.

Ce programme est juste un jeu de commandes donnant le même résultat que si ces commandes étaient tapées individuellement. En les plaçant dans un fichier programme, nous pouvons répéter la série de fonctions simplement en tapant DO EXEMPLE. Pour raccourcir un peu l'exemple, nous effacerons d'abord (DELETE) le troisième enregistrement du fichier.

```
. USE MAIL

. 3

. DELETE

00001 ENREGISTREMENT(S) EFFACE(S)

. PACK

COMPACTAGE TERMINE, 00002 ENREGISTREMENT(S) TRANSFERE(S)

. DO EXEMPLE

00001  FRANCIS    PRADEL    100 RUE FEDERALE         BELLEY    01284
00002  SAMUEL     BEN       30 OUEST RUE ALEMEDA     ST.LOUIS  97821
00001  PRADEL     BELLEY
00002  SAMUEL     ST.LOUIS

       PAGE No 00001
  12/02/85

                              LISTE MAILING

       PRENOM     NOM       VILLE

       FRANCIS    PRADEL    BELLEY
```

Qu'est-ce que cela nous apporte de plus ? Nous aurions aussi bien pu taper cela directement et aussi rapidement dans dBASE II. Prenez patience. Nous allons réaliser des programmes beaucoup plus utiles.

COMMUNIQUER A PARTIR D'UN PROGRAMME

Nous désirons souvent que nos programmes puissent poser des questions, et y apporter des réponses appropriées. C'est ce que les programmeurs appellent les entrées-sorties « E/S ». Il y a toutes sortes de manières de gérer les entrées-sorties à partir d'un programme dBASE II, mais pour l'instant nous n'utiliserons que les commandes ? et ACCEPT.

La commande ? a déjà été évoquée, et peut être traduite par : « quelle est la valeur de ». On l'utilise pour afficher le contenu d'une variable, ou d'un message entre des guillemets. Plusieurs variables ou messages peuvent être affichés en les séparant chacune par une virgule.

```
. ? "BONJOUR"

BONJOUR

. ? "BONJOUR", "ERIC"

BONJOUR  ERIC

. STORE "GILLES" TO PRENOM

 . ? PRENOM

GILLES

. ? "BONJOUR", PRENOM

BONJOUR GILLES
```

La commande ACCEPT est identique à STORE, ce qui veut dire qu'elle place toute information entrée au clavier dans une variable mémoire. On peut également ajouter un message à l'écran lors de la question. Les deux points avant l'entrée de données sont fournis par dBASE II.

```
. ACCEPT TO PRENOM

: ERIC

. ? PRENOM

ERIC

. ACCEPT "QUEL EST VOTRE NOM ?" TO NOM

QUEL EST VOTRE NOM ? : MARTIN
```

Nous allons utiliser maintenant l'éditeur pour créer un programme appelé EXEMPLE2 de façon à essayer toutes ces commandes.

```
. MODIFY COMMAND EXEMPLE2

* EXEMPLE2.CMD
 ACCEPT "QUEL EST VOTRE NOM" TO NOM
? NOM, "C'EST UN JOLI NOM !"
                                (tapez CTRL W)
. DO EXEMPLE2

QUEL EST VOTRE NOM : DORIS
DORIS  C'EST UN JOLI NOM !
```

Vous pourriez maintenant écrire votre propre programme, qui permette l'interaction entre une personne et un ordinateur.

ECRIRE UN PROGRAMME UTILE

Il nous faudra apprendre beaucoup plus de commandes pour pouvoir commencer à écrire des programmes complets. Plutôt que de continuer avec des explications détaillées de chaque commande, nous allons commencer à écrire un programme d'impression d'étiquettes utilitaire. Comme de nouvelles commandes sont nécessaires, elles seront introduites dans le contexte du programme d'étiquettes.

D'ABORD ELABORER UN PROJET

Avant d'écrire un programme informatique, on doit analyser les exigences de l'application. Avant même d'allumer l'ordinateur, vous pourriez écrire votre projet en « pseudo-code », un langage très proche de la langue naturelle. Le programme serait alors traduit dans un langage que peut comprendre l'ordinateur.

Nous numéroterons chaque ligne de ce que nous appellerons le pseudo-code pour pouvoir s'y référer dans le chapitre suivant.

Projet de programme d'impression des étiquettes :

1. Utiliser le fichier de la liste mailing.
2. Lire le premier enregistrement.
3. Imprimer les contenus de l'enregistrement dans le format suivant.
   PRENOM NOM
   ADRESSE
   CODE POSTAL, VILLE
4. Aller à l'enregistrement suivant.
5. Continuer le processus jusqu'à ce que tous les noms et adresses soient affichés.

VERSION POUR UNE ETIQUETTE

Les trois premières lignes sont faciles à traduire. Ecrivons la première ébauche de ETIQUETTE.CMD et utilisons ces simples lignes. Nous commencerons également à utiliser des commentaires qui décriront l'objectif du programme.

```
. MODIFY COMMAND ETIQUET

*  ETIQUET.CMD   VERSION 1
*  CE PROGRAMME AFFICHERA DES ETIQUETTES DE MAILING
*  A PARTIR DU FICHIER MAIL.DBF
*

USE MAIL
? PRENOM, NOM
? ADRESSE
? CODEPOST, VILLE
                    (tapez CTRL W)
. DO ETIQUET

FRANCIS     PRADEL
100 RUE FEDERALE
01284 BELLEY
```

Pour un premier jet, ce programme ne fonctionne pas si mal. Le seul problème est l'espace qui se trouve entre le prénom et le nom.

Cet espace en trop est dû au fait que dBASE II ajoute des blancs qui complètent la rubrique. Quand on lui dit d'afficher la rubrique, il affiche également des blancs. Les espaces en trop peuvent être supprimés, en utilisant la fonction TRIM.

Modifions notre programme et refaisons-le tourner.

```
. MODIFY COMMAND ETIQUET

*  ETIQUET.CMD   VERSION 2
*  CE PROGRAMME AFFICHERA DES ETIQUETTES DE MAILING
*  A PARTIR DU FICHIER MAIL.DBF
*
USE MAIL
? TRIM(PRENOM), NOM
? ADRESSE
? CODEPOST, VILLE


. DO ETIQUET

FRANCIS PRADEL
100 RUE FEDERALE
01284 BELLEY
```

Le résultat est meilleur.

Les instructions des lignes 4 et 5 de notre projet viendront plus tard. La commande SKIP va permettre de se déplacer vers l'enregistrement suivant, mais nous n'avons pas encore trouvé le moyen de faire répéter la commande d'affichage des étiquettes jusqu'à la limite de la fin du fichier.

LA BOUCLE « FAIRE TANT QUE » (DO WHILE)

Nous avons besoin d'une boucle. C'est une technique de programmation pour répéter une série de commandes plusieurs fois. dBASE II va nous permettre de créer une boucle DO WHILE pour ce type de processus. Pour que notre programme puisse afficher les étiquettes pour tous les enregistrements de notre fichier MAIL, il doit s'occuper de deux choses. Il doit répéter la commande d'affichage d'une étiquette après avoir sauté à l'enregistrement suivant (SKIP), et s'arrêter lorsque le dernier enregistrement a été affiché.

Nous utiliserons la variable système EOF pour indiquer à notre programme le moment où il doit en sortir. Lorsqu'on demande à dBASE II de sauter (SKIP) après le dernier enregistrement, la boucle DO WHILE s'arrête parce que nous venons d'atteindre la fin du fichier (EOF = end of file).

Pour continuer dans la boucle jusqu'à ce que EOF soit vrai, nous allons commencer notre boucle avec DO WHILE .NOT. EOF (FAIRE TANT QUE NON FIN DE FICHIER). Ceci peut sembler un peu barbare, mais la syntaxe de dBASE II l'exige. Les points de chaque côté du mot NOT font également partie de la syntaxe. A la fin de notre programme, nous placerons ENDDO. Ceci veut dire que le programme devra retourner à l'instruction DO WHILE, et de nouveau tester la fin de fichier (EOF).

Ce type de boucle est tellement classique en programmation qu'il nous servira d'armature dans de nombreuses applications.

```
* ARMATURE DO WHILE
* CETTE ARMATURE NOUS SERVIRA TANT QU'IL SERA NECESSAIRE DE TRAITER
* UN FICHIER ENREGISTREMENT PAR ENREGISTREMENT, JUSQU'A LA FIN DU
* FICHIER.
*
DO WHILE .NOT. EOF
   *
   * TRAITEZ L'ENREGISTREMENT
   *
   SKIP
ENDDO
```

La plupart des programmeurs décalent leurs instructions qui sont dans une boucle. C'est seulement une manière de rendre le programme plus facile à lire. Ajoutons cette armature dans notre programme d'étiquettes, et affichhons de nouveau les étiquettes.

```
. MODIFY COMMAND ETIQUET

* ETIQUET.CMD   VERSION 3
* CE PROGRAMME AFFICHERA DES ETIQUETTES DE MAILING
* A PARTIR DU FICHIER MAIL.DBF
*
USE MAIL
DO WHILE .NOT. EOF
   ? TRIM(PRENOM), NOM
   ? ADRESSE
   ? CODEPOST, VILLE
SKIP
ENDDO
              (tapez CTRL W)

. DO ETIQUET

FRANCIS PRADEL
100 RUE FEDERALE
01284 BELLEY
 *** ENREGISTREMENT : 00002
SAMUEL BEN
 30 OUEST RUE ALEMEDA
97821 ST. LOUIS
*** ENREGISTREMENT : 00002
```

Bon, c'est déjà mieux, mais il y a encore du travail. Les étiquettes ont été affichées pour deux enregistrements, et le programme s'est arrêté après le dernier enregistrement. Notre seul problème maintenant c'est le « ENREGISTREMENT : 00002 » après chaque étiquette. Le numéro d'enregistrement est affiché par dBASE II chaque fois qu'il y a un saut (SKIP).

Cela peut s'arranger, mais voyons d'abord comment fonctionne ce programme. Nous suivrons le schéma du programme en pseudo-code comme tout à l'heure pour rendre l'analyse plus facile.

1. Utiliser le fichier de liste mailing.
2. Aller au premier enregistrement.
3. Contrôler pour être sûr que l'on n'est pas au dernier enregistrement.
4. Afficher le format de l'étiquette.
5. Sauter au deuxième enregistrement. dBASE II affiche le nouveau numéro de l'enregistrement lorsqu'il saute (SKIP). C'est pourquoi « ENREGISTREMENT : 00002 » est affiché.
6. Contrôler la fin de fichier (EOF).
7. Afficher une deuxième étiquette.
8. Sauter de nouveau. Il n'est pas possible de sauter (SKIP) après la fin du fichier, c'est pour cette raison que nous obtenons « ENREGISTREMENT : 00002 » une seconde fois.
9. Contrôler la fin de fichier (EOF). Si maintenant EOF est vrai, alors le programme s'arrête.

   Nous pouvons maintenant observer que EOF est seulement vrai lorsque nous essayons de sauter (SKIP) après la fin du fichier, et non pas lorsque le pointeur d'enregistrement est au dernier enregistrement.

   Le programme fonctionne correctement, mais nous ne souhaitons pas obtenir « ENREGISTREMENT : 00002 » à l'écran. On peut supprimer de tels messages à l'écran par la commande SET TALK OFF.

   Une autre modification serait d'avoir chaque étiquette séparée par une ligne vierge. On peut obtenir ce résultat avec une simple commande ?, qui veut dire « afficher une ligne vierge ». Nous séparerons chaque étiquette avec trois lignes vierges. Pour la bonne raison que la plupart des étiquettes disponibles dans le commerce comportent six lignes. Lorsque nous commençons à imprimer des étiquettes sur l'imprimante, nous n'aurons besoin que d'un enregistrement par étiquette. Les trois lignes de données et les trois lignes vierges rempliront correctement l'étiquette.

```
. MODIFY COMMAND ETIQUET

*  ETIQUET.CMD   VERSION 4
*  CE PROGRAMME IMPRIMERA DES ETIQUETTES MAILING
*  A PARTIR DU FICHIER MAIL.DBF

SET TALK OFF
USE MAIL
DO WHILE .NOT. EOF
   ? TRIM(PRENOM), NOM
   ? ADRESSE
   ? CODEPOST, VILLE
   ?
   ?
   ?
   SKIP
ENDDO


. DO ETIQUET

FRANCIS PRADEL
100 RUE FEDERALE
01284 BELLEY

SAMUEL BEN
30 OUEST RUE ALEMEDA
97821 ST.LOUIS
```

L'ENVOI DES ETIQUETTES A L'IMPRIMANTE

Maintenant que notre programme d'étiquettes fonctionne, nous avons besoin d'une trace papier. Les étiquettes sur écran ne sont pas d'une très grande utilité. Pour mettre en route une imprimante à partir du programme, on place seulement la commande SET PRINT ON. Ceci aura pour effet de diriger tout ce qui va à l'écran vers l'imprimante. L'impression est stoppée avec la commande SET PRINT OFF. C'est logique, n'est-ce pas ?

```
. MODIFY COMMAND ETIQUET

*  ETIQUET.CMD   VERSION 5
*  CE PROGRAMME IMPRIMERA DES ETIQUETTES MAILING
*  A PARTIR DU FICHIER MAIL.DBF

SET PRINT ON
SET TALK OFF
USE MAIL
DO WHILE .NOT. EOF
   ? TRIM(PRENOM), NOM
   ? ADRESSE
   ? CODEPOST, VILLE
   ?
   ?
   ?
   SKIP
ENDDO
SET PRINT OFF
```

UN PEU DE FANTAISE AVEC LES INSTRUCTIONS IF

Nous allons modifier notre programme pour qu'il n'affiche que certains enregistrements. Par exemple, nous n'afficherons que les étiquettes des personnes habitant dans une certaine ville, ou ayant un code postal particulier. Une liste mailing ne rend service que si nous pouvons l'utiliser sélectivement.

Vous allez comprendre avec un exemple concret. Supposons qu'il y ait 5.000 noms sur une liste mailing. Chaque lettre coûte environ 15 centimes de frappe, 1,20 F de main-d'oeuvre, et 1,80 F de timbre, donc un total de 3,15 F par lettre. Les mailings sont faits à partir de cette liste tous les trimestres.

```
      3,15 F   par lettre
x 5.000,00     lettres
------------
 15.750,00 F   par mailing
x     4        mailings par an
------------
 63.000,00 F   par an
```

Si seulement 10 % de ces lettres peuvent être exclues, à partir d'une sélection faite sur l'adresse ou un autre critère, on réalise une économie chaque année de 6.300 F par an. Cela vous rembourse largement sur l'achat du logiciel dBASE II.

On peut obtenir ces économies uniquement avec un seul mot, IF. Cette commande permet à dBASE II de savoir s'il doit exécuter ou non certaines opérations. La structure de la commande IF est identique à celle du DO WHILE. On teste une condition, et si elle est vraie, toutes les commandes entre le IF et le ENDIF sont exécutées.

IF condition vraie
exécuter un certain nombre de commandes ENDIF

En ajoutant une construction IF dans notre programme d'étiquettes, nous pouvons imprimer des enregistrements de façon sélective.

Nous allons laisser notre programme d'étiquettes de côté, et retaper un programme sous un nouveau nom, ETIQSEL.

```
. MODIFY COMMAND ETIQSEL

*  ETIQSEL VERSION 1
*  CE PROGRAMME IMPRIMERA DES ETIQUETTES DE MAILING
*  A PARTIR DU FICHIER MAIL.DBF SUR LE SEUL CRITERE DES VILLES
*
SET TALK OFF
USE MAIL
SET PRINT ON
DO WHILE .NOT. EOF
   IF VILLE = "ST.LOUIS"
      ? TRIM(PRENOM), NOM
      ? ADRESSE
      ? CODEPOST, VILLE
      ?
      ?
      ?
   ENDIF
   SKIP
ENDDO
SET PRINT OFF
```

```
. DO ETIQSEL

SAMUEL BEN
30 OUEST RUE ALEMEDA
97821 ST. LOUIS
```

Les lignes du programme, entre le IF et le ENDIF, n'ont été exécutées que pour les enregistrements dont la ville était égale à ST. LOUIS. Nous avons donc économisé 50 % de nos frais de postage.

PERMETTRE A L'UTILISATEUR DE FAIRE UN CHOIX

Puisque nous pouvons imprimer des étiquettes sur des critères sélectifs, pourquoi ne pas laisser à l'utilisateur du programme le choix de décider de la ville. Nous ajouterons donc une question avec la commande ACCEPT, qui permettra de rentrer la ville à sélectionner.

```
. MODIFY COMMAND ETIQSEL

*  ETIQSEL   VERSION 2
*  CE PROGRAMME IMPRIMERA DES ETIQUETTES DE MAILING
*  A PARTIR DU FICHIER MAIL.DBF APRES AVOIR SAISI
*  LA VILLE A SELECTIONNER.
*
SET TALK OFF
USE MAIL
ACCEPT "POUR QUELLE VILLE VOULEZ-VOUS QUE J'IMPRIME LES ETIQUETTES ?" ;
TO CHOIX
SET PRINT ON
DO WHILE .NOT. EOF
   IF VILLE = CHOIX
      ? TRIM(PRENOM), NOM
      ? ADRESSE
      ? CODEPOST, VILLE
      ?
      ?
      ?
   ENDIF
   SKIP
ENDDO
SET PRINT OFF
```

```
. DO ETIQSEL

POUR QUELLE VILLE DOIS-JE IMPRIMER LES ETIQUETTES ? : ST.LOUIS

SAMUEL BEN
30 OUEST RUE ALEMEDA
97821 ST. LOUIS
```

AJOUTER UN MENU

Nous avons maintenant deux programmes pour imprimer des étiquettes, ETIQUETTE.CMD et ETIQSEL.CMD. Un utilisateur comme nous qui connaît dBASE II, sera capable d'utiliser l'un des deux programmes avec la commande DO. Si quelqu'un d'autre utilise nos programmes, il devra soit être assisté, soit recevoir des instructions détaillées. Il existe une méthode qui permet à de nouveaux utilisateurs d'imprimer des étiquettes sans avoir à apprendre à se servir de dBASE II.

La démarche la plus simple est d'utiliser d'abord un menu. Les nouveaux utilisateurs se sentiront tout de suite à l'aise avec dBASE II, et seront capables ensuite de faire l'apprentissage des commandes dans un deuxième temps.

Ecrire un programme de menu avec dBASE II est un exercice simple. Tous les programmes de menu effacent d'abord l'écran, présentent les choix disponibles, et permettent à l'utilisateur de sélectionner une des différentes possibilités. Dès que vous aurez créé un exemple de menu, vous en saurez suffisamment pour en créer d'autres qui s'adapteront à la plupart des situations.

Nous devons apprendre de nouvelles commandes pour écrire un bon menu. La première est la commande ERASE, qui efface simplement l'écran, et qui doit donc se trouver dans chaque programme. Il serait difficile de lire un menu, si l'écran est embarrassé de texte des commandes précédentes. Vous utiliserez donc ERASE dans tous vos programmes, chaque fois que vous voudrez commencer avec un écran bien net.

On a besoin d'une boucle, qui représentera le menu tant que l'utilisateur n'a pas choisi d'arrêter le programme. Une boucle DO WHILE continuera tant que la condition qui suit le WHILE est vraie. Nous allons utiliser l'instruction DO WHILE T, au départ de notre programme. T est une variable système qui est égale à vrai (true). La boucle continuera inlassablement.

Il y a deux manières de quitter une boucle. CANCEL stoppe le programme et revient à dBASE II. QUIT permet de s'arrêter et de revenir à CP/M.

Avec ce programme de menu, nous allons commencer à inclure des commentaires dans le programme lorsqu'ils sont nécessaires. Cela vous permettra de comprendre le programme dès maintenant, et peut-être un peu plus tard lorsque vous voudrez le mettre à jour.

```
. MODIFY COMMAND MENU

* MENU.CMD    VERSION 1
* CE PROGRAMME PRESENTERA LES DIFFERENTES OPTIONS DE SORTIES
* D'ETIQUETTES. IL OFFRIRA EGALEMENT UNE POSSIBILITE D'AIDE.

* EVITER QUE DBASE ENVOIE DES MESSAGES A L'ECRAN
SET TALK OFF

* DEMARRAGE DE LA BOUCLE
DO WHILE T

* PRESENTATION DU MENU
  ERASE
  ?
  ?
  ? "_____________________________________________________________"
  ? "                MENU D'IMPRESSION DES ETIQUETTES"
  ?
  ?
  ?
  ? "      1. IMPRESSION DES ETIQUETTES POUR TOUTES LES VILLES"
  ?
  ? "      2. IMPRESSION DES ETIQUETTES POUR CERTAINES VILLES"
  ?
  ? "      3. EXPLICATIONS "
  ?
  ? "      4. QUITTER LE MENU ET REVENIR A DBASE"
  ?
  ? "      5. QUITTER LE MENU ET REVENIR A CP/M"
  ?
  ?
```

```
   * DEMANDER CE QUE VEUT L'UTILISATEUR
   ACCEPT  "          QUE DOIS-JE FAIRE" TO CHOIX

   * FAIRE CE QUE VEUT L'UTILISATEUR
   IF CHOIX = "1"
      DO ETIQUET
      * IMPRIMER TOUTES LES ETIQUETTES
   ENDIF

   IF CHOIX = "2"
      DO ETIQSEL
      * IMPRIMER LES ETIQUETTES POUR CERTAINES VILLES
   ENDIF

   IF CHOIX = "3"
      DO AIDE
      * PRESENTATION D'EXPLICATIONS SUR ECRAN
   ENDIF

   IF CHOIX = "4"
      CANCEL
      * REVENIR A DBASE
   ENDIF

   IF CHOIX = "5"
      QUIT
      * REVENIR A CP/M
   ENDIF

ENDDO
```

Un programme de menu doit avoir une structure standard. Un écran rempli d'options est affiché, on demande une sélection, et le programme correspondant est exécuté. Le programme est cerné par une boucle, de telle façon qu'il continue jusqu'à ce que l'utilisateur ait fini sa sélection.

Afin de commencer par le programme de menu, l'utilisateur doit d'abord mettre en route dBASE II et taper DO MENU, ou démarrer le programme à partir de CP/M avec la commande DBASE MENU. Chaque programme dBASE II peut être exécuté à partir de CP/M.

DES EXPLICATIONS AU PROGRAMME

Lorsque nous avons écrit le menu, nous avons inclu un programme d'Aide. Chaque fois que nous écrivons un menu, il est bon d'y ajouter une aide pour l'utilisateur. Cette aide peut comporter des explications, ou fournir le nom de la personne compétente à contacter. Ecrire un programme d'aide est facile. Cela consiste à effacer l'écran, afficher un peu de texte, et attendre que l'utilisateur désire revenir au menu.

```
. MODIFY COMMANDE AIDE

* AIDE.CMD     VERSION 1
* PRESENTATION D'UN ECRAN D'AIDE POUR LE PROGRAMME
* D'IMPRESSION DES ETIQUETTES.
*

ERASE

* PLACEZ VOS MESSAGES D'AIDE A CET ENDROIT
?
?
?  "POUR TOUTE QUESTION, CONTACTEZ VOTRE INFORMATICIEN"
?
?
ACCEPT "APPUYEZ SUR RETURN POUR CONTINUER" TO ATTENDRE
```

L'instruction ACCEPT fonctionne de telle façon que l'écran reste en place jusqu'à ce que la touche RETURN soit actionnée. Le programme retourne alors à MENU.CMD. ATTENDRE est appelé une variable « morte », parce qu'elle ne manipule aucune information.

QUELQUES MOTS SUR LA COMMANDE DO

dBASE II permet à un programme d'en exécuter un autre. C'est ce que l'on appelle un programme « emboîté ». Lorsqu'un programme en fait exécuter un autre, on dit que le premier appelle le second. Lorsque le programme appelé est terminé, dBASE retournera au programme qui l'a appelé par la commande DO.

Dans dBASE II, les appels par DO peuvent avoir 16 niveaux de profondeur, ce qui veut dire qu'un programme peut faire exécuter un programme, qui peut lui-même faire exécuter un programme, qui peut à son tour en faire exécuter un autre, etc. N'imbriquez pas trop vos programmes, parce que vous risquez d'en perdre le fil un peu plus tard. C'est un peu comme si vous vous regardiez dans un double jeu de miroirs.

CONCLUSION

Si vous avez suivi tous les exemples de ce chapitre, vous commencez à avoir une idée de ce que peut être la programmation. C'est finalement l'inlassable répétition de construction de projets, d'écriture de code, de mise au point, encore plus de projets, encore plus de code, et encore plus de mise au point. Maintenant vous commencez à comprendre pourquoi les programmeurs touchent de bons salaires.

Nous en arrivons au dernier chapitre recommandé pour les débutants. Le reste de cet ouvrage avance à un rythme plus rapide. L'ensemble des notions vues jusqu'ici est une bonne préparation aux chapitres suivants, mais vous pourriez aussi lire ce livre complètement. Il peut s'avérer nécessaire de consulter quelques ouvrages sur la programmation listés dans la bibliographie, si cela vous paraît trop ardu.

CHAPITRE IV

DEVELOPPEMENT DES TECHNIQUES dBASE II STANDARD

Comme vous pouvez vous en apercevoir à la lecture de ce manuel, nous avons simplement effleuré la surface du langage de dBASE II. Comme l'introduction le disait, l'objectif de cet ouvrage n'est pas de redéfinir chaque commande, mais de présenter un aperçu du travail.

Ce chapitre vous aidera à élargir ce point de vue, en vous montrant de nombreuses techniques classiques utilisées dans la programmation dBASE II. Malheureusement, les concepts étudiés sont tellement larges qu'une démarche structurée est vaine. Nous avons essayé de présenter les nouvelles commandes dans leur ordre d'apparition dans les programmes.

Bien entendu, ces exemples seront utilisés dans les futurs chapitres, aussi il est conseillé de ne pas en sauter.

MISE EN PLACE DE L'ENVIRONNEMENT DE TRAVAIL

La première nécessité de tout programme sera de créer son environnement propre. Il faudra donc se préoccuper de la manière avec laquelle dBASE II apparaîtra à l'utilisateur, et également de la façon dont on peut entrer les données. La commande SET servira à placer un certain nombre de paramètres, soit allumés soit éteints (ON/OFF).

Nous avons vu la commande SET TALK OFF, qui empêche l'envoi des messages dBASE II à l'écran. Plusieurs SET sont combinés avec l'entrée de données et ils opèrent un

contrôle sur la saisie des données en mode plein écran. Voici une liste partielle de ces commandes SET et leurs actions particulières :

SET	ACTIONS
BELL ON	Mis sur ON, le terminal émet un bip à la fin du champ d'une rubrique.
CONFIRM OFF	Mis sur ON, on doit appuyer sur la touche RETURN pour passer à la rubrique suivante.
INTENSITY ON	Mis sur ON, le terminal affiche sur l'écran les caractères en vidéo inverse.

Beaucoup de ces paramètres sont affaire de goût personnel, donc la voie la plus sûre est de les laisser au début dans leur position originale. Lorsque vous devriendrez plus expérimenté avec dBASE II, vous pourrez commencer à vous en servir un peu plus.

Une nouvelle commande SET, SET ESCAPE OFF, est vraiment réservée aux personnes qui ont une très grande habitude de dBASE II. En mode normal, toute pression sur la touche escape stoppera le programme en exécution. Si cela vous arrive lorsqu'un programme est en train de mettre à jour un fichier, les résultats sur ce fichier pourraient être désastreux. Lorsque ESCAPE est éteint (SET ESCAPE OFF), la touche escape n'a plus aucun effet. Cette commande n'aura plus d'utilité tant que le programme n'est pas terminé. Si ESCAPE est OFF et si un programme se trouve dans une boucle sans fin, ou tout autre problème, le seul moyen d'en sortir est d'appuyer le bouton de reset.

FONCTIONS INTERNES

Beaucoup de petites tâches, qui dans d'autres langages nécessiteraient l'écriture de petites routines de programme, sont fournies par dBASE II. Nous appellerons ces routines des fonctions. Une fonction manipule une expression ou une variable, et la convertit sous une nouvelle forme.

Dans presque toutes les fonctions, on fournit un argument (qui sera manipulé par la fonction). dBASE II traduira la fonction et l'argument pour restituer ensuite l'information sous la forme désirée.

Nous avons déjà utilisé la fonction TRIM. Elle ramène le contenu d'une rubrique particulière sans les espaces qui remplissent le reste de la rubrique.

```
. ? NOM, PRENOM

MARTIN          DANIEL

. ? TRIM(NOM), PRENOM

MARTIN DANIEL
```

Beaucoup de fonctions donnent un résultat qui est soit vrai soit faux. Par exemple, FILE(« nom-de-fichier ») est vrai si le nom de fichier est trouvé sur le disque. Cette fonction peut être utilisée par un programme pour tester l'existence d'un fichier de données.

Ce type de fonction vraie ou fausse peut être utilisé dans une instruction IF pour déterminer la nouvelle action à entreprendre. Par exemple :

```
IF FILE("MAIL.DBF")
        USE MAIL
ELSE
        ? "MAIL.DBF n'est pas sur le disque"
ENDIF
```

Si le fichier MAIL.DBF est présent sur le disque, la fonction FILE est vraie, et l'on utilisera MAIL (USE). Le point d'interrogation fait partie de la syntaxe dBASE II.

La fonction !( ) convertira toutes les lettres d'une variable, ou d'une instruction entre guillemets, en lettres majuscules.

```
. ? !("bonjour")

BONJOUR
```

Nous allons maintenant écrire des programmes plus complexes, un certain nombre de fonctions vont donc être introduites.

CONDITIONS MULTIPLES

Il y a beaucoup d'occasions où plus d'une condition doit être contrôlée par une instruction IF ou DO WHILE. La méthode correcte est de connecter chaque condition avec .AND. ou .OR. (ET ou OU). Il peut être également intéressant de savoir si une condition n'est pas vraie (.NOT.). Nous avons vu cela dans la commande DO WHILE .NOT. EOF.

Lorsque les conditions deviennent trop complexes, on a recours à l'utilisation de parenthèses pour les séparer. Voici un exemple court d'un programme d'inventaire imaginaire :

```
LIST FOR VENDEUR = "AMSTRAD" .AND. (PRIX > 100 .AND. PRIX < 300)
```

Cette commande a pour effet de lister les articles fabriqués par la société Amstrad dont le prix est compris entre 100 et 300 francs. Il faut un peu d'habitude pratique pour bien se débrouiller avec des conditions multiples.

GERER LA MEMOIRE

Lorsque l'on réalise des systèmes importants qui impliquent beaucoup de fichiers CMD, il faut réserver une attention particulière aux variables mémoire utilisées. La limite maximum est de 64 variables mémoire disponibles, et l'on doit supprimer certaines variables lorsqu'elles ne sont plus nécessaires (RELEASE).

Si l'on n'y prend pas garde, la mémoire peut être remplie avant même la fin du programme. Les programmes peuvent sauvegarder l'information mémoire pour qu'elle soit ré-employée par d'autres programmes. En utilisant cette technique, un programme peut envoyer des messages et des résultats à un autre programme. C'est ce que l'on appelle le « passage de paramètres ».

Avec dBASE II, toutes les variables mémoire sont disponibles pour être utilisées et modifiées par chaque programme. C'est un avantage et un inconvénient. Le programmeur doit donc être prudent lorsqu'il change une variable dans un programme, si le même nom de variable est employé dans d'autres programmes.

Une pratique classique est de supprimer les variables (RELEASE) qui sont créées par un programme, et mettre la mémoire dans son état initial, avant de quitter le programme. Beaucoup de petits modules peuvent alors être interconnectés sans que l'on ait à s'inquiéter des modifications des variables mémoire.

S'IL VOUS PLAIT, CASE OU IF ?

Dans certains types de programmes, comme les menus, il est habituel de rencontrer une grande série d'instructions IF. Chaque IF permettra la sélection d'une des fonctions du menu. L'instruction dBASE II, DO CASE, est une commande très puissante, qui s'adapte tout à fait aux situations de conditions multiples. La syntaxe du DO CASE est similaire à celle de l'instruction DO WHILE.

```
DO CASE
CASE condition 1
instructions
CASE condition 2
instructions
CASE condition N
instructions
OTHERWISE
instructions
ENDCASE
```

Beaucoup de CASE peuvent être contrôlés, et chaque condition est testée à son tour. Lorsqu'une condition apparaît comme vraie, toutes les instructions sont exécutées jusqu'à la prochaine instructions CASE. Après la rencontre du premier CASE vrai, l'instruction suivante à être exécutée sera la suite des instructions qui se trouvent après le ENDCASE. Si aucune des conditions n'est vraie, c'est le OTHERWISE qui est exécuté. La commande OTHERWISE est facultative.

Reprenons notre programme de menu du chapitre précédent afin de lui appliquer l'instruction DO CASE.

```
. MODIFY COMMAND MENU

* MENU.CMD     VERSION 2
* CE PROGRAMME PRESENTE DIFFERENTES OPTIONS D'IMPRESSION DES
* ETIQUETTES. IL FOURNIT EGALEMENT UNE AIDE A L'UTILISATEUR.

* PREVENTION DES MESSAGES ENVOYES PAR DBASE II SUR L'ECRAN
SET TALK OFF

* ENTREE DANS LA BOUCLE
DO WHILE T
* PRESENTATION DU MENU
  ERASE
  ?
  ?
  ? "_____________________________________________________________"
  ? "                MENU D'IMPRESSION DES ETIQUETTES"
  ? "
  ?
  ?
  ? "      1. IMPRESSION DES ETIQUETTES POUR TOUTES LES VILLES"
  ?
  ? "      2. IMPRESSION DES ETIQUETTES POUR CERTAINES VILLES"
  ?
  ? "      3. EXPLICATIONS "
  ?
  ? "      4. QUITTER LE MENU ET REVENIR A DBASE"
  ?
  ? "      5. QUITTER LE MENU ET REVENIR A CP/M"
  ?
  ?

  * DEMANDER CE QUE VEUT L'UTILISATEUR
  ACCEPT  "         QUE DOIS-JE FAIRE" TO SUITE

  * FAIRE CE QUE VEUT L'UTILISATEUR
  DO CASE
```

```
   CASE SUITE = "1"
      DO ETIQUETTE
      * IMPRESSION DE TOUTES LES ETIQUETTES

   CASE SUITE = "2"
      DO ETIQSEL
      * IMPRESSION DES ETIQUETTES POUR CERTAINES VILLES

   CASE SUITE = "3"
      DO AIDE
      * PRESENTATION D'EXPLICATIONS SUR ECRAN

   CASE SUITE = "4"
      CANCEL
      * REVENIR A DBASE

   CASE SUITE = "5"
      QUIT
      * REVENIR A CP/M

   ENDCASE

ENDDO
```

L'instruction DO CASE rend les programmes plus faciles à lire et à écrire. Nos programmes les utiliseront abondamment.

LES BOUCLES DO WHILE EN DETAIL

Les applications nécessitent souvent qu'un programme répète une procédure un certain nombre de fois, ou tant qu'une condition est vérifiée. La forme DO WHILE .NOT. EOF est une condition typique de boucle. Nous allons décrire brièvement quelques méthodes classiques de contrôle des boucles.

CONDITION DE BOUCLE	OBJECTIF
DO WHILE T	Ce type de boucle est souvent utilisé pour constituer la boucle supérieure d'un programme de menu principal. C'est une boucle sans fin, parce que T sera toujours vrai. La seule façon d'en sortir est d'utiliser la commande QUIT.
DO WHILE .NOT. EOF	Permet le controle des programmes qui traitent un enregistrement à la fois, jusqu'à l'atteinte de la fin de fichier.
STORE T TO CONTINUE DO WHILE CONTINUE	On crée une variable pour contrôler la boucle. Lorsqu'on veut sortir de cette boucle, on met CONTINUE à F, c'est-à-dire faux. La fois suivante, CONTINUE est testé et la boucle prendra fin.
STORE 0 TO COMPTEUR DO WHILE COMPTEUR < 10	Si l'on souhaite qu'une boucle se répète un certain nombre de fois. On crée une variable compteur. Chaque fois qu'une tâche est achevée, COMPTEUR sera incrémenté de un. Lorsque le nombre de passages désiré est achevé, et que COMPTEUR devient égal à 10, la boucle s'arrête.

Il est souhaitable que vous utilisiez des boucles standard dans vos programmes dBASE II. Vous gagnerez en temps d'écriture et rendrez les programmes plus faciles à comprendre.

Il est possible d'imbriquer des boucles DO WHILE, ce qui veut dire qu'une boucle peut se trouver à l'intérieur d'une autre. Il est fondamental que chaque boucle soit complètement imbriquée dans l'autre. Voici un exemple correct de boucle imbriquée :

```
DO WHILE T
   DO WHILE .NOT. EOF
     ? NOM
     SKIP
   ENDDO
ENDDO
```

L'indentation de chaque niveau de boucle DO n'est pas obligatoire, mais cela vous aide à suivre l'ordre d'imbrication correct. Les mauvaises imbrications de boucles peuvent avoir des résultats imprévisibles.

CREATION DE FORMATS DE SAISIE

Une des caractéristiques les plus agréables de la programmation dBASE II est la possibilité de créer des formats sur l'écran qui auront exactement la même forme que les documents originaux. Vous disposez d'un éditeur plein écran, de la même façon que dans la fonction EDIT ou APPEND. Lorsque beaucoup d'enregistrements doivent être ajoutés, cela se révèle très pratique.

On crée le format en déclarant la position sur l'écran du terminal de chaque partie d'information à afficher. Un terminal classique a 24 lignes de 80 caractères chacune. Comme dans beaucoup de langages de programmation, dBASE II commence à compter à partir de 0, autrement dit la première ligne d'écran est la ligne 0. dBASE II se réserve cette ligne pour son propre usage. Les colonnes vont de 0 à 79, donc le premier caractère est affiché à la colonne 0.

La commande @ SAY, qui se traduirait par « à telle position afficher », est utilisée pour entrer ces coordonnées. On indique d'abord la position de la ligne, suivie par la position de la colonne. Pour placer le mot « BONJOUR » sur la première ligne disponible à la position colonne 10, nous utiliserons la commande suivante :

```
@ 1,9 SAY "BONJOUR"
```

La position colonne est 9 parce que nous commençons à compter à partir de 0. La commande @ SAY demande à dBASE II d'imprimer simplement le texte qui suit. Si nous avions voulu permettre la saisie d'une donnée dans une variable à cette position, nous aurions utilisé la commande @ GET.

Si des variables mémoire sont utilisées dans le format, elles doivent être initialisées avec les commandes STORE, de façon à déclarer à dBASE II leur type et leur capacité.

Après avoir initialisé les variables et défini leur position écran, on fournit la commande READ et le format sera affiché sur le terminal. Nous pouvons déplacer le curseur et saisir les données dans chacune des rubriques du GET. Lorsque toutes les rubriques sont

passées, toutes les données saisies viendront se placer dans les variables utilisées dans les GET. Voici un exemple de jeu de commandes qui affichera un masque de saisie écran.

```
. MODIFY COMMAND ECRAN1

STORE "                " TO nom
STORE "                                " TO adresse
STORE "           " TO ville
STORE "  " TO departement
STORE "      " TO code postal
@ 4,4   SAY "Nom"
@ 4,20  GET nom
@ 6,4   SAY "Adresse"
@ 6,20  GET adresse
@ 8,4   SAY "Ville"
@ 8,11  GET ville
@ 8,25  SAY "Departement"
@ 8,33  GET département
@ 8,45  SAY "CODE POSTAL"
@ 8,50  GET code postal
READ
```

```
. DO ECRAN1

Nom            :                :

Adresse        :                                  :

Ville  :         : Département  :  :    Code postal  :        :
```

La possibilité de création de formats pour faire des masques d'écran ou des états imprimés est l'un des avantages majeurs sur des langages comme le BASIC. Une simple commande READ permet toutes les éditions en mode plein écran.

ZIP

Quoique les commandes @ SAY et @ GET sont extrêmement puissantes, créer un format complexe peut être un processus relativement long. Pour simplifier cette tâche, un programme utilitaire appelé ZIP est fourni avec dBASE II. ZIP est un générateur de

programmes dBASE II, ce qui veut dire qu'il peut écrire des programmes dBASE II d'après vos spécifications.

Les programmes écrits par ZIP incluent toutes les instructions @ **SAY** et @ **GET** nécessaires pour présenter un format complet à dBASE II. Tout ce que vous avez à faire est d'exécuter ZIP à partir de CP/M, dessiner votre format sur l'écran, et déclencher l'écriture du programme par ZIP. Le programme est écrit sur le disque, prêt à fonctionner avec dBASE II. Les instructions détaillées des opérations ZIP peuvent être retrouvées dans le manuel dBASE II.

Pour travailler avec ZIP, il faut suivre quelques règles de base. Pour obtenir l'apparition de texte dans une instruction @ **SAY**, tapez simplement le texte sur l'écran. Vous devez traiter vos variables avec le signe @ pour les instructions @ **SAY**, et un # pour les instructions @ **GET**.

Lorsque vous demandez à ZIP d'écrire le programme, il va produire trois fichiers : un fichier ZIP qui contient une copie de votre format pour des modifications futures, un fichier ZPR qui est une copie imprimable de votre écran pour les besoins de documentation, et un fichier CMD ou FMT qui contient les instructions @ **SAY** et @ **GET** pour le format.

Une fois que le programme est terminé, vous pouvez faire démarrer dBASE II et utiliser le nouveau programme comme si c'était vous qui l'aviez écrit. Voici un exemple de format, suivi par le programme dBASE II produit par l'utilitaire ZIP.

```
+-----------------------------------------------------------+
!                                                           !
!   Nom : #nom                                              !
!                                                           !
!   Adresse : #adresse                                      !
!                                                           !
!   Ville : #ville                                          !
!                                                           !
!   Département : #département     Code postal : #cod:post !
!                                                           !
!                                                           !
!                                                           !
+-----------------------------------------------------------+

   S'il vous plait, tapez vos nom et adresse
```

```
@  0,15 SAY "================================================"
@  1,15 SAY "!                                              !"
@  2,15 SAY "!   Nom :"
@  2,26 GET nom
@  2,58 SAY "!"
@  3,15 SAY "!                                              !"
@  4,15 SAY "!   Adresse :"
@  4,29 GET adresse
@  4,58 SAY "!"
@  5,15 SAY "!                                              !"
@  6,15 SAY "!   Ville :"
@  6,26 GET ville
@  6,58 SAY "!"
@  7,15 SAY "!                                              !"
@  8,15 SAY "!   Département :"
@  8,27 GET departement
@  8,39 SAY "Code postal :"
@  8,45 GET code post
@  8,58 SAY "!"
@  9,15 SAY "!                                              !"
@ 10,15 SAY "!                                              !"
@ 11,15 SAY "!                                              !"
@ 12,15 SAY "================================================"
@ 14,21 SAY "S'il vous plait, tapez vos nom et adresse"
```

Ce programme a été créé en moins de 5 minutes avec l'utilitaire ZIP. Si vous désirez le modifier, vous demandez à ZIP de le lire de nouveau à partir du disque pour en produire une nouvelle version.

SAUVEGARDE DES FORMATS DANS UN FICHIER FMT

Au lieu de placer des commandes @ SAY et @ GET dans un fichier CMD, elles peuvent être sauvegardées dans un fichier FMT. C'est exactement la même chose qu'un fichier CMD, mais qui ne contient pas d'autres instructions que SAY et GET. Pour afficher un format dans un fichier CMD, vous devez faire exécuter le fichier par DO. Pour utiliser un fichier FMT, indiquez simplement à dBASE II la commande SET FORMAT TO le fichier en question. Lorsque l'on fournit la commande READ, cela a pour effet d'afficher le format défini par le programme.

La distinction entre les deux méthodes d'affichage des formats est minine et c'est une affaire de goût personnel dans la plupart des cas. Vous pouvez expérimenter ces deux techniques.

ENVOI DES FORMATS A L'IMPRIMANTE

Un format qui vient d'être créé pour l'affichage sur le terminal, peut aussi être envoyé à l'imprimante. La commande SET PRINT ON aura pour effet d'envoyer vers l'imprimante toutes les instructions **@ SAY.** Toute instruction **@ GET** sera ignorée, et n'ira pas vers l'imprimante.

Pour utiliser l'imprimante, le moyen habituel est de créer une version du format pour l'écran, et de convertir ensuite tout *@* **GET** en *@* **SAY**, ce qui permettra d'utiliser les mêmes formats pour l'écran et pour l'imprimante.

MIGNON COMME UNE IMAGE (PICTURE)

Avec la commande **@ GET**, il existe une option disponible, la clause PICTURE. Elle définira le format d'affichage et de saisie sur l'écran.

Un exemple classique est l'entrée de la date. C'est souvent la peste des programmeurs, parce qu'il est très difficile de demander aux gens d'entrer les dates sous un format correct. Certains utilisent Janvier 1, 1986, d'autres 1/1/86, ou 01-01-86, etc.

Si on inclut **@ 10,20 GET DATE PICTURE « 99/99/99 »** dans un programme, la rubrique date sera affichée avec le format suivant :

: / / :

Seuls les nombres seront acceptés entre les barres obliques ; si vous entrez un caractère, le terminal émettra un bip. De plus, le symbole / est facultatif. Il est important d'avoir un seul format de date.

Le type de données permises dans une rubrique est déterminé par le type de caractères utilisés dans la commande PICTURE. Dans l'exemple précédent, il s'agissait du chiffre 9, ce qui oblige donc à entrer des nombres. Le signe ! convertira automatiquement toute

entrée en caractères majuscules. Vous pourrez utiliser cette astuce dans vos programmes de menu. Les autres caractères symboliques de PICTURE sont définis dans le manuel dBASE II.

ATTENTE D'UN SIMPLE CARACTERE (WAIT)

De nombreux programmes, comme les menus, se contentent d'une commande de saisie d'un seul aractère. Ce qui signifie que la touche return n'a pas besoin d'être appuyée pour valider l'entrée du caractère.

La commande dBASE II qui correspond à la saisie d'un caractère est WAIT TO. WAIT ressemble à ACCEPT, mais on ne rentre qu'un caractère et il n'y a pas de message à afficher. Le caractère saisi est stocké dans une variable mémoire.

```
. WAIT TO SUITE

WAITING A

. ? SUITE

A
```

La possibilité d'utiliser l'entrée d'une simple touche est intéressante, mais le message WAITING est désagréable. Le seul moyen d'empêcher ce message est d'utiliser SET CONSOLE OFF avant le WAIT. Lorsque CONSOLE est mis OFF, rien n'apparaît à l'écran, mais le clavier reste opérationnel pour les entrées de données. Les lignes suivantes de codes permettront une simple entrée de données sans le message WAITING à l'écran.

```
MODIFY COMMAND ESSAI
SET CONSOLE OFF
WAIT TO SUITE
SET CONSOLE ON
```

Il n'est pas nécessaire de stocker l'entrée de la donnée dans une variable, WAIT peut être utilisé seul.

```
MODIFY COMMAND ESSAI
SET CONSOLE OFF
WAIT
SET CONSOLE ON
```

Ce moyen est très utilisé pour stopper un programme jusqu'à la pression d'une touche.

L'ART DE LA MISE AU POINT

La rencontre de « bug » est une chose courante chez les programmeurs. La programmation avec dBASE II est plus facile que la plupart des langages, mais il est encore possible de faire des fautes. Heureusement, il existe d'excellents outils de dépannage disponibles.

Il y a deux SET, ECHO et STEP, qui fournissent la plupart des possibilités de dépannage. Lorsque ECHO est mis ON, toute commande d'un fichier CMD est affichée sur l'écran pendant son exécution. STEP permet au programmeur d'exécuter une seule ligne à la fois. STEP offre également le choix d'exécuter le pas suivant, ou d'entrer une commande directement au clavier.

Lorsqu'à la fois ECHO et STEP sont mis ON, un programme peut être exécuté une ligne à la fois, et étudié dans son déroulement logique. A tout moment, on peut arrêter le programme et tester les variables ou entrer de nouvelles commandes. Cela ressemble un peu à l'utilisation d'un programme moniteur pour déboguer les instructions du langage d'assemblage.

Ecrivons un petit programme d'exemple et utilisons ces commandes afin de les étudier.

```
. MODIFY COMMAND TEST

USE MAIL
LIST NOM
ACCEPT "CHOISIR UNE VILLE" TO CLE
LISTE FOR VILLE = CLE

. DO TEST

00001 PRADEL
00002 LERAY
CHOISIR UNE VILLE : YSEURES
00002 FRANCIS     PRADEL        1 RUE DES VIGNES    03400 YSEURES
```

Si vous mettez ECHO sur ON (SET ECHO ON), vous pouvez observer chaque ligne du programme dBASE II pendant son exécution. Vous verrez apparaître à la fois les lignes du programme et les résultats de ces lignes sur l'écran.

```
. SET ECHO ON

. DO TEST
DO TEST
USE MAIL
LIST NOM
00001 PRADEL
00002 LERAY
ACCEPT "CHOISIR UNE VILLE" TO CLE
CHOISIR UNE VILLE : YSEURES
LIST FOR VILLE = CLE
00002 FRANCIS     PRADEL        1 RUE DES VIGNES    03400 YSEURES
```

Lorsque vous utilisez également STEP ON, chaque ligne du programme peut être étudiée en détail. Si vous tapez Y, le programme continue; si vous tapez N, il s'arrête pour entrer une commande.

```
. SET STEP ON

SET STEP ON

. DO TEST

DO TEST
ETAPE SUIVANTE  Y: = OUI,   N: = CMI DE CLAVIER,   ESC: = ANNULATION ?

USE MAIL
ETAPE SUIVANTE  Y: = OUI,   N: = CMI DE CLAVIER,   ESC: = ANNULATION ?

LIST NOM
00001 PRADEL
00002 LERAY


ACCEPT "CHOISIR UNE VILLE" TO CLE
ETAPE SUIVANTE  Y: = OUI,   N: = CMI DE CLAVIER,   ESC: = ANNULATION ?
```

En tapant N, vous pouvez maintenant entrer une commande directement à partir du clavier. Par exemple, vous pourriez examiner le contenu de la variable clé.

```
. LIST MEMORY

LIST MEMORY
CLE         (C)  YSEURES
** TOTAL **      01 VARIABLE USED  00007 OCTETS UTILISES

ETAPE SUIVANTE  Y: = OUI,   N: = CMI DE CLAVIER,   ESC: = ANNULATION ?

LIST FOR VILLE = CLE
00002 FRANCIS     PRADEL        1 RUE DES VIGNES    03400 YSEURES
```

Si vous avez besoin d'obtenir une copie imprimée de cette phase de dépannage, DEBUG peut être mis ON. Cela a pour effet d'envoyer les lignes en écho sur l'imprimante seulement, et empêcher que l'écran soit encombré par les lignes de programme.

STRUCTURE DE PROGRAMMES STANDARD

Le meilleur moyen d'éviter les erreurs est de les prévenir. Nous avons développé précédemment un format standard pour vos programmes. Ces bonnes habitudes vous permettront d'écrire des programmes dBASE II d'une façon naturelle.

Je vous suggère d'utiliser un modèle qui a fait ses preuves. Vous pourriez le suivre jusqu'à ce que vous ayez votre propre style de programmation.

Commentaires — Nom, Date, Auteur, Objectif ou but du programme

Initialisation du format d'écran et des variables

Mise en place d'une boucle si elle est nécessaire

Présentation d'écran ou menu d'options

Entrées de l'utilisateur

Traitement de l'entrée et exécution de la fonction appropriée

Bouclage jusqu'à la fin des opérations

Supprimer toutes les variables créées dans ce fichier de commandes

Restaurer la mémoire dans l'état où elle se trouvait au débuut du programme

CONCLUSION

Nous sommes maintenant prêts pour aborder la programmation des applications. La suite de cet ouvrage vous aidera à écrire des programmes qui utilisent les commandes et les techniques étudiées précédemment. C'est peut-être le moment de réviser les chapitres précédents.

CHAPITRE V

REALISATION D'UN SYSTEME dBASE II COMPLET

ETEIGNEZ VOTRE ORDINATEUR

Depuis quelque temps, nous avons utilisé l'ordinateur pour essayer tous les exemples. Vous avez travaillé les exemples, n'est-ce pas ? Il est temps de résoudre une application complète avec dBASE II, et la première chose à faire lorsque l'on commence un nouveau programme est d'éteindre l'ordinateur.

Prenez le temps de penser à votre application, et de créer un projet et des solutions. Le projet peut inclure des détails sur la taille et la structure de vos fichiers, les types de rapports à imprimer et le pseudo-code pour tous les programmes que vous écrirez. Après tout, est-ce que vous achetez une maison les yeux fermés ?

QUELS TYPES DE PROGRAMME ALLONS-NOUS ECRIRE ?

Le manuel dBASE II montre un système complet de comptabilité écrit en langage dBASE II. Cette technique d'enseignement de la programmation pose un problème, car la complexité du système de comptabilité utilisé dans les exemples peut obscurcir la démonstration des principes de programmation. Le lecteur termine la lecture en se disant, « pourquoi le programme passe les écritures dans ce fichier, avant d'écrire un état sur un autre fichier ? »

Nous adopterons une démarche différente, et en utilisant une application beaucoup plus simple, les habitudes standard de programmation vous apparaîtront plus clairement.

Une fois ces techniques bien comprises, vous serez capables de les appliquer à des problèmes plus complexes.

L'exemple d'application que nous allons créer est une liste de mailing pour une Amicale d'anciens élèves d'un collège, mais le programme peut aisément être modifié pour beaucoup d'autres usages. La plupart des applications exécutent les mêmes opérations sur un jeu de fichiers : ajouter, supprimer, modifier, rechercher, et imprimer les enregistrements. Il est nécessaire également d'écrire des rapports commerciaux, et des programmes qui vont lier plusieurs fichiers entre eux.

Chacun de ces programmes constitue un module indépendant, qui exécute une seule fonction logique. Si les modules sont réalisés et documentés soigneusement, ils peuvent être réutilisés par d'autres applications. Ce qui vous permet d'écrire vos futurs programmes en combinant des modules testés précédemment.

SYSTEME DE LISTE DE MAILING SIMPLE

Story Book University est une petite école des arts et des sciences. Comme beaucoup d'autres universités, son premier objectif est de collecter des cotisations de ses anciens élèves. Pour accomplir ce travail, elle a toute une équipe d'employés de bureau, dont le seul travail est de tenir à jour une liste de mailing de ces anciens élèves. A part la constitution du fichier des adresses des anciens élèves et leur mise à jour (pour qu'ils puissent recevoir les relances de cotisation), chaque cotisation doit pouvoir être retrouvée sur le journal des cotisations des anciens élèves.

Les employés de bureau souhaiteraient qu'un ordinateur puisse exécuter ces tâches à leur place, car l'essentiel de leur travail se passe à des tâches répétitives et fastidieuses, telles que la suppression d'informations doubles dans les fichiers, l'impression des étiquettes mailing, ou l'établissement de références croisées des cotisations.

ANALYSE DU DEROULEMENT DU TRAVAIL D'UNE APPLICATION

Nous devons analyser en premier lieu la manière dont l'information est traitée. Ce qui inclut le volume d'informations à saisir, la fréquence des modifications et des suppressions, et les connaissances informatiques des futurs utilisateurs du programme. Si on ne considère pas ces différents facteurs, il est fort possible d'écrire un programme

qui exécute les fonctions souhaitées mais dont le résultat soit plus difficile à obtenir que les anciennes procédures manuelles.

Les nouvelles cotisations et les nouveaux membres seront saisis à des intervalles réguliers, pas lots et en une seule fois. Après leur saisie, ils devront être vérifiés. Pendant ce processus de révision, il ne devrait pas y avoir de problème pour corriger une information.

Les utilisateurs voudront également faire des recherches dans la base de données, et mettre à jour des informations saisies précédemment. Quelquefois, ils souhaiteront simplement consulter le fichier sans penser forcément à un membre particulier.

A des intervalles réguliers, les étiquettes de mailing devront être produites dans l'ordre des codes postaux pour tenir compte des taxes postales. Il sera nécessaire également de produire un listing de tous les anciens élèves par ordre alphabétique de leur nom.

Un rapport doit pouvoir lister les cotisations de chaque ancien élève, et donner les sous-totaux pour chaque niveau de classe. Un autre rapport sera nécessaire pour faire les regroupements des cotisations faites par chaque ancien élève.

Les utilisateurs de ce système sont pour la plupart des personnes non habituées avec les ordinateurs. Il ne sera donc pas possible que toutes ces personnes apprennent un nouveau langage pour exécuter leur travail. On s'oriente donc vers une démarche de programmation à partir d'un menu. Les utilisateurs les plus avertis pourront être capables d'utiliser dBASE II pour créer des rapports de toutes sortes avec la base de données des anciens élèves.

Lorsque sont impliqués de gros volumes d'informations, il est souvent souhaitable d'utiliser des saisies de données à l'écran qui ressemblent au format des données originales. Le programme doit présenter le même format pour le plus de fonctions possibles. Le passage au nouveau système doit être aussi peu déroutant que possible.

QUELLE DOIT ETRE LA RAPIDITE DU PROGRAMME

Les programmes d'exemple n'ont pas été construits dans un souci de rapidité. Ils sont abondamment documentés de façon à être aussi faciles à lire et à comprendre que possible. Il est toujours plus facile d'améliorer les performances d'un programme bien écrit, que de modifier et de comprendre un programme qui a été réalisé de façon hermétique et compliquée.

Pour rendre ces programmes d'introduction aussi clairs que possible, nous traiterons seulement la partie noms et adresses des anciens élèves dans cette application de mailing. Une deuxième partie, qui gérerait les cotisations individuelles, pourrait être ajoutée par le lecteur.

Bien que le projet original soit imaginé pour l'ensemble de l'application, l'ajout de nouveaux jeux de programme constituera un exemple pratique d'évolution de tout système informatique.

L'ALGORITHME D'UN SIMPLE FICHIER MAILING

Nous allons créer l'ensemble du projet de ce système sous forme de pseudocode. Cette description globale sera suivie par une série de descriptions plus affinées. Si le pseudo-code est écrit avec suffisamment de détails, il pourra être traduit rapidement en code dBASE II. C'est exactement la même chose lorsque l'on écrit un livre à partir d'une ébauche.

Au fur et à mesure de la découverte du système, le nom du fichier de commande (CMD), qui exécute telle fonction, sera imprimé. Les chapitres suivants ajouteront plus de précision au projet, et présenteront alors le programme dBASE II définitif.

1) Présenter le message de l'auteur (AUTEUR)
   Ceci va permettre d'afficher le copyright, et fournir à l'utilisateur quelque chose à lire pendant la mise en route du programme.

2) Initialiser le système (INIT)
   Ceci doit être fait au début du programme de telle façon que tout fichier ou toute variable qu'utilisera le système, soit défini à un endroit particulier.

3) Exécuter une des actions suivantes jusqu'à ce que l'utilisateur en ait terminé avec le système (MAITRE)

4) Afficher l'enregistrement en cours (MAITRE)
   S'il a été marqué pour effacement, alors l'indiquer sur l'écran (TESTSUPR)

5) Les actions suivantes traiteront l'enregistrement courant :
   Effacer l'enregistrement (EFFACE)
   Modifier l'enregistrement (MODIF)
   Imprimer l'enregistrement (IMPRIMER)

6) Les autres enregistrements seront consultés de la manière suivante :
   Se déplacer en arrière dans le fichier (MAITRE)
   Se déplacer en avant dans le fichier (MAITRE)
   Chercher un enregistrement particulier (CHERCHE)

7) Afficher des instructions d'assistance sur le système (AIDE)

8) Présenter une sélection des opérations pour l'entretien du fichier des anciens élèves : (ENTRET)
   Supprimer les doublons du fichier (TESTDBL) (SUPRDBL)
   Vérifier les nouvelles entrées (VERIFNVX)
   Vérifier les enregistrements supprimés (VERIFSUP)
   Supprimer les enregistrements effacés (PURGE)
   Ajouter des valeurs par défaut lors de l'entrée de données (MODIFVAR)
   Faire la sauvegarde des fichiers (BACKUP)

9) Présenter le choix suivant de documents : (DOCUMENT)
   Imprimer des étiquettes pour chaque ancien élève (ETIQUET)
   Imprimer un document pour chaque membre qui liste le total des cotisations et fournit des sous-totaux pour chaque promotion (COTIS.FRM)
   Imprimer l'enregistrement complet de chaque membre (LISTING)

10) Quitter le programme et retourner à CP/M.

DEFINIR LA STRUCTURE DE DONNEES

Avant de commencer avec dBASE II la création d'un fichier DBF, on doit établir un « dictionnaire de données ». C'est simplement un tableau qui liste les noms, les types, les tailles et la description de chaque item distinct de données.

L'établissement de cette liste est aussi important que le produit fini. On s'apercevra que certains items d'information sont plus utiles, découpés en plusieurs parties, tandis que d'autres apparemment sans relation peuvent être réunis en un seul.

On utilisera cette liste pour créer les fichiers dBASE II qui stockeront les données. L'utilisation des mêmes conventions dBASE II au sujet des genres et des types de données permettront un processus de conversion plus facile.

Nom	Type	Taille	Description
Id numéro	C	5	Un nombre unique sera attribué à chaque ancien élève pour permettre une identification correcte.
Prénom	C	10	Par la séparation du nom et du prénom de chaque ancien élève, le nom pourra être utilisé pour une opération de tri.
Nom	C	10	
Adresse	C	20	Puisqu'il s'agit du domicile de l'ancien élève, le nom de l'employeur n'est pas nécessaire.
Ville	C	10	
Départeement	C	2	
Code postal	C	5	Cinq caractères suffisent pour stocker le code postal.
N° téléphone	C	14	On laisse un peu de place pour le code téléphonique de la région.
Promotion	C	2	
Cotisation	N	10	Si quelqu'un cotise plus de 9.999.999,99, l'école pourra s'acheter un nouvel ordinateur.
Nouveau membre	L	1	Ceci est une rubrique logique qui peut avoir deux valeurs : soit vraie soit fausse (true ou false : T ou F). On s'en servira pour marquer l'enregistrement et signifier un nouveau membre.

QUELLES RUBRIQUES DOIVENT ETRE INDEXEES

dBASE II gère automatiquement jusqu'à sept index pour chaque fichier. Lorsqu'un enregistrement est ajouté (APPEND) ou modifié (EDIT), chacun des index doit être mis à jour. Malheureusement, chaque index supplémentaire a pour effet de ralentir le processus de mise à jour.

Lorsque l'on rattache des fichiers d'index, on doit considérer la fréquence d'utilisation de chaque index. Les fichiers d'index qui sont rarement utilisés pourraient être créés le moment venu. Plutôt que de ralentir la saisie des données pour l'ensemble du

programme, il serait préférable d'attendre cinq ou dix minutes lorsque l'on a à éditer un document rare.

Puisque notre liste de mailing est envisagée pour un volume important d'entrées de données, seules les trois plus importantes rubriques seront indexées. Le nom sera indexé pour permettre au module de recherche de trouver (FIND) un élève particulier. L'indexation sur le code postal est nécessaire pour l'impression des étiquettes et le document listant les cotisations doit pouvoir être imprimé dans l'ordre d'années des promotions.

ETABLISSONS UNE NORME DE DOCUMENTATION

Avant que vous vous précipitiez sur l'écriture des programmes, j'attire votre attention sur la forme et le style des programmes. Beaucoup de programmeurs écrivent d'abord les programmes, reviennent ensuite en arrière et ajoutent des commentaires. Cette méthode est satisfaisante pour un simple petit projet personnel, mais la première application un peu plus importante se terminera après un temps beaucoup plus long que prévu.

Un programme abondamment documenté ne doit pas seulement tourner correctement, mais doit pouvoir être écrit de façon compréhensible pour n'importe quel autre programmeur. Même si vous n'avez jamais montré vos programmes à quelqu'un d'autre, ces programmes vous apparaîtront complètement nouveaux dans quelques mois.

Un test intéressant serait de lire le programme à haute voix à un autre programmeur. Si vous ne pouvez pas l'expliquer clairement, c'est qu'il n'est peut-être pas si bien écrit que cela.

Plusieurs expériences ont été développées sur dBASE II, comme dans d'autres langages. Empruntez ces expériences jusqu'à ce que vous développiez votre propre style.

Commencez chaque programme avec une ligne de commentaire qui inclut : le nom du programme, les initiales du programmeur, et la date de la dernière modification.

Entrez toutes les commandes dBASE II en majuscules, et les noms de variables en minuscules.

Décalez toutes les instructions dans une boucle DO ou dans une structure IF... ENDIF.

Utilisez des noms de variables standard pour contrôler les boucles et les entrées de données (ACCEPT INPUT).

Laissez des lignes vierges entre des groupes logiques de commandes.

Entrez des commentaires lorsque vous commencez une nouvelle fonction.

METTRE A JOUR UN DOSSIER DE DOCUMENTATION

Lorsque vous commencez à écrire de beaux programmes bien documentés, c'est une bonne habitude que de tenir un dossier qui contient la dernière version de chaque programme. La méthode la plus simple est d'imprimer chaque semaine, une copie de chaque programme nouvellement modifié ou créé. Le dossier contiendra également des copies de la structure de chaque fichier de base de données et également de chaque document que le système peut produire. A la première occasion où vous aurez à faire une modification sur le système, vous apprécierez d'avoir déjà fait une grande partie du travail.

COMMENCER TRES TOT A REDIGER UN MANUEL UTILISATEUR

C'est la partie de tout système informatique toujours faite en dernier. Comme la plupart des projets de programme sont toujours en retard, le manuel utilisateur vient toujours longtemps après que le système soit installé.

Ce manuel contiendra des exemples de tous les menus, une description de chaque fonction du système. Des exemples de documents imprimés seront particulièrement utiles. A moins que vous souhaitiez expliquer personnellement chaque fonction à chaque nouvel utilisateur du système, vous pourriez essayer de terminer le manuel utilisateur en même temps que le programme.

CHAPITRE VI

CREATION DE LA BASE DE DONNEES ET DES FICHIERS ECRAN

ALLUMEZ VOTRE ORDINATEUR

Il est temps de commencer à se servir de dBASE II, et de mettre en place le fichier DBF et les fichiers d'index pour notre programme de mailing.

```
A>dBASE

. CREATE

DONNEZ LE NOM DU FICHIER : AMICALE

DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP   NOM,TYPE,DIM,DECIMALE(S)
 001     ID:NO,N,5
 002     PRENOM,C,10
 003     NOM,C,10
 004     ADRESSE,C,20
 005     VILLE,C,10
 006     DEPT,C,2
 007     COD:POST,C,5
 008     TEL,C,14
 009     PROMOTION,C,2
 010     TOT:COTIS,N,10,2
 011     NOUVEAU,L,1
 012     <CR>
VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ? N

. USE AMICALE
```

. **APPEND**

ENREGISTREMENT 00001

ID:NO : **1**
PRENOM : **ALFRED**
NOM : **CHAMOUSSET**
ADRESSE : **20 RUE DES PISTES**
VILLE : **CHAMBERY**
DEPT : **73**
CODEPOST : **73000**
TEL : **79-69-30-40**
PROMOTION : **80**
TOT:COTIS : **100**
NOUVEAU : **T**

ENREGISTREMENT 00002

ID:NO : **2**
PRENOM : **GERARD**
NOM : **LETOURNEUR**
ADRESSE : **24 AVENUE V. HUGO**
VILLE : **CORMELLE**
DEPT : **14**
CODEPOST : **14000**
TEL : **31-84-26-40**
PROMOTION : **68**
TOT:COTIS : **300**
NOUVEAU : **T**

ENREGISTREMENT 00003

ID:NO : **3**
PRENOM : **DIEGO**
NOM : **MARTIN**
ADRESSE : **151 RUE DES PEUPLIERS**
VILLE : **ROUBAIX**
DEPT : **59**
CODEPOST : **59057**
TEL : **20-27-40-12**
PROMOTION : **80**
TOT:COTIS : **30**
NOUVEAU : **T**

```
ENREGISTREMENT 00004

ID:NO      : 4
PRENOM     : DOMINIQUE
NOM        : LEMAIRE
ADRESSE    : 47, RUE DE L'ASSOMPTION
VILLE      : MONTBELLIARD
DEPT       : 25
CODEPOST   : 25200
TEL        : 81-24-36-12
PROMOTION  : 69
TOT:COTIS  : 0
NOUVEAU    : T

ENREGISTREMENT 00005

ID:NO      : <CR>

. INDEX ON !(NOM) TO NOM
. INDEX ON CODEPOST TO CP
. INDEX ON PROMOTION TO ANNEE
```

La première rubrique Nom a été indexée sur !(NOM). De cette façon, on stocke le contenu de NOM dans le fichier d'index avec toutes les lettres en majuscules. dBASE II traite les lettres minuscules de façon distincte des lettres majuscules. Si un nom est entré comme Smith, FIND SMITH ne fonctionnera pas. En créant un index sur une version majuscule de NOM, la recherche avec une clé en majuscules par la commande FIND retrouvera toujours l'élève que l'on cherche.

CONVENTIONS POUR LES FORMATS DE DOCUMENTS

Tous les formats dans les programmes ont été créés avec ZIP comme des fichiers FMT. Le code qui suit chaque écran peut être saisi à la main, si vous n'avez pas une copie de l'utilitaire ZIP. Plusieurs conventions décrivant les formats ont été définies par ZIP. Une variable qui est utilisée dans une commande @ SAY est précédée par « @ », et « # » définit une variable pour un @ GET.

Pour rendre les écrans faciles à lire, toutes les variables sont indiquées en caractères gras.

CREATION DE FORMATS POUR TOUS USAGES

Un format doit être aussi généralisé que possible. Ceci permettra à un simple fichier FMT d'être utilisé par de nombreux programmes. Si l'on veut faire des modifications dans les programmes, une modification du fichier FMT affectera le reste du système.

Plusieurs noms de variables standard sont utilisés dans les formats. La variable MODE nous montre quel programme est exécuté. EFFACE montre lorsque l'enregistrement courant a été marqué pour effacement. MESSAGE1, MESSAGE2, et MESSAGE3 affichent des messages. Si le message doit être modifié, les variables peuvent être facilement redéfinies dans les programmes. COMMANDE est une variable standard, qui sera utilisée dans tous les programmes pour obtenir (GET) la nouvelle fonction à exécuter.

UTILISATION DES NOMS DE RUBRIQUES DANS LES FORMATS

Le nom de rubrique est utilisé dans une instruction @ **SAY**, dBASE II affichera la valeur (SAY) de cette rubrique issue de l'enregistrement courant. Dans une instruction @ **GET**, la valeur de l'enregistrement courant sera placée dans le format, et vous pourrez la modifier. Quelle que soit la façon dont on quitte le format après l'exécution de la commande READ, les informations seront écrites dans le fichier.

Une technique de programmation classique avec dBASE II consiste à ranger le contenu d'une rubrique (STORE) dans une variable mémoire, et de saisir la variable par la commande GET. Lorsque le READ est terminé, la variable mémoire est replacée (REPLACE) dans la rubrique. Cette technique permet au programmeur d'examiner, et éventuellement de rejeter, la donnée avant de la placer dans le fichier. Pour permettre une lecture claire du programme, la variable mémoire utilisée dans la commande @ **GET** est habituellement désignée par la lettre m suivie du nom de la rubrique.

LE FICHIER FMT UTILISE DANS LES EXEMPLES DE PROGRAMMES

SAISIE.FMT est utilisé pour ajouter ou modifier des enregistrements. Ce format utilise des variables mémoire pour les fonctions @ **GET**. Ce format est également utilisé dans AJOUT.CMD et MODIF.CMD.

ENTRET.FMT est le format du menu pour ENTRET.CMD.

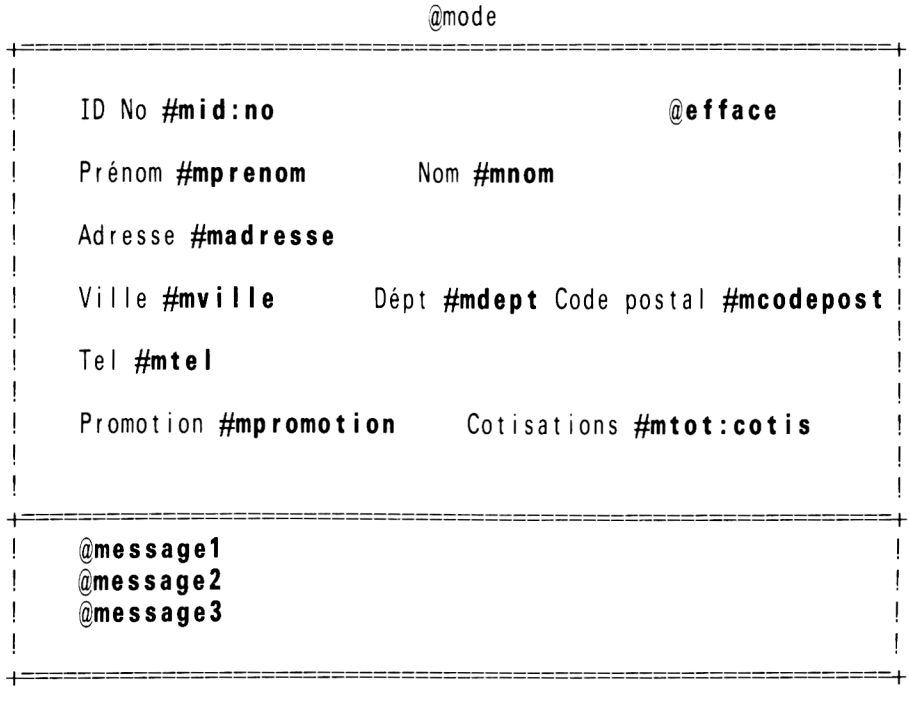
DOCUMENT.FMT est le format de menu pour DOCUMENT.CMD.

AFFICHE.FMT est utilisé pour afficher l'enregistrement courant. Les vrais noms de rubriques sont utilisés dans les instructions @ **SAY**. C'est le cas dans MAITRE.CMD, VERIFSUP.CMD et VERIFNVX.CMD.

DOUBLONS.FMT affiche deux enregistrements à la fois; il sert à comparer les enregistrements en double dans SUPRDBL.CMD.

SAISIE.FMT

Image-écran :



Code dBASE II :

```
@  1,27 SAY mode
@  2, 5 SAY "+=================================================+"
@  3, 5 SAY "!                                                 !"
@  4, 5 SAY "!    ID N°"        @  4,19 GET mid:no
@  4,51 SAY efface
@  4,63 SAY "!"
@  5, 5 SAY "!                                                 !"
@  6, 5 SAY "!    Prénom"
@  6,15 GET mprenom
@  6,33 SAY "Nom"
@  6,38 GET mnom
@  6,63 SAY "!"
@  7, 5 SAY "!                                                 !"
@  8, 5 SAY "!    Adresse"
@  8,17 GET madresse
@  8,63 SAY "!"
@  9, 5 SAY "!                                                 !"
@ 10, 5 SAY "!    Ville"
@ 10,14 GET mville
@ 10,29 SAY "Dépt"
@ 10,35 GET mdept
@ 10,45 SAY "Code postal"
@ 10,49 GET mcodepost
@ 10,63 SAY "!"
@ 11, 5 SAY "!                                                 !"
@ 12, 5 SAY "!    Tél"
@ 12,15 GET mtel
@ 12,63 SAY "!"
@ 13, 5 SAY "!                                                 !"
@ 14, 5 SAY "!    Promotion"
@ 14,24 GET mpromotion
@ 14,36 SAY "Cotisations"
@ 14,49 GET mtot:cotis
@ 14,63 SAY "!"
@ 15, 5 SAY "!                                                 !"
@ 16, 5 SAY "!                                                 !"
@ 17, 5 SAY "+=================================================+"
@ 18, 5 SAY "!"
@ 18, 8 SAY message1
```

```
@ 18,63 SAY "!"
@ 19, 5 SAY "!"
@ 19, 8 SAY message2
@ 19,63 SAY "!"
@ 20, 5 SAY "!"
@ 20, 8 SAY message3
@ 20,63 SAY "!"
@ 21, 5 SAY "!"
@ 22, 5 SAY "+==========================================================+"
```

ENTRET.FMT

Image-écran :

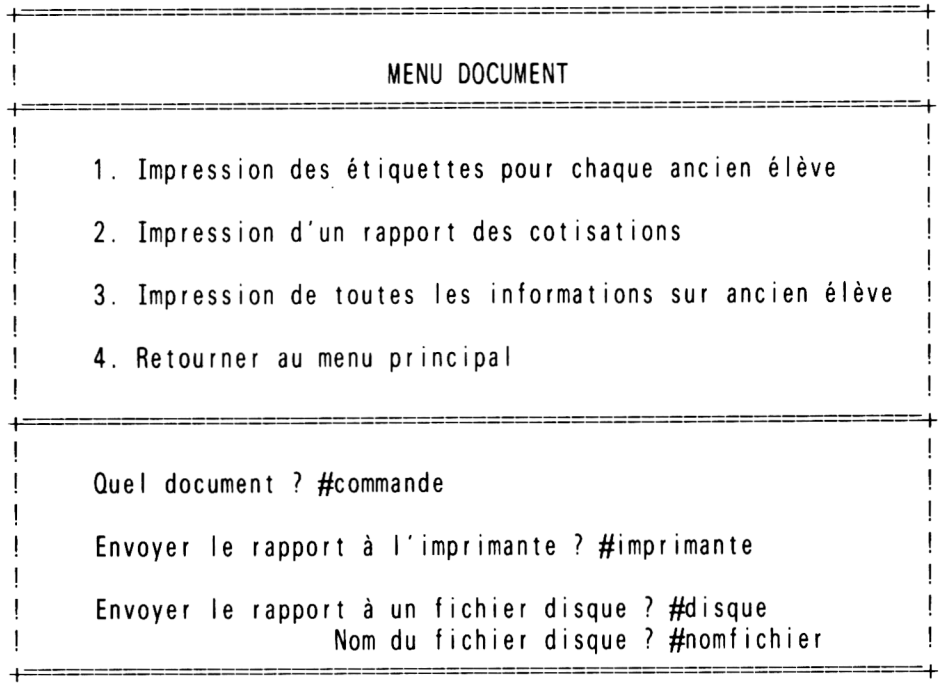
```
+============================================================+
!                                                            !
!                  MENU DE MISE A JOUR FICHIER               !
+============================================================+
!                                                            !
!     1. Trouver et supprimer les entrées doubles            !
!                                                            !
!     2. Vérifier les nouvelles entrées                      !
!                                                            !
!     3. Vérifier les entrées marquées pour effacement       !
!                                                            !
!     4. Supprimer les entrées marquées pour effacement      !
!                                                            !
!     5. Entrer les valeurs par défaut pour de nouvelles entrées !
!                                                            !
!     6. Sortie et appel de PIP.COM pour le backup des fichiers !
!                                                            !
!     7. Retourner au menu principal                         !
!                                                            !
!     S'il vous plait, choisissez une option #commande       !
!                                                            !
+============================================================+
```

Code dBASE II :

```
@  1, 5 SAY "+============================================================+"
@  2, 5 SAY "!                                                            !"
@  3, 5 SAY "!                MENU DE MISE A JOUR FICHIER                 !"
@  4, 5 SAY "+============================================================+"
@  5, 5 SAY "!                                                            !"
@  6, 5 SAY "!  1. Trouver et supprimer les entrées doubles               !"
@  7, 5 SAY "!                                                            !"
@  8, 5 SAY "!  2. Vérifier les nouvelles entrées                         !"
@  9, 5 SAY "!                                                            !"
@ 10, 5 SAY "!  3. Vérifier les entrées marquées pour effacement          !"
@ 11, 5 SAY "!                                                            !"
@ 12, 5 SAY "!  4. Supprimer les entrées marquées pour effacement         !"
@ 13, 5 SAY "!                                                            !"
@ 14, 5 SAY "!  5. Entrer les valeurs par défaut pour de nouvelles entrées !"
@ 15, 5 SAY "!                                                            !"
@ 16, 5 SAY "!  6. Sortie et appel de PIP.COM pour le backup des fichiers  !"
@ 17, 5 SAY "!                                                            !"
@ 18, 5 SAY "!  7. Retourner au menu principal                            !"
@ 19, 5 SAY "!                                                            !"
@ 20, 5 SAY "!  S'il vous plait, choisissez une option"
@ 20,43 GET commande
@ 20,64 SAY "!                                                            !"
@ 21, 5 SAY "!                                                            !"
@ 22, 5 SAY "+============================================================+"
```

DOCUMENT.FMT

Image-écran

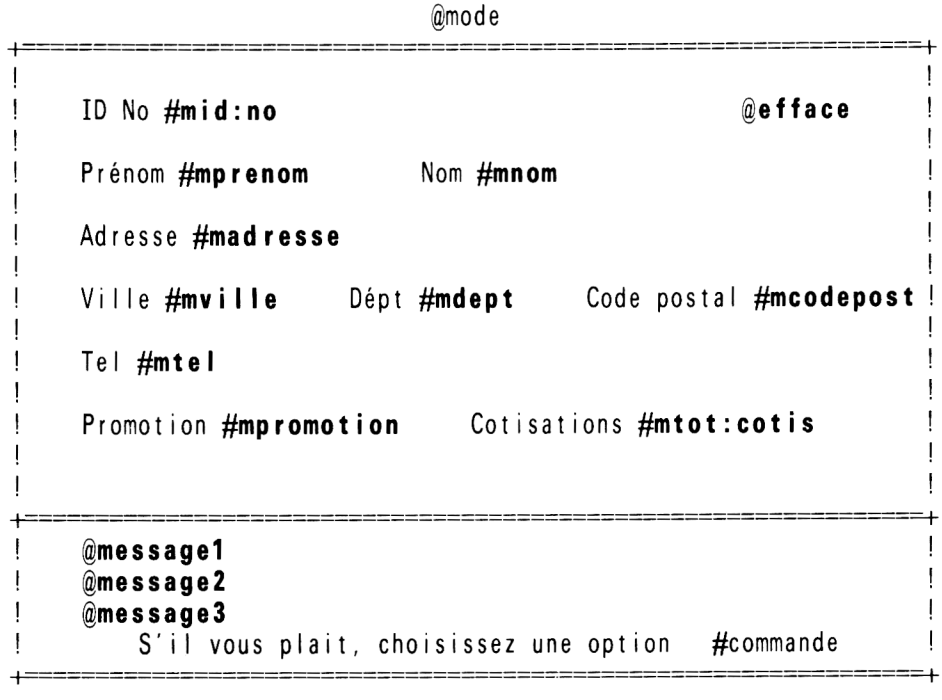


Code dBASE II :

```
@  1, 5 SAY "+============================================================+"
@  2, 5 SAY "!                                                            !"
@  3, 5 SAY "!                        MENU DOCUMENT                       !"
@  4, 5 SAY "+============================================================+"
@  5, 5 SAY "!                                                            !"
@  6, 5 SAY "!   1. Impression des étiquettes pour chaque ancien élève    !"
@  7, 5 SAY "!                                                            !"
@  8, 5 SAY "!   2. Impression d'un rapport des cotisations               !"
@  9, 5 SAY "!                                                            !"
@ 10, 5 SAY "!   3. Impression de toutes les information sur ancien élève !"
@ 11, 5 SAY "!                                                            !"
@ 12, 5 SAY "!   4. Retourner au menu principal                           !"
@ 13, 5 SAY "!                                                            !"
@ 14, 5 SAY "+============================================================+"
@ 15, 5 SAY "!                                                            !"
@ 16, 5 SAY "!   Quel document ?"
@ 16,32 GET commande
@ 16,64 SAY "!"
@ 17, 5 SAY "!                                                            !"
@ 18, 5 SAY "!   Envoyer le rapport à l'imprimante ?"
@ 18,42 GET imprimante
@ 18,64 SAY "!"
@ 19, 5 SAY "!"
@ 20, 5 SAY "!   Envoyer le rapport à un fichier disque ?"
@ 20,42 GET disque
@ 20,64 SAY "!"
@ 21, 5 SAY "!                        Nom du fichier disque ?"
@ 21,42 GET nomfichier
@ 21,64 SAY "!"
@ 22, 5 SAY "+============================================================!"
```

AFFICH.FMT

Image-écran :



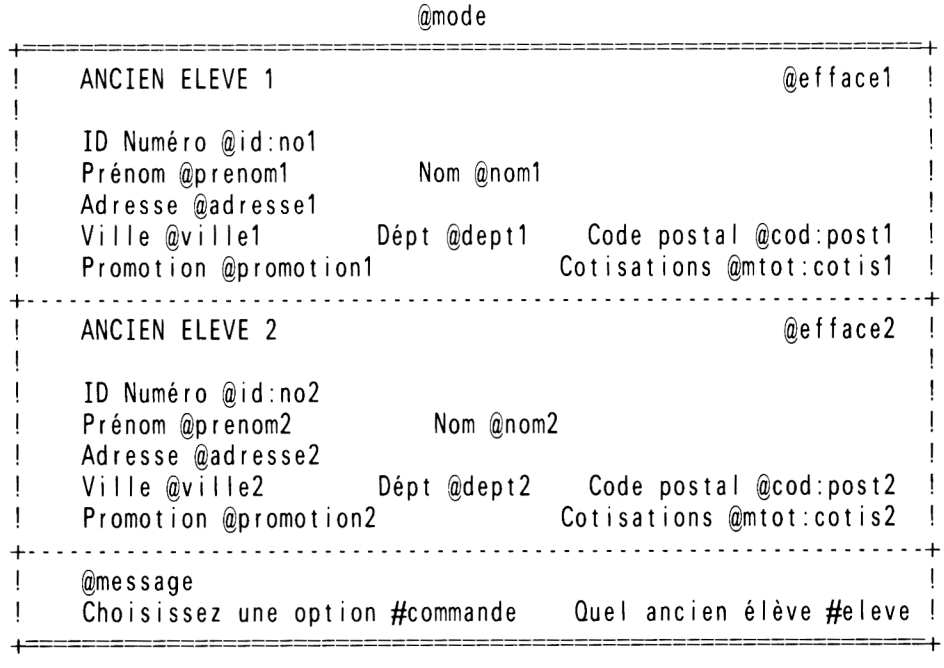
Code dBASE II :

```
@  1,27 SAY mode
@  2, 5 SAY "+================================================+"
@  3, 5 SAY "!                                                !"
@  4, 5 SAY "!    ID Numéro"
@  4,19 GET mid:no
@  4,51 SAY efface
@  4,63 SAY "!"
@  5, 5 SAY "!                                                !"
@  6, 5 SAY "!    Prénom"
@  6,15 GET mprenom
```

```
@  6,33 SAY "Nom"
@  6,38 GET mnom
@  6,63 SAY "!"
@  7, 5 SAY "!                                                          !"
@  8, 5 SAY "!    Adresse"
@  8,17 GET madresse
@  8,63 SAY "!"
@  9, 5 SAY "!                                                          !"
@ 10, 5 SAY "!    Ville"
@ 10,14 GET mville
@ 10,29 SAY "Dept"
@ 10,35 GET mdept
@ 10,45 SAY "Code postal"
@ 10,49 GET mcodepost
@ 10,63 SAY "!"
@ 11, 5 SAY "!                                                          !"
@ 12, 5 SAY "!    Tél"
@ 12,15 GET mtel
@ 12,63 SAY "!"
@ 13, 5 SAY "!                                                          !"
@ 14, 5 SAY "!    Promotion"
@ 14,24 GET mpromotion
@ 14,36 SAY "Cotisations"
@ 14,49 GET mtot:cotis
@ 14,63 SAY "!"
@ 15, 5 SAY "!                                                          !"
@ 16, 5 SAY "!                                                          !"
@ 17, 5 SAY "+==========================================================+"
@ 18, 5 SAY "!"
@ 18, 8 SAY message1
@ 18,63 SAY "!"
@ 19, 5 SAY "!"
@ 19, 8 SAY message2
@ 19,63 SAY "!"
@ 20, 5 SAY "!"
@ 20, 8 SAY message3
@ 20,63 SAY "!"
@ 21, 5 SAY "!          S'il vous plait, choisissez une option"
@ 21,52 GET commande
@ 21,63 SAY "!"
@ 22, 5 SAY "+==========================================================+"
```

DOUBLONS.FMT

Image écran :



Code dBASE II :

```
@  1,27 SAY mode
@  2, 5 SAY "+=============================================================+"
@  3, 5 SAY "!   ELEVE 1"
@  3,51 SAY efface1
@  3,63 SAY "!"
@  4, 5 SAY "!                                                             !"
@  5, 5 SAY "!   Id no"
@  5,19 SAY id:no1
@  5,63 SAY "!"
@  6, 5 SAY "!   Prénom"
```

```
@  6,15 SAY prenom1
@  6,32 SAY "Nom"
@  6,37 SAY nom1
@  6,63 SAY "!"
@  7, 5 SAY "!   Adresse"
@  7,17 SAY adresse1
@  7,63 SAY "!"
@  8, 5 SAY "!   Ville"
@  8,14 SAY ville1
@  8,30 SAY "Dépt"
@  8,36 SAY dept1
@  8,46 SAY "Code postal"
@  8,50 SAY cod:post1
@  8,63 SAY "!"
@  9, 5 SAY "!   Promotion"
@  9,24 SAY promotion1
@  9,36 SAY "Cotisations"
@  9,50 SAY tot:cotis1
@  9,63 SAY "!"
@ 10, 5 SAY "+=============================================================+"
@ 11, 5 SAY "!   ELEVE 2"
@ 11,51 SAY Efface2
@ 11,63 SAY "!"
@ 12, 5 SAY "!                                                             !"
@ 13, 5 SAY "!   ID No"
@ 13,19 SAY id:no2
@ 13,63 SAY "!"
@ 14, 5 SAY "!   Prénom"
@ 14,15 SAY prenom2
@ 14,32 SAY "Nom"
@ 14,37 SAY nom2
@ 14,63 SAY "!"
@ 15, 5 SAY "!   Adresse"
@ 15,17 SAY adresse2
@ 15,63 SAY "!"
@ 16, 5 SAY "!   Ville"
@ 16,14 SAY ville2
@ 16,30 SAY "Dépt"
@ 16,36 SAY dept2
@ 16,46 SAY "Code postal"
@ 16,50 SAY cod:post2
```

```
@ 16,63 SAY "!"
@ 17, 5 SAY "!   Promotion"
@ 17,24 SAY promotion2
@ 17,36 SAY "Cotisations"
@ 17,49 SAY tot:cotis2
@ 17,63 SAY "!"
@ 18, 5 SAY "+-------------------------------------------------------------+"
@ 19, 5 SAY "!"
@ 19, 9 SAY message
@ 19,63 SAY "!"
@ 20, 5 SAY "!   Choisissez une option"
@ 20,27 GET commande
@ 20,40 SAY "Quel élève"
@ 20,53 GET eleve
@ 20,63 SAY "!"
@ 21, 5 SAY "+=============================================================+"
```

CHAPITRE VII

PROGRAMME POUR UNE LISTE DE MAILING UTILISANT UN SEUL FICHIER

MAITRE.CMD

PSEUDO-CODE :

Afficher le message de l'auteur (AUTEUR)
Initialiser les variables et utiliser le fichier (INIT)
Créer la boucle principale du programme
    Contrôler le témoin d'effacement de l'enregistrement courant (TESTSUPR)
    Afficher l'enregistrement courant et obtenir l'option suivante
    Faire une des actions suivantes :
        Ajouter un enregistrement (AJOUT)
        Reculer d'un enregistrement
        Effacer l'enregistrement courant ou le rappeler (EFFACE)
        Modifier l'enregistrement courant (MODIF)
        Avancer d'un enregistrement
        Afficher un écran d'explication (AIDE)
        Présenter le menu du fichier d'entretien (ENTRET)
        Imprimer l'enregistrement courant (IMPRIMER)
        Sortir vers CP/M
        Présenter le menu des documents (DOCUMENT)
        Chercher un enregistrement par son nom (CHERCHE)
    Boucler et recommencer

FICHIER FMT UTILISE : AFFICH.FMT

VARIABLES LOCALES : MESSAGE1, MESSAGE2, MESSAGE3, MODE et COMMANDE

APPELE PAR : Lancé par CP/M avec : DBASE MAITRE
Lancé à partir de dBASE II avec : DO MAITRE

CODE DBASE II :

```
* maitre.cmd
* programme principal du système de mailing des anciens élèves

* affichage du message de l'auteur pendant l'initialisation du système
DO auteur
* initialisation des variables, mise en place de l'environnement, utilisation
  des fichiers, etc.
DO init
* mise en place de la boucle
DO WHILE t

   * mise en place de l'écran et des messages
   SET FORMAT TO affich
   STORE "A)jout, R)evenir, E)fface/Rappel, M)odif" TO message1
   STORE "P)lus loin, S)ecours, N)ettoyage, I)mpression, TO message2
   STORE "Q)uitter dBase et revenir @ CP/M, D)ocuments, C)herche
         TO message3
   STORE "Menu principal" TO mode
   STORE " " TO commande
   * chercher si l'enregistrement courant est marqué pour effacement
   DO testsupr

   * supprimer l'enregistrement courant, et prendre en compte la réponse
    READ

   STORE !(COMMANDE) TO COMMANDE

   * exécuter la fonction choisie
   DO CASE

       CASE commande = "A"
       DO ajout
```

```
      CASE commande = "R"
      * se déplacer en arrière d'un enregistrement
      SKIP -1

      CASE commande = "E"
      * supprimer le marquage d'effacement
      DO efface

      CASE commande = "M"
      DO modif

      CASE commande = "P"
      * se déplacer en avant d'un enregistrement
      SKIP

      CASE commande = "S"
      DO aide

      CASE commande = "N"
      Do entret

      CASE commande = "I"
      DO imprimer

      CASE commande = "Q"
      ERASE
      * prévenir les messages propres à dBASE II
      SET CONSOLE OFF
      QUIT

      CASE commande = "D"
      DO document

      CASE commande = "C"
      DO cherche

   ENDCASE

* retour de la boucle
ENDDO
```

AUTEUR.CMD

PSEUDO-CODE :

Effacer l'écran
Afficher une page de texte

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* auteur.cmd
ERASE
?
?
?
?
?
?
?
?
? "                    Système de mailing des anciens élèves"
?
?
?
?
?
? "                    par Adam B. Green"
```

INIT.CMD

PSEUDO-CODE :

Définir l'environnement de travail du système
Créer les fichiers mémoires nécessaires
Utiliser les fichiers avec leurs propres index

VARIABLES LOCALES : MID :NO, MPRENOM, MNOM, MADRESSE, MVILLE, MDEPT, MCOD :POST, MTEL, MPROMOTION, MTOT :COTIS

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* init.cmd
* ce programme est appelé une seule fois, au démarrage du programme
* maitre.cmd. Il initialisera les variables, mettra en place
* l'environnement, etc.
* placez ici toutes les commandes qui affecteront l'ensemble du système.

* définir l'environnement avec les commandes set
* indiquer le choix de luminosité, bip, validation, etc
SET TALK OFF
SET INTENSITY OFF

* vérifier l'existence du fichier ajout.mem
* s'il n'existe pas, réinitialiser les variables mémoire
* puis les sauvegarder dans un fichier mem, et effacer la mémoire
IF .NOT. FILE ("Ajout.mem" )
    STORE " " TO efface
    STORE 0 TO mid:no
    STORE "          " TO mprenom
    STORE "          " TO mnom
    STORE "                    " TO adresse
    STORE "          " TO mville
    STORE "  " TO mdept
    STORE "     " TO mcod:post
    STORE "               " TO mtel
    STORE "  " TO mpromotion
    STORE 0.00 TO mtot:cotis
    SAVE TO ajout
    RELEASE ALL
ENDIF

* mise en place du fichier des index
 USE amicale INDEX nom, cp, année
```

TESTSUPR.CMD

NOUVELLES TECHNIQUES :

Un enregistrement marqué pour effacement, a un astérisque placé avant la première rubrique. En testant l'existence d'un astérisque, un programme peut savoir si l'enregistrement a été effacé. Alors, l'instruction IF *, est vraie pour les enregistrements effacés.

PSEUDO-CODE :

Tester l'enregistrement courant pour effacement
Envoyer en retour le résultat dans la variable appelée efface

VARIABLE LOCALE : EFFACE

APPELE PAR : MAITRE.CMD, SUPRDBL.CMD, VERIFSUP.CMD, VERIFNVX.CMD

CODE dBASE II :

```
* testsupr.cmd
* si l'enregistrement en cours est marqué pour effacement
* alors stocker le message dans la variable pour affichage
* sinon, la variable efface est nulle
IF *
   STORE "Efface" TO efface
ELSE
    STORE " " TO efface
ENDIF
```

AJOUT.CMD

NOUVELLES TECHNIQUES :

Lorsqu'on utilise la commande READ pour entrer les données dans un fichier, il est préférable de créer un jeu temporaire de variables pour les utiliser avec un fichier format FMT. La donnée est alors lue dans ces variables, et est ensuite placée dans un nouvel

enregistrement, créé avec la commande APPEND BLANK. Cela permettra de modifier et de valider la donnée avant de la placer dans le fichier.

Les variables vides ont été créées par INIT.CMD, et stockées dans AJOUT.MEM.

PSEUDO-CODE :

Mise en place de l'écran pour l'ajout d'enregistrements
Mise en place d'une boucle jusqu'à la fin du traitement par l'utilisateur
    Obtenir un nouveau jeu de variables mémoire pour les entrées de données
    Afficher l'écran et obtenir les nouvelles données
    Si un numéro ID ou un nom a été entré
        Ajouter un enregistrement vide au fichier de données
        Placer les données entrées dans l'enregistrement
        Marquer l'enregistrement comme étant une nouvelle entrée
    Si le numéro d'identification (ID) ou le nom a été entré
        Mettre en place la boucle pour sortir au prochain passage
    Reboucler
    Nettoyer toutes les variables locales de la mémoire
    Remettre la mémoire dans son état initial

FICHIER FMT UTILISE : SAISIE.FMT

VARIABLES LOCALES : EFFACE, MID :NO, MPRENOM, MNOM, MADRESSE, MVILLE, MDEPT, MCOD :POST, MTEL, MPROMOTION, MTOT :COTIS

MODE, MESSAGE1, MESSAGE2, MESSAGE3, ENCORE

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* ajout.cmd
* ce programme ajoutera les enregistrements au fichier en cours
* les enregistrements seront marqués comme nouveaux
* pour une vérification et une fusion ultérieures

* mise en place de l'écran pour la saisie
SET FORMAT TO saisie
```

```
* boucler tant que l'opération d'ajout d'enregistrements n'est pas terminée
STORE t TO encore
DO WHILE encore

   * Obtenir un nouveau jeu de variables mémoire pour la saisie
   RESTORE from ajout
   STORE "Ajouter un nouveau nom et une adresse" TO mode
   STORE "Entrer autant de noms que vous voulez." TO message1
   STORE "Lorsque c'est fait, placer un espace dans ID et valider toutes
     les zones" TO message2
   STORE " " TO message3
   STORE t TO encore

   * Laissons l'utilisateur entrer les données
   READ

   * placer une routine de vérification des données à cet endroit
   * par exemple, vous pourriez vous assurer de la saisie du code postal

   * Ajouter un nouvel enregistrement, si ID et NOM comportent des
     informations
   IF mid:no <> 0 .OR. mnom <> " "

       * ajouter un nouvel enregistrement au fichier
       APPEND BLANK

       * remplir avec les nouvelles données
       REPLACE id:no WITH mid:no, prénom WITH mprénom, nom WITH mnom,
       REPLACE adresse WITH madresse, ville WITH mville,
       REPLACE dept WITH mdept, cod:post WITH mcod:post, tel WITH mtel
       REPLACE promotion WITH mpromotion, tot:cotis WITH mtot:cotis

       * marquer l'enregistrement comme nouveau
       REPLACE nouveau WITH t

   ELSE
       * il n'y a pas d'autres enregistrements à ajouter
       * mise en place de la boucle de fin
       STORE f TO encore

   ENDIF
```

```
* reboucler
ENDDO

* supprimer les variables locales
RELEASE ENCORE

* remettre les variables initiales
RESTORE FROM AJOUT
```

EFFACE.CMD

PSEUDO-CODE :

Tester l'enregistrement en court pour effacement
S'il a été effacé, rappelez-le
Autrement effacez-le

APPELE PAR : MAITRE.CMD, SUPRDBL.CMD, VERIFSUP.CMD, VERIFNVX.CMD

CODE dBASE II :

```
* EFFACE.CMD

IF *
    RECALL
ELSE
    DELETE
ENDIF
```

MODIF.CMD

NOUVELLES TECHNIQUES :

Modifier les données d'un enregistrement avec la commande READ ressemble à l'ajout de nouvelles données, pour cela la donnée en cours sera stockée (STORE) dans des variables temporaires. On peut afficher et modifier la donnée avec la commande GET. Une fois la nouvelle donnée saisie, le programme doit pouvoir modifier ou contrôler la donnée dans l'enregistrement avant de la replacer (REPLACE).

Cette technique n'est pas obligatoire. Un fichier FMT peut être utilisé avec les variables de la rubrique actuelle, mais aucune validation ne sera possible.

PSEUDO-CODE :

Ranger les données de l'enregistrement en cours dans des variables temporaires
Mise en place de l'écran pour la saisie des données
Laissons l'utilisateur modifier la donnée
Exécution d'un contrôle de vraisemblance sur les nouvelles données
Placer la nouvelle donnée dans l'enregistrement en cours
Effacer toutes les variables locales
Mettre la mémoire dans l'état initial

FICHIER FMT UTILISE : SAISIE.FMT

VARIABLES LOCALES : MID :NO, MPRENOM, MNOM, MADRESSE, MVILLE, MDEPT, MCOD :POST, MTEL, MPROMOTION, MTOT :COTIS

MODE, MESSAGE1, MESSAGE2, MESSAGE3

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* modif.cmd
* ce programme modifiera l'enregistrement en cours

* sauvegarder sur disque les variables mémoire en cours
 SAVE TO temp

* sauvegarder les variables rubrique dans des variables mem pour modification
STORE id:no TO mid:no
STORE prénom TO mprenom
STORE nom TO mnom
STORE adresse TO madresse
STORE ville TO mville
STORE dept TO mdept
STORE tel TO mtel
STORE promotion TO mpromotion
STORE tot:cotis TO mtot:cotis
```

```
* mise en place de l'écran et message de modification
SET FORMAT TO saisie
STORE "Modifier le nom et l'adresse" TO mode
STORE "Entrez le nouveau nom et l'adresse" TO message1
STORE " " TO message2
STORE " " TO message3

* laissons l'utilisateur entrer les données
READ

* placer une routine de contrôle de modification à cet endroit
* par exemple, vous pourriez vérifier la saisie du code postal

* replacer la donnée modifiée dans le même enregistrement
REPLACE id:no WITH id:no, prenom WITH mprenom
REPLACE nom WITH mnom, adresse WITH madresse
REPLACE ville WITH mville, dept WITH mdept
REPLACE cod:post WITH mcod:post, tel WITH mtel
REPLACE promotion WITH mpromotion, tot:cotis WITH mtot:cotis

* remettre la mémoire à l'état initial
RESTORE FROM temp
```

AIDE.CMD

PSEUDO-CODE :

Effacer l'écran
Afficher une page écran d'explications
Attendre la pression d'une touche
Revenir au programme appelant
APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* aide.cmd
* afficher une page écran d'explications
```

```
ERASE
?
?
? "                    S'il vous plait, contactez votre informaticien !"
?
?
?
?
? "                         Appuyez sur une touche pour continuer"

SET CONSOLE OFF
WAIT
SET CONSOLE ON
```

IMPRIMER.CMD

NOUVELLES TECHNIQUES :

Si l'indicateur d'imprimante est mis (SET PRINT ON), toutes les instructions @ SAY seront envoyées à l'imprimante. Si les variables SAY ont le même nom que les variables de rubrique, les contenus de l'enregistrement en cours seront imprimés. On peut se servir de cette technique pour imprimer l'enregistrement sous la forme d'un document formaté, tel que des factures ou des relances de paiement.

PSEUDO-CODE :

Sélectionner l'imprimante pour obtenir toutes les commandes @ SAY
Mise en place du format avec les variables de rubrique
Les envoyer à l'imprimante

APPELE PAR : MAITRE.CMD, SUPRDBL.CMD, VERIFNVX.CMD, VERIFSUP.CMD

CODE dBASE II :

```
* imprimer.cmd
* impression de l'enregistrement en cours sur l'imprimante du système

* affecter les commandes @ SAY pour l'imprimante
SET PRINT ON
```

```
* impression de l'enregistrement en cours
@  2, 5 SAY "Numéro ID"
@  2,17 SAY id:no
@  4, 5 SAY "Nom"
@  4,17 SAY prenom
@  4,32 SAY nom
@  6, 5 SAY "Adresse"
@  6,17 SAY adresse
@  8, 5 SAY "Ville"
@  8,17 SAY ville
@  8,32 SAY dept
@  8,42 SAY cod:post
@ 10, 5 SAY "Tel"
@ 10,17 SAY tel
@ 12, 5 SAY "Promotion"
@ 12,22 SAY promotion
@ 14, 5 SAY "Cotisations"
@ 14,22 SAY tot:cotis

* mettre l'imprimante hors fonction
SET PRINT OFF
```

CHERCHE.CMD

NOUVELLES TECHNIQUES :

Les variables utilisées dans la commande FIND peuvent être utilisées comme des macro. Le signe & doit précéder le nom de la variable.

La commande FIND distingue bien les lettres en minuscules des lettres en majuscules. Ce qui signifie que si l'on recherche SMITH, on ne le trouvera pas. Vous devez donc exiger que les noms soient saisis en majuscules. Ce n'est pas seulement une question d'esthétique, mais une nécessité si vous ne voulez pas que votre ordinateur devienne encore moins utile qu'un bloc de papier. Pour supprimer ce désagrément, la rubrique nom sera indexée en lettres majuscules avec la fonction !(variable). Ce qui aura pour effet de convertir en majuscules la clé de recherche de la commande FIND.

Lorsque dBASE II trouve la commande FIND, le pointeur d'enregistrement est déplacé vers l'enregistrement correspondant. Si aucun enregistrement ne correspond, le pointeur

d'enregistrement sera égal à 0. L'instruction # 0 est vraie lorsque l'enregistrement n'a pas été trouvé.

PSEUDO-CODE :

Obtenir le nom pour la clé de recherche
La convertir en majuscules
Essayer de trouver l'enregistrement correspondant
Il n'a pas été trouvé
    Le dire à l'utilisateur
Il a été trouvé
    C'est maintenant l'enregistrement en cours

VARIABLE LOCALE : NOM

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* cherche.cmd
* ce programme recherchera un enregistrement par le nom

* nettoyer l'écran et faire la demande du nom
ERASE

?
?
?
?
?
ACCEPT "S'il vous plait, tapez le nom pour la recherche" TO nom

* convertir le nom en majuscules pour la recherche
STORE !(nom) TO nom

* rechercher l'enregistrement correspondant
FIND &nom
* il n'est pas dans le fichier
IF # = 0
```

```
   * nettoyer l'écran et afficher que le nom n'a pas été trouvé
   ERASE
   ?
   ?
   ?
   ?
   ?
   ? nom, " n'est pas dans le fichier"
   ? "Appuyez sur une touche pour revenir au menu"

   SET CONSOLE OFF
   WAIT
   SET CONSOLE ON

ENDIF

* le pointeur d'enregistrement pointe sur le nom qui a été trouvé

* suppression de la variable locale
RELEASE nom
```

ENTRET.CMD

PSEUDO-CODE :

Mise en place de la boucle jusqu'à la fin du traitement par l'utilisateur
    Mise en place du menu écran pour les fonctions d'entretien du fichier
    Exécution de l'une des actions suivantes :
        Vérifier les doublons (TESTDBL)
        Vérifier les nouvelles entrées (VERIFNVX)
        Vérifier les enregistrements marqués pour effacement (VERIFSUP)
        Supprimer les enregistrements marqués pour effacement (PURGE)
        Ajouter des valeurs par défaut pour l'ajout d'enregistrement (MODIFVAR)
        Sauvegardes des fichiers de données (BACKUP)
        Quitter ce programme

Reboucler

Nettoyer toutes les variables locales

FICHIER FMT UTILISE : ENTRET.FMT

VARIABLE LOCALE : ENCORE

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* entret.cmd
* menu de fichier entretien

* mise en placre
DO WHILE encore

   * mise en place de l'écran
   SET FORMAT TO entret

   * que faire ensuite
   STORE " " TO commande
   READ

   * exécuter la fonction désirée
   DO CASE

       CASE commande = "1"
       * contrôle des doublons
       DO testdbl

       CASE commande = "2"
       * vérification des nouveaux enregistrements
       DO verifnvx

       CASE commande = "3"
       * vérification des enregistrements effacés
       DO verifsup

       CASE commande = "4"
       * supprimer les enregistrements marqués pour effacement
       DO purge
```

```
      CASE commande = "5"
      * ajouter de nouvelles valeurs aux variables
      DO modifvar

      CASE commande = "6"
      * sauvegarder le disque
      DO backup

      CASE commande = "7"
      * mise en place de la boucle pour sortir
      STORE f TO encore

   ENDCASE

* reboucler
ENDDO

* suppression de la variable locale
RELEASE encore
```

TESTDBL.CMD

NOUVELLES TECHNIQUES :

Pour trouver les doublons dans une rubrique, indexer le fichier sur cette rubrique, et lister le fichier complet. Les enregistrements qui correspondront aux rubriques indexées apparaîtront ensemble.

Le fichier de l'Amicale doit être contrôlé pour les homonymies sur les noms. Puisqu'il est indexé sur cette rubrique, les homonymes seront placés les uns à la suite des autres. Le programme se déplacera à travers tout le fichier dans l'ordre de l'index, comparant chaque enregistrement avec le suivant.

Si l'enregistrement en cours est le dernier du fichier, le pointeur d'enregistrement ne se déplacera pas sur la commande SKIP. Le programme doit contrôler la fin de fichier (EOF) lorsqu'il est en train de sauter, pour éviter de lire le même enregistrement deux fois de suite, et risquer de se prendre lui-même pour un homonyme.

PSEUDO-CODE :

Sauvegarder l'image mémoire en cours dans un fichier mem
Commencer au premier enregistrement
Boucler jusqu'à ce que l'utilisateur soit à la fin du fichier
    Afficher un message pendant la recherche
    la rubrique nom de l'enregistrement en cours pour comparaison
    Se déplacer à l'enregistrement suivant
    Si le nouveau nom est égal à l'ancien et tant que la fin de fichier n'est pas atteinte
        Traiter les deux enregistrements qui correspondent (SUPRDBL)
Boucler et répéter le traitement avec le nouvel enregistrement en cours
Restaurer l'image mémoire initiale

VARIABLES MEMOIRES : ENCORE, ANCNOM
APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* testdbl.cmd
* contrôle des enregistrements comportant des homonymes sur la rubrique nom

* Puisque ce programme utilise beaucoup de nouvelles variables
* sauvegarder celles qui se trouvent actuellement en mémoire, et les replacer
   à la fin du traitement
SAVE TO Temp

* commencer au début du fichier
GOTO TOP

* mise en place de la boucle jusqu'à la fin du traitement complet
* ou bien lorsque l'utilisateur en décidera
STORE t TO encore
DO WHILE encore .AND. .NOT. EOF

   * afficher quelque chose à lire pour l'utilisateur
   * tant que le programme recherche les doublons
   ERASE
   ?
   ?
   ?
   ?
   ?
   ? "                          Recherche des enregistrements identiques"
```

```
   * sauvegarder le nom courant, en majuscules, pour comparaison
   STORE !(nom) TO ancnom

   * se déplacer vers l'enregistrement suivant pour comparer
   SKIP

   * si un enregistrement correspondant est trouvé et si non fin de fichier
   * montrer les deux enregistrements à l'utilisateur, qui décidera
   IF ancnom = !(nom) .AND. .NOT. EOF
       DO suprdbl
   ENDIF

* reboucler et retester
ENDDO

* remettre la mémoire à l'état initial
RESTORE FROM temp
```

SUPRDBL.CMD

PSEUDO-CODE :

Revenir sur le premier des homonymes
Sauvegarder ses données pour affichage
Contrôler pour voir s'il a été marqué pour effacement (TESTSUPR)

Se déplacer plus avant sur le deuxième doublon
Sauvegarder ses données pour affichage
Contrôler pour voir s'il a été marqué pour effacement (TESTSUPR)

Traiter les deux homonymes jusqu'à ce que l'utilisateur ait fini
    Mettre en place l'écran pour les afficher
    Afficher les deux doublons et demander que faire ensuite
    Si le premier enregistrement a été traité
        Revenir au premier doublon
    Exécuter l'une des fonctions suivantes
        Aller voir d'autres doublons
        Effacer ou rappeler l'enregistrement en cours (EFFACE)
        Modifier l'enregistrement en cours (MODIF)
        Imprimer l'enregistrement en cours (IMPRIMER)
        Fin de consultation des doublons

S'il était nécessaire de traiter le premier doublon
Remettre le pointeur d'enregistrement sur le deuxième
Reboucler et traiter de nouveau les doublons

S'il n'y a pas d'autres doublons
Envoyer un message en retour à TESTDBL

FICHIER FMT UTILISE : DOUBLONS.FMT

VARIABLES LOCALES : ID :NO1, PRENOM1, NOM1, ADRESSE1, VILLE1, DEPT1, COD :POST1, TEL1, PROMOTION1, TOT :COTIS1, EFFACE1

ID :NO2, PRENOM2, NOM2, ADRESSE2, VILLE2, DEPT2, COD :POST2, TEL2, PROMOTION2, TOT :COTIS2, EFFACE2

MESSAGE, MODE, COMMANDE, RECORD, ENCORE, ELEVE

APPELE PAR : TESTDBL.CMD

CODE dBASE II :

```
* suprdbl.cmd
* ce programme permet de traiter les enregistrements identiques

* obtenir la première duplication de la donnée à l'affichage
SKIP -1
STORE id:no TO id:no1
STORE prenom TO prénom1
STORE nom TO nom1
STORE adresse TO adresse1
STORE ville TO ville1
STORE dept TO dept1
STORE cod:post TO cod:post1
STORE tel TO tel1
STORE promotion TO promotion1
STORE tot:cotis TO tot:cotis1
* chercher si l'enregistrement est marqué pour effacement
DO testsupr
STORE efface TO efface1
```

```
* revenir au deuxième enregistrement double et obtenir ses données
SKIP
STORE id:no TO id:no2
STORE prenom TO prénom2
STORE nom TO nom2
STORE adresse TO adresse2
STORE ville TO ville2
STORE dept TO dept2
STORE cod:post TO cod:post2
STORE tel TO tel2
STORE promotion TO promotion2
STORE tot:cotis TO tot:cotis2

* chercher si cet enregistrement est marqué pour effacement
DO testsupr
STORE efface TO efface2

* mise en place de la boucle
STORE t TO encore
DO WHILE encore

   * mise en place du format écran et des variables
   SET FORMAT TO DOUBLONS.FMT
   STORE "C)ontinuer, E)fface, E)dit, I)mprimer, S)ortie" TO message
   STORE "Suppression des enregistrements doubles" TO mode
   STORE " " TO commande
   STORE " " TO record

   * montrer les deux enregistrements et demander la suite du traitement
   READ

   * se déplacer vers l'enregistrement à traiter
   IF enregistrement = "1"
       SKIP -1
   ENDIF

   * traiter l'enregistrement
   DO CASE

       CASE commande = "C" .OR. commande = "S"
       * mise en place de la boucle pour sortir
       STORE f TO encore
```

```
        CASE commande = "E"
        DO efface

        CASE commande = "M"
        DO modif

        CASE commande = "I"
        DO imprimer

    ENDCASE

    * revenir à l'enregistrement correct
    IF record = "1"
        SKIP
    ENDIF

* reboucler
ENDDO

* si d'autres doublons sont trouvés
* mise en place de la boucle pour continuer dans TESTDBL
IF commande = "C"
   STORE t TO encore
ENDIF
```

VERIFNVX.CMD

NOUVELLES TECHNIQUES :

La commande LOCATE FOR est semblable à la commande FIND, sauf qu'elle déplace le pointeur d'enregistrement vers l'enregistrement qui correspond à la variable indiquée. On peut faire une recherche sur plusieurs rubriques, et les rubriques de recherche n'ont pas besoin d'être indexées. L'enregistrement suivant correspondant est trouvé avec la commande CONTINUE. Lorsque l'on ne trouve pas d'autres enregistrements correspondants, EOF passe à la valeur logique vraie. La commande LOCATE démarre toujours au début du fichier.

PSEUDO-CODE :

Commencer en examinant le premier nouvel enregistrement
Boucler jusqu'à la fin du traitement par l'utilisateur ou lorsque la fin du fichier a été atteinte
  Mise en place de l'écran pour afficher le nouvel enregistrement
  Contrôler l'enregistrement en cours pour effacement (TESTSUPR)
  L'afficher et demander que faire ensuite
  Faire ce qui suit
    Accepter l'enregistrement comme permanent
    Effacer ou rappeler l'enregistrement en cours (EFFACE)
    Modifier l'enregistrement en cours (MODIF)
    Imprimer l'enregistrement en cours (IMPRIMER)
    Fin de consultation des nouveaux enregistrements
    Continuer l'examen des nouveaux enregistrements
Reboucler
Vider les variables locales
Remettre la mémoire à l'état initial

FICHIER FMT UTILISE : AFFICH.FMT

VARIABLES LOCALES : ENCORE, MODE, MESSAGE1, MESSAGE2, MESSAGE3, COMMANDE

APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* verifnvx.cmd
* vérifier et placer les nouveaux enregistrements dans le fichier

* commencer l'examen des nouveaux enregistrements
LOCATE FOR nouveau

* mise en place de la boucle
STORE t TO encore
DO WHILE encore .AND. (.NOT. EOF)
```

```
   * mise en place de l'écran et des messages
   SET FORMAT TO affiche
   STORE "Vérification des nouveaux enregistrements" TO mode
   STORE "A)ccepter, C)ontinuer, E)ffacer Rappeler" TO message1
   STORE "M)odifier, I)mprimer, S)ortir" TO message2
   STORE " " TO message3
   STORE " " TO commande

   * contrôler pour voir si l'enregistrement est marqué pour effacement
   DO testsupr

   * montrer l'enregistrement en cours sur l'écran, saisir un caractère
   READ

   * traiter l'enregistrement
   DO CASE

       CASE commande = "A"
       * supprimer la marque d'un nouvel enregistrement
       REPLACE nouveau WITH f

       CASE commande = "E"
       DO efface

       CASE commande = "M"
       DO modif

       CASE commande = "I"
       DO imprimer

       CASE commande = "S"
       * mise en place de la boucle pour sortir
       STORE f TO encore

       CASE commande = "C"
       continue

   ENDCASE

* reboucler
ENDDO
```

```
* supprimer les variables locales
* RELEASE mode, message1, message2, message3

* remettre la mémoire dans son état initial
STORE t TO encore
```

VERIFSUP.CMD

PSEUDO-CODE :

Ce programme exécute les mêmes fonctions sur les enregistrements qui ont été marqués pour effacement, que le module VERIFNVX sur les nouveaux enregistrements.

FICHIER FMT UTILISE : AFFICH.FMT

VARIABLES LOCALES : ENCORE, MODE, MESSAGE1, MESSAGE2, MESSAGE3, COMMANDE

APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* verifsup.cmd
* vérifier les enregistrements marqués pour effacement

* commencer à examiner les enregistrements effacés
LOCATE FOR *

* mise en place de la boucle
STORE t TO encore
DO WHILE encore .AND. (.NOT. EOF)

   * mise en place de l'écran
   SET FORMAT TO affich
   STORE "Vérification des enregistrements effacés" TO mode
   STORE "C)ontinuer, M)odifier, E)ffacer/Rappeler, I)mprimer, S)ortir"
      TO message1
   STORE " " TO message2
   STORE " " TO message3
   STORE " " TO commande
```

```
   * contrôler si l'enregistrement courant est marqué pour effacement
   DO testsupr

   * le montrer sur l'écran
   READ

   * que faire ensuite
   DO CASE

       CASE commande = "E"
       DO efface

       CASE commande = "M"
       DO modif

       CASE commande = "I"
       DO imprimer

       CASE commande = "S"
       STORE f TO encore
       CASE commande = "C"
       CONTINUE
   ENDCASE

ENDDO

* supprimer les variables mémoire locales
* RELEASE mode, message1, message2, message3

* remettre la mémoire dans son état initial
STORE t TO encore
STORE "3" TO commande
```

PURGE.CMD

PSEUDO-CODE :

Effacement de l'écran
S'assurer que l'utilisateur désire supprimer les enregistrements marqués

S'il veut le faire
    Placer un message sur l'écran pendant la suppression
    Supprimer les enregistrements marqués pour effacement

VARIABLE LOCALE : SUIVANT

APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* purge.cmd
* suppression des enregistrements du fichier
* qui ont été marqués pour effacement

* s'assurer que l'utilisateur veut bien supprimer les enregistrements
ERASE
?
?
? "**********   ATTENTION !  ********  ATTENTION !  ******************"
?
? "Tous les enregistrements marqués seront définitivement effacés."
?
? "Tapez O pour continuer, n'importe quelle autre touche pour annuler"

SET CONSOLE OFF
WAIT TO suivant
SET CONSOLE ON

* s'il ne désire pas supprimer définitivement
IF !(suivant) = "O"

   * donner à l'utilisateur quelque chose à lire pendant que le fichier
       est compacté
   ERASE
   ?
   ?
   ?
   ? "Les enregistrements qui ont été marqués pour effacement"
   ? "sont supprimés du fichier"
```

```
   * compacter le fichier pour supprimer les enregistrements effacés
   PACK

ENDIF

* supprimer les variables locales
RELEASE suivant
```

MODIFVAR.CMD

NOUVELLES TECHNIQUES :

Les variables sont affichées avec la commande @ SAY et modifiées avec @ GET. Les valeurs par défaut pour la saisie des données peuvent être produites par un stockage de données dans des variables mémoire utilisées par le fichier AJOUT.CMD pour le fichier SAISIE.FMT. Lorsqu'un enregistrement est ajouté, les valeurs par défaut sont récupérées lorsque l'on presse la touche return, ou bien de nouvelles valeurs sont tapées pour remplacer celles par défaut.

PSEUDO-CODE :

Mettre les variables mémoire en cours dans un fichier temporaire
Obtenir les variables utilisées par AJOUT.CMD
Mise en place de l'écran pour entrer les nouvelles valeurs
Laisser l'utilisateur modifier ses propres variables
Sauvegarder ces nouvelles variables par défaut dans AJOUT.MEM
Restaurer les variables mémoire initiales

FICHIER FMT UTILISE : SAISIE.FMT

VARIABLES LOCALES : MID :NO, MPRENOM, MNOM, MADRESSE, MVILLE, MDEPT,
MCOD :POST, MTEL, MPROMOTION, MCOTISATIONS,
MESSAGE1, MESSAGE2, MESSAGE3, MODE

APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* modifvar.cmd
* ce qui permettra la saisie des variables par défaut dans AJOUT.CMD

* obtenir les anciennes variables mémoire pour l'ajout d'enregistrements
RESTORE FROM ajout

* mise en place de l'écran et des messages
SET FORMAT TO saisie
STORE "Entrez les données dans une rubrique" TO message1
STORE "Cette donnée sera utilisée lors de l'ajout des enregistrements"
      TO message2
STORE " " TO message 3
STORE "Création de nouveaux messages de données" TO mode

* obtenir les nouvelles valeurs par défaut
READ

* ranger les nouvelles variables
SAVE TO ajout

* remettre la mémoire à son état initial
STORE "5" TO commande
```

BACKUP.CMD

NOUVELLES TECHNIQUES :

La commande QUIT TO permet de sortir de dBASE II, de faire tourner un autre programme, et de retourner au programme dBASE II. Elle supprime toutes les explications supplémentaires que l'on serait obligé de fournir au nouvel utilisateur dans l'utilisation de CP/M.

PSEUDO-CODE :

Nettoyer l'écran
Demander si l'utilisateur veut sauvegarder les fichiers de données

S'il le veut
    Sortir de CP/M
    Exécuter PIP.COM
    Revenir à dBASE II et exécuter MAITRE.CMD

VARIABLE LOCALE : SUIVANT

APPELE PAR : ENTRET.CMD

CODE dBASE II :

```
* backup.cmd
* faire une copie de sauvegarde du disque

* nettoyer l'écran et s'assurer que l'utilisateur veut faire une sauvegarde
ERASE
?
?
?
?
?
? "Appuyer sur B pour un backup des données, n'importe quelle autre touche
   pour revenir au menu"

SET CONSOLE OFF
WAIT TO suivant
SET CONSOLE ON

* si l'utilisateur veut sauvegarder les fichiers
IF !(suivant) = "B"
    * sortir vers CP/M, exécuter PIP.COM, puis redémarrer un programme dBASE
    QUIT TO "pip", "dbase maitre"
ENDIF

* supprimer la variable locale
RELEASE SUIVANT
```

DOCUMENT.CMD

NOUVELLES TECHNIQUES :

Le texte qui a été affiché sur l'écran peut aussi être sauvegardé dans un fichier disque, identique au fichier texte créé avec un traitement de texte. La commande SET ALTERNATE TO crée un fichier pour recevoir le texte, et SET ALTERNATE ON envoie toutes les sorties vers ce fichier.

PSEUDO-CODE :

```
Mise en place d'une boucle qui se répètera jusqu'à la fin du traitement
    Mise en place du format de menu sur écran avec les messages
    Demander que faire ensuite
    Si le document va vers l'imprimante
        Allumer l'imprimante
    Si le document va vers un fichier disque et que l'on fournit un nom de fichier
        Envoyer toutes les sorties vers le fichier nommé
    Exécuter ce qui suit
        Impression des étiquettes (ETIQUET)
        Impression d'un document des cotisations (COTIS.FRM)
        Impression de chacune des sommes totales de cotisations de l'enregistrement
          (LISTING)
        Quitter le menu document
    Eteindre l'imprimante
    Stopper l'envoi en sortie vers un fichier disque
Reboucler
Supprimer les variables locales
Remettre la mémoire dans son état initial
Mise en place des index corrects
```

FICHIER FMT UTILISE : DOCUMENT.FMT

VARIABLES LOCALES : ENCORE, IMPRIMANTE, DISQUE, NOMFICHIER, COMMANDE

APPELE PAR : MAITRE.CMD

CODE dBASE II :

```
* document.cmd
* présenter le menu des documents imprimés

* mise en place de la boucle et répéter jusqu'à la fin du traitement
STORE t TO encore
DO WHILE encore

   * mise en place de l'écran et des variables mémoire
   SET FORMAT TO DOCUMENT
   STORE " " TO imprimante
   STORE " " TO disque
   STORE "              " TO nomfichier
   STORE " " TO commande

   * demander que faire ensuite
   READ

   * mise en place de l'imprimante si nécessaire
   IF imprimante = "O"
       SET PRINT ON
   ENDIF

   * envoyer le document sur un fichier texte si la réponse disque est oui,
   * et si un nom de fichier a été donné
   IF disque = "O" .AND. nomfich <> " "
       SET ALTERNATE TO &nomfich
       SET ALTERNATE ON
   ENDIF

   * effacement de l'écran pour le document
   ERASE

   * faire le document demandé
   DO CASE

       CASE commande = "1"
       * utiliser le fichier dans l'ordre des codes postaux
       USE amicale INDEX cod:post
       DO étiquet
```

```
      CASE commande = "2"
      * utiliser le fichier dans l'ordre des promotions
      USE amicale INDEX année
      REPORT FORM cotis

      CASE commande = "3"
      * utiliser le fichier dans l'ordre alphabétique des noms
      DO listing

      CASE commande = "4"
      STORE f TO encore

   ENDCASE
   * éteindre l'imprimante et l'envoi sur disque
   * s'ils sont éteints, cela n'aura aucun effet
   SET PRINT  OFF
   SET ALTERNATE OFF

ENDDO

* restauration des variables locales
RELEASE imprimante, disque, nomfichier

* récupérer les variables initiales
STORE t TO encore
STORE "R" TO commande

* utilisation du fichier avec tous les index
USE amicale INDEX nom,cod:post,année
```

ETIQUET.CMD

PSEUDO-CODE :

Mise en place d'une boucle pour imprimer les étiquettes pour l'ensemble du fichier
    Impression des étiquettes dans le format suivant
        Prénom Nom
        Adresse
        Ville, Dépt Cod :post
    Impression de 3 lignes vides pour sauter à l'étiquette suivante
    Aller à l'enregistrement suivant
Reboucler

APPELE PAR : DOCUMENT.CMD

CODE dBASE II :

```
* etiquet.cmd
* ce programme imprimera les étiquettes de mailing
* pour tous les anciens élèves

* mise en place de la boucle d'impression des étiquettes pour l'ensemble
  du fichier
DO WHILE .NOT. EOF

   * imprimer une étiquette
   ? TRIM(prénom), nom
   ? adresse
   ? TRIM(codepost) - "," , ville
   ?
   ?
   ?

   * aller à l'enregistrement suivant
   SKIP

* reboucler et imprimer l'étiquette suivante
ENDDO
```

LISTING.CMD

NOUVELLES TECHNIQUES :

Si un document doit posséder des sous-totaux basés sur les contenus d'une rubrique, le fichier sera alors utilisé (USE) dans l'ordre de son INDEX. Tous les enregistrements ayant la même valeur sur la rubrique indexée seront placés ensemble.

```
. REPORT FORM COTIS

ENTRER LES OPTIONS, M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR PAGE, M=0,W=75
DESIREZ-VOUS UN EN-TETE (Y/N) ? Y
DONNEZ L'EN-TETE : Etat des cotisations de l'Amicale
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? N
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
DESIREZ-VOUS DES SOUS-TOTAUX (Y/N) ? Y
DONNEZ LE CHAMP DE SOUS-TOTAL : ANNEE
DESIREZ-VOUS UNIQUEMENT UN RESUME DU RAPPORT (Y/N) ? N
CHANGEMENT DE PAGE APRES LES SOUS-TOTAUX (Y/N) ? N
ENTREZ LE SOUS-TITRE  POUR LES SOUS-TOTAUX : COTISATIONS POUR L'ANNEE
COLONNE LONGUEUR,CONTENU
001     5,ID:NO
ARE TOTALS REQUIRED? (Y/N) N
002     10,NOM
ENTER HEADING: NOMS
003     10,PRENOM
ENTER HEADING:
 004     10,VILLE
ENTER HEADING: VILLES
005     15,TOT:COTIS
ENTER HEADING: COTISATIONS
ARE TOTALS REQUIRED? (Y/N) Y
006     <CR>
```

AUTRDOC.CMD

NOUVELLES TECHNIQUES :

Imprimer un document à partir d'un programme, à la place de la commande REPORT, nécessite un calcul de la part du programmeur sur l'espace disponible sur la page de l'imprimante.

Haut de page

Titre du rapport

Données de l'enregistrement numéro 1

Données de l'enregistrement numéro 3

Données de l'enregistrement numéro n

Bas de page

Nombre d'enregistrements/page
= (longueur page — longueur haut de page)/nombre de lignes par page

PSEUDO-CODE :

```
Mise en place d'une boucle extérieure pour répéter le traitement jusqu'à la fin du fichier
    Aller en haut de la page suivante de l'imprimante
    Imprimer le titre du document
    Mise en place d'une boucle intérieure de façon à imprimer 15 enregistrements par
        page et s'arrêter si l'on atteint la fin du fichier
        Impression de l'enregistrement sur trois lignes
        Augmenter le nombre d'enregistrements imprimés sur cette page
        Aller à l'enregistrement suivant
    Reboucler
Reboucler
```

VARIABLE LOCALE : RECORD

APPELE PAR : DOCUMENT

CODE dBASE II :

```
*  listing.cmd
* format d'impression pour imprimer l'ensemble des données d'un enregistre-
  ment pour chaque ancien élève

* mise en place de la boucle extérieure de façon à ce qu'elle s'arrête après
  le dernier enregistrement
DO WHILE .NOT. EOF

* aller en haut de la page suivante
   EJECT

   * impression du titre
   ?
   ? "              Liste des membres de l'Amicale des anciens élèves
   ? "            ---------------------------------------------------"
   ?
   * mise en place de la boucle intérieure de façon à ce qu'elle imprime
   * 15 enregistrements par page
   * et s'arrête au dernier enregistrement
   STORE 0 TO record
   DO WHILE record < 15 .AND. (.NOT. EOF)

       * impression de l'enregistrement sur quatre lignes
       ? prénom, nom
       ? adresse, ville, dept, cod:post
       ? "NUMERO IDENTIFICATION", id:no, " ANNEE DE FIN D'ETUDE", année,
          tot:cotis
       ?

       * incrémentation du compteur d'enregistrement
       STORE record + 1 TO record

       * aller à l'enregistrement suivant
       SKIP

   * reboucler et imprimer un autre enregistrement
   ENDDO
```

```
* reboucler et imprimer une nouvelle page
ENDDO

* supprimer la variable locale
RELEASE record
```

SORTIE DES TROIS DOCUMENTS

ETIQUET.CMD :

```
ALFRED CHAMOUSSET
20, RUE DES PISTES
CHAMBERY, 73 73000

GERARD LETOURNEUR
24, AVENUE V. HUGO
CORMELLE, 14 14000

DIEGO MARTIN
151, RUE DES PEUPLIERS
ROUBAIX, 59 59057

DOMINIQUE LEMAIRE
47, RUE DE L'ASSOMPTION
MONTBELLIARD, 25 25200
```

COTIS.FRM :

```
PAGE NO. 00001

                    Etat des cotisations de l'Amicale

  ID          NOM                          VILLE                COTISATIONS

* COTISATIONS POUR L'ANNEE      68
   2 LETOURNEUR      GERARD          CORMELLE                        300.00
** SOUS-TOTAL **
                                                                    300.00
```

```
* COTISATIONS POUR L'ANNEE     69
   4 LEMAIRE        DOMINIQUE     MONTBELLIARD                    0.00
** SOUS-TOTAL **
                                                                   0.00

* COTISATIONS POUR L'ANNEE     80
   1 CHAMOUSSET     ALFRED        CHAMBERY                      100.00
   3 MARTIN         DIEGO         ROUBAIX                        30.00
** SOUS-TOTAL **
                                                                 130.00

** TOTAL **                                                      430.00
LISTING.CMD :

                  Liste des membres de l'Amicale
                 ---------------------------------

   CHAMOUSSET    ALFRED
   20 RUE DES PISTES             CHAMBERY         73 73000
   ID NUMERO    1    PROMOTION  80                          100.00

   LETOURNEUR    GERARD
   24 AVENUE V. HUGO             CORMELLE         14 14000
   ID NUMERO    2    PROMOTION  68                          300.00

   MARTIN        DIEGO
   151 RUE DES PEUPLIERS         ROUBAIX          59 59057
   ID NUMERO    3    PROMOTION  80                           30.00

   LEMAIRE       DOMINIQUE
   47 RUE DE L'ASSOMPTION        MONTBELLIARD     25 25200
   ID NUMERO    4    PROMOTION  69                            0.00
```

CHAPITRE VIII

RATTACHEMENT D'UN AUTRE FICHIER A AMICALE.DBF

Notre système de mailing est maintenant opérationnel et exécute les fonctions qui étaient prévues par le projet initial. L'étape suivante sera d'ajouter une possibilité de retrouver des cotisations individuelles. Les fichiers actuels contiennent l'ensemble des cotisations de chaque membre, mais à l'Université Story Book la Direction a besoin d'enregistrements plus détaillés. Ils veulent enregistrer chaque cotisation individuelle et faire des recoupements d'informations avec les enregistrements existants des cotisations globales des anciens élèves.

La solution la plus simple serait d'ajouter des rubriques à la base AMICALE.DBF pour gérer cette information. Trois rubriques sont nécessaires pour chaque cotisation : la date, le montant et les commentaires. Malheureusement, il n'est pas possible de prédire le nombre de cotisations faites par chaque élève. Certains ont cotisé dix fois, tandis que d'autres n'ont jamais versé un seul centime.

Si nous réservons de la place dans chaque enregistrement pour un nombre maximum de cotisations, nous gâcherons alors beaucoup d'espace. Une autre solution serait de réserver trente-deux rubriques par enregistrement. Mais puisqu'il y en a déjà onze de prises, il ne reste plus que sept groupes de rubriques pour les cotisations. Il doit donc y avoir une meilleure solution.

Ce type de problème est résolu avec dBASE II en triant les cotisations individuelles dans un fichier séparé et en rattachant ensuite ce fichier au premier fichier. Le nouveau fichier sera appelé COTIS.DBF et contiendra un enregistrement pour chaque cotisation. Cette méthode économisera de la place dans le fichier principal Amicale, et permettra également de traiter les anciens élèves les plus généreux.

Une fois ce nouveau fichier créé (CREATE), on pourra alors exécuter les recoupements souhaités sur le fichier Amicale. Chaque ancien élève a reçu un numéro d'identification

unique (ID). Si chaque cotisation est stockée avec son numéro de cotisant, on peut facilement écrire un programme pour rattacher ensemble les deux fichiers.

CREATION DE COTIS.DBF

Les nouvelles rubriques seront ajoutées en premier lieu au dictionnaire de données pour l'application de liste de mailing.

Nom	Type	Taille	Description
ID Numéro	N	5	Cette rubrique permet de rattacher chaque enregistrement à l'élève correspondant de AMICALE.DBF
Date	C	8	
Montant	N	10	
Comment	C	30	

Pour tester la technique de rattachement de deux fichiers (LINK), créons le fichier (CREATE) et entrons quelques données. Pour rester cohérent avec AMICALE.DBF, les totaux des exemples de cotisations dans COTIS.DBF seront égaux au total des cotisations déjà saisies pour chaque élève.

```
. CREATE COTIS

DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
 CHAMP   NOM,TYP,DIM,DECIMALE(S)
 001     ID:NO,N,5
 002     DATE,C,8
 003     MONTANT,N,10
 004     COMMENT,C,30
 005     <CR>

VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ? N

.USE COTIS

ENREGISTREMENT 00001

ID:NO     : 1
DATE      : 2/3/86
MONTANT   : 30
COMMENT   :
```

```
ENREGISTREMENT 00002

ID:NO     : 1
DATE      : 3/4/86
MONTANT   : 70
COMMENT   :

ENREGISTREMENT 00003

ID:NO     : 2
DATE      : 6/10/86
MONTANT   : 150
COMMENT   :

ENREGISTREMENT 00004

ID:NO     : 2
DATE      : 10/10/86
MONTANT   : 150
COMMENT   :

ENREGISTREMENT 00005

ID:NO     : 3
DATE      : 7/5/86
MONTANT   : 30
COMMENT   :

ENREGISTREMENT 00006

ID:NO     : <CR>

. INDEX ON ID:NO TO NO:COT

00005 ENREGISTREMENT(S) INDEXE(S)
```

Puisque COTIS.DBF sera lié à AMICALE.DBF sur le numéro d'identification (ID), il est judicieux d'utiliser le même nom de rubrique dans chaque fichier. Il est aussi nécessaire d'indexer ce nouveau fichier sur le numéro d'identification afin de permettre des accès rapides à toutes les cotisations qui correspondent.

Le nom de la rubrique index montre qu'elle est basée sur le numéro ID et qu'elle est destinée au fichier de cotisation.

RATTACHEMENT DE AMICALE.DBF AVEC COTIS.DBF

AMICALE.DBF est utilisé dans la première zone de mémoire (PRIMARY) et COTIS.DBF est utilisé (USE) avec l'index sur la rubrique numéro ID dans la zone secondaire (SECONDARY). Lorsqu'un élève est sélectionné pour un recoupement, le numéro ID est utilisé comme une clé pour trouver les enregistrements correspondants dans COTIS.DBF. Soyons plus clair avec un exemple.

```
. USE AMICALE
. SELECT SECONDARY
. USE COTIS INDEX NO:COT
. SELECT PRIMARY
. DISPLAY NOM,ID:NO
00001  LORIS              1
. SELECT SECONDARY
. FIND 1
. DISPLAY
00001         12/3/86          30.00
```

Voilà comment on peut trouver le premier enregistrement qui correspond. C'est aussi simple que cela puisque COTIS.DBF est dans l'ordre des index sur le numéro ID, tous les enregistrements avec le même numéro d'ID seront placés ensemble. Nous sautons simplement (SKIP) pour trouver le prochain enregistrement correspondant. C'est lorsque le numéro d'ID change que nous savons que toutes les cotisations correspondantes ont été trouvées.

```
. SKIP

ENREGISTREMENT  00002

00002        13/4/86          70.00

ENREGISTREMENT  00003

. DISPLAY

00003        26/10/86         150.00
```

Avant d'écrire un programme pour rattacher des fichiers, découvrons une petite astuce technique. Dans notre exemple, nous examinons le numéro d'ID de l'élève et nous utilisons ce nombre avec la commande FIND. Notre programme stockera le numéro d'ID dans une variable. Les variables utilisées avec la commande FIND peuvent être utilisées comme des macros, et les macros doivent être du type caractère. Le numéro d'ID est de type numérique, aussi il faut donc le convertir en caractère par la fonction STR. Cette fonction prend le nom de la variable et la longueur comme argument, et ramène la même variable comme des caractères. Cela paraît compliqué, mais c'est seulement une instruction de plus dans un programme.

PSEUDO-CODE DE RATACH.CMD

```
Utiliser AMICALE.DBF indexé sur le nom dans la première zone
Utiliser COTIS.DBF indexé sur le numéro ID dans la deuxième zone
Demander la saisie du nom de l'élève pour la recherche
Trouver l'enregistrement de cet élève
Faire ce qui suit :
    Si l'élève n'a pu être trouvé
        Le dire à l'utilisateur
    Si l'élève n'a jamais cotisé
        Le dire à l'utilisateur
    Trouver la première cotisation correspondante
        Présenter toutes les autres cotisations au moyen des numéros ID
        correspondants
```

CODE dBASE II

. MODIFY COMMAND RATTACH

```
* rattach.cmd

* exemple de programme pour rattacher des fichiers sur une rubrique
   commune
SET TALK OFF

* utiliser les deux fichiers
USE AMICALE INDEX NOM
SELECT SECONDARY
USE COTIS INDEX NO:COT
SELECT PRIMARY

* trouver le nom de l'élève désiré
ACCEPT "S'il vous plait, entrez le nom de l'élève à rechercher" TO NOM

* convertir le nom en majuscules et trouver le bon enregistrement
STORE !(NOM) TO NOM
FIND &NOM

* exécutez ce qui suit :
DO CASE

     case # = 0
     * l'élève n'est pas trouvé
     ? NOM, "n'est pas dans le fichier"
     ? " Appuyez sur Return pour continuer"
     SET CONSOLE OFF
     WAIT
     SET CONSOLE ON

     CASE TOT:COTIS = 0
     * cet élève n'a jamais cotisé
     ? "Ne perdons pas notre temps !"
```

```
    OTHERWISE
    * convertir le n° d'ID en caractères
    STORE STR(ID:NO,5) TO clé

    * se placer sur le fichier cotisations
    SELECT SECONDARY

    * trouver le premier enregistrement correspondant
    find &clé

    * afficher un en-tête
    ? "Les cotisations pour", nom, "sont :

    * présenter chaque cotisation jusqu'à la fin du fichier
    * ou bien jusqu'à la rencontre d'un nouveau numéro ID
    DO WHILE .NOT. EOF .AND. STR(ID:NO,5) = clé
        * montrer un enregistrement de cotisation
        ? date, montant, comment
        * aller à l'enregistrement suivant
        SKIP

    * reboucler
    ENDDO

ENDCASE

* suppression des variables locales
RELEASE  clé, nom

* revenir sur la première zone mémoire
SELECT PRIMARY
```

```
. DO LINK

SVP, tapez le nom de l'élève pour la recherche : Loris

Les cotisations de LORIS sont :

2/3/86          30.00
3/4/86          70.00
```

```
. DO LINK

SVP, tapez le nom de l'élève pour la recherche : Dubois

Ne perdons pas notre temps !

. DO LINK

SVP, tapez le nom de l'élève pour la recherche : Schuss

SCHUSS n'est pas dans le fichier
Appuyez sur Return pour continuer
```

EXTENSIONS POUR LE SYSTEME DE GESTION DES COTISATIONS

Si les programmes sont rédigés de façon modulaire, comme les nôtres, le rattachement de fonctions supplémentaires n'est pas dramatique. Avec les techniques acquises grâce à cet ouvrage, vous devez maintenant être capable d'écrire un jeu de programmes pour effectuer les fonctions suivantes :

Ajout d'une cotisation (CAJOUT)

Modification d'une cotisation (CMODIF)

Suppression d'une cotisation (CSUPR)

Impression d'un listing de toutes les cotisations de tous les membres de l'Amicale (CLISTING)

Présentation de toutes les cotisations d'un élève donné (CMONTRE)

Retour au Menu principal

Deuxième partie

GUIDE D'UTILISATION

TABLE DES MATIERES

GUIDE D'UTILISATION

INTRODUCTION

CHAPITRE I

CHAPITRE II

CHAPITRE III

CHAPITRE IV

CHAPITRE V

CHAPITRE VI

INTRODUCTION

dBASE II est un logiciel de gestion de bases de données relationnelles de dimensions moyennes utilisant des commandes simples qui vous permettent de :

- Créer une base de données.
- Editer une base de données, la mettre à jour et lui ajouter des enregistrements avec un minimum de manipulation.
- Gagner énormément sur le temps de programmation du fait de l'indépendance des programmes et des données (ces dernières sont définies indépendamment des programmes qui les utilisent).
- Générer des états en sortie d'une ou plusieurs bases de données, faire automatiquement des multiplications, divisions, sous-totaux et totaux.
- Utiliser les possibilités des écrans en mode page pour la saisie formatée des bordereaux d'entrée des données.

dBASE II est un système performant; nous vous conseillons de prendre la peine de lire complètement le manuel avant de l'utiliser.

LES CONVENTIONS UTILISEES DANS CE MANUEL

Mots en minuscules	Les mots que l'utilisateur doit taper
Mots en majuscules	Les commandes et messages dBASE II
^...^	Indique les différentes options de dBASE II
[...]	Indique des commandes optionnelles
<...>	Partie obligatoire d'une commande de dBase II
<**enter**>	Représente la touche clavier ENTER ou RETURN

LA CONFIGURATION MINIMUM NECESSAIRE

dBASE II nécessite pour fonctionner les configurations suivantes :

- un micro-ordinateur de type Amstrad CPC 6128 ou PCW 8256.
- une ou plusieurs unités disques floppy.
- une disquette système CPM Plus.
- une imprimante (facultative).

LES CARACTERISTIQUES DE dBASE II

dBASE II permet :

65.535	enregistrements de
1000	caractères au maximum
32	rubriques ou champs
254	caractères par rubrique
1.8×10^{63}	est la plus grande valeur numérique possible
1×10^{-63}	est la plus petite valeur numérique possible
10	chiffres par valeur numérique
254	caractères par chaîne de caractères
254	caractères par ligne de commande
254	caractères pour les titres des rapports
100	caractères pour les clés index
5	accumulateurs pour la totalisation

SAUVEGARDE DE DISQUETTE

AVANT D'UTILISER dBASE II, FAITES UNE COPIE DE LA DISQUETTE D'ORIGINE FOURNIE, GARDEZ L'ORIGINAL A PART, UTILISEZ LA COPIE.

- Si vous ne disposez que d'un seul lecteur disquettes, procédez comme suit :

— Mettre votre disquette système CP/M Plus dans votre lecteur et frappez :

```
A>DISCKIT3 (CPC 6128)
```

ou

```
A>DISCKIT (PCW 8256)
```

— Ensuite, vous n'aurez qu'à suivre les indications qui s'inscriront alors à l'écran.

- Si vous êtes l'heureux possesseur de deux unités de disques, la procédure est la suivante :

— Mettre dans l'unité de disque A une disquette formatée et dans l'unité B la disquette originale dBASE II et frapper :

```
A>PIP A:=B:*.*
```

Nous vous conseillons en outre de prendre la précaution de sauvegarder périodiquement vos données en recopiant la disquette de travail sur une disquette vierge.

COMPLEMENT AU MANUEL dBASE II AMSTRAD (page 163)

COMMENT REALISER LA PREMIERE COPIE D'EXPLOITATION DE LA DISQUETTE SOURCE dBASE II A PARTIR DU PCW 8256 ?

Les opérations à réaliser dans l'ordre sont les suivantes :

1. Formater une disquette

2. Si le fichier PROFILE.FRA de la disquette Système CPM PLUS n'a pas été renommé, faites-le en tapant :

A>REN PROFILE.SUB=PROFILE.FRA <RETURN>

3. Éteindre et rallumer le PCW 8256 ou alors rebootez-le. Le système se chargera et exécutera automatiquement les instructions et les différentes commandes contenues dans le fichier PROFILE.SUB. Le résultat sera la création d'une unité de disque virtuelle de 112 Ko, puis un transfert automatique d'un certain nombre d'utilitaires système tel que PIP.COM. A la fin de cette opération, le prompt de disponibilité s'affichera alors à l'écran :

A>

4. Taper :

A>M: <RETURN>

sélectionne le drive M par défaut et l'écran affiche alors :

M>

5. A l'aide de l'utilitaire ERA(SE), supprimer tous les fichiers et utilitaires présents sur l'unité M, à l'exception de PIP.COM

6. Introduire la disquette source dBASE II dans le drive A et commencer le transfert des différents fichiers en tapant :

M>PIP M:=A:DBASE.COM <RETURN>

7. Répéter cette opération pour les autres fichiers. Toutefois avant d'opérer le transfert du fichier DBASEMSG.TXT sur le drive M, remplacer la disquette source dBASE II par la disquette formatée et opérer vers A tout le contenu du drive (à l'exception de PIP.COM) M en tapant :

M>PIP A:=DBASE.COM <RETURN>

M>ERA M:DBASE.COM <RETURN>

Répéter cette double opération avec les autres fichiers du drive M.

8. Reprendre la procédure à partir du point 6 jusqu'au total transfert sur votre disquette de travail la totalité de la disquette source dBASE II.

9. Pour tout autre copie de travail du produit dBASE II, utiliser la copie ci-dessus réalisée (appelée aussi Master de copie) avec l'utilitaire système DISCKIT

A> DISCKIT <RETURN>

Attention, toute copie de la disquette source dBASE II qui a été réalisée, doit être exclusivement réservée à votre usage privé sous peine de poursuite judiciaire.

REMARQUE

Il est tout à fait possible de simplifier cette procédure en créant directement votre fichier d'auto-exécution PROFILE.SUB qui vous génèrera directement un disque virtuel M vide et procèdera ensuite au transfert de PIP.COM dans ce dernier. Dans ce cas, commencer la procédure de transfert à partir du point 6.

PRÉLIMINAIRES AVANT DE DÉMARRER dBASE II

Pendant que votre disquette système CP/M Plus se trouve dans l'unité A, frappez les instructions :

```
A>SETKEYS KEYS.WP
```

Cela aura pour effet l'initialisation des codes de claviers compatibles dBASE II. Pour des informations complémentaires sur cette ligne de commande, veuillez vous reporter à votre manuel d'utilisation.

Notes spécifiques au PCW 8256

1) Il existe une possibilité de créer un disque virtuel avec le PCW 8256. Pour sa création, reportez-vous à votre manuel fourni avec votre AMSTRAD ou consultez le fichier PROFILE.FRA. se trouvant sur votre disquette-système CP/M Plus, en frappant :

```
A>TYPE PROFILE.FRA
```

La première ligne de ce fichier vous montre la définition de la Ram Disk M.

Après la création de votre disque virtuel, à l'aide de PIP copiez dans l'unité ainsi définie, les fichiers programmes dBASE. Pour ce faire, procédez comme suit :

— Mettez votre disquette dBASE dans l'unité A et frappez :

```
A>PIP M:=A:DBASE
A>PIP M:=A:DBASEOVR.COM
```

— Retirez votre disquette dBASE II de l'unité A et remplacez-la par une disquette contenant vos fichiers de données ou vos fichiers programmes.

— Puis frappez dans l'ordre :

```
A>M:<RETURN>
M>DBASE<RETURN>
. Set default to A<Return>
```

Remarque : Vous pourrez procéder de manière identique pour l'utilitaire ZIP.COM et ses deux fichiers associés : DGEM.OVL et ZSCRN.OVL.

2) Sur le PCW 8256, la touche CTRL est remplacée par la touche ALT.

CHAPITRE I

INTRODUCTION AUX BASES DE DONNEES

COMMENT DEMARRER dBASE II ?

Pour démarrer une session dBASE II, frappez dBASE et pressez la touche <ENTER>.

Le programme demande la date du jour que l'utilisateur fournira et qui sera utilisée comme date de création ou de mise à jour des fichiers dBASE. Pour ignorer la question, pressez de nouveau la touche <ENTER>.

Lorsque dBASE II est prêt à recevoir une commande de l'utilisateur, un point (.) apparaît au début de la ligne.

Pour vous montrer les possibilités de dBASE, nous vous proposons de créer une base de données; cela ne prendra que quelques minutes.

COMMENT CREER UNE BASE DE DONNEES

Nous allons créer un fichier de noms pour une application d'envoi de courrier. Chaque enregistrement comprendra les rubriques suivantes :

- un nom de 20 caractères,
- une adresse de 25 caractères,
- une ville de 25 caractères,
- un département de 2 caractères,
- et un code postal de 5 caractères.

Frappez **CREATE.**

dBASE demande le nom du fichier en posant la question suivante :

```
*** DONNEZ LE NOM DU FICHIER :
```

Entrez le nom du fichier suivant les spécifications standard CP/M : un nom commencant par une lettre et d'une longueur de 8 caractères, ne contenant ni deux-points ni espaces. Puisque nous créons un fichier de noms, appelons-le **NOMS**.

Lorsque vous frappez la touche <RETOUR>, dBase II crée un fichier appelé **NOMS.DBF**. La partie du nom figurant après le point est une extension générée par dBASE II : c'est l'abréviation de **database file** (Chapitre V, Types de fichiers).

Dans le traitement des bases de données, chaque rubrique que vous voulez entrer dans un ensemble simple est appelée *champ* et l'ensemble est appelé un *enregistrement* (Section V, les bases de données).

Le fichier **NOMS** comportera des enregistrements de cinq rubriques. dBase doit connaître le nom de chaque rubrique, le type de données qu'elle contiendra, sa dimensiom et le nombre de décimales si la donnée est numérique.

```
.create
DONNEZ LE NOM DU FICHIER :
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP   NOM  TYPE DIM DECIMALE(S)
001
```

Les noms des rubriques peuvent avoir jusqu'à 10 caractères et peuvent être entrés en majuscules et/ou minuscules. Le nom doit commencer par une lettre et ne doit pas contenir d'espaces, mais peut, par contre, avoir des « : » insérés. Limitez les abréviations, l'ordinateur comprendra ce que vous dites, mais l'utilisateur peut-être pas !

Le type de données est spécifié par une simple lettre :

C pour caractères.
N pour valeurs Numériques.
L Logiques.

Dans notre exemple, toutes les rubriques contiendront des caractères. La dimension maximale d'un champ est de 254 caractères. Si le champ est numérique et les places des décimales sont spécifiées, le point séparant la partie décimale compte pour un caractère.

Ayant défini les rubriques, le type de données qu'elles contiendront, ainsi que leur longueur, entrons donc maintenant les informations. Quand vous aurez terminé, l'écran apparaîtra sous la forme suivante :

```
.create
DONNEZ LE NOM DU FICHIER : NOMS
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP  NOM  TYPE DIM  DECIMALE(S)
001    nom,c,20<ENTER>
002    adresse,c,25<ENTER>
003    ville,c,25<ENTER>
004    department,c,2<ENTER>
005    cod postal,c,5<ENTER>
DEFINITION INCORRECTE DU NOM DU CHAMP
005    cod:postal,c,5<ENTER>
006    <ENTER>
```

Remarquez qu'au niveau de la définition de la rubrique n° 5, nous avons inséré un blanc entre Cod et postal; l'erreur nous a été signalée par dBASE II nous donnant ainsi la possibilité de la corriger par COD :POSTAL.

Notons aussi que le type de données pour le code postal a été défini comme « caractère », bien que ce soit des nombres.

La raison de cette définition est qu'une commande dBase telle que **TOTAL** peut totaliser tous les champs numériques dans un enregistrement (sans que vous les listiez tous spécifiquement).

Définissant le champ de code postal en numérique serait simplement une perte de temps, car comme nous pouvons utiliser les opérateurs relationnels « SUPERIEUR A », « INFERIEUR A », « EGAL A OU PAS EGAL A » sur les chaînes de caractères, cela ne nous gênera pas pour effectuer des tris sur le code postal (tris que nous aimerons peut-être faire plus tard).

La touche (ENTER) nous permet de terminer la phase de définition des rubriques. dBASE II sauvegarde alors la structure dans le fichier **NOMS.DBF**, et demande si l' on veut commencer à introduire immédiatement des informations.

Répondez par **Y** si oui.

COMMENT SAISIR LES INFORMATIONS

Si votre écran ne permet pas le positionnement du curseur en (x.y).
Le numéro de l'enregistrement est affiché à l'écran, suivi de la rubrique à saisir; la saisie se fait ligne par ligne, en mode continu avec défilement des lignes. Le nom de chaque rubrique s'affiche sur l'écran, la longueur de chaque rubrique est délimitée par « : », le curseur se positionnant à l'endroit où vous devez commencer la saisie.

Quand vous remplissez le champ ou frappez <enter>, le champ suivant apparaît. Lorsque le dernier champ de l'enregistrement a été saisi (ou ignoré), un nouvel enregistrement commence.

Pour arrêter la saisie, frappez <enter> au niveau de la première rubrique d'un nouvel enregistrement.

Si votre écran dispose de la fonction de positionnement du curseur en (x.y).
L'écran est automatiquement effacé, puis le numéro de l'enregistrement et les différentes rubriques sont affichés simultanément en haut à gauche de l'écran avec le curseur positionné au début du premier caractère de la première rubrique.

(Si vous choisissez l'un des terminaux standard figurant sur la liste d'installation, les champs peuvent être en vidéo inversée ou en demi-intensité. Si vous désirez le changer plus tard, vous pouvez le mettre hors service en utilisant l'option « Y — CHANGE/MODIFY » dans la procédure d'installation).

```
RECORD 00001
NOM         :                    :
ADRESSE     :                         :
VILLE       :                         :
DEPARTMENT  :   :
COD:POSTAL  :       :
```

NOTE Si votre écran ne ressemble pas à l 'exemple précédent, vous avez un problème avec INSTALL. **Recommencez l'installation**.

Les rubriques sont délimitées par « : ». Quand un champ a été rempli ou si vous frappez <enter>, le curseur descend au champ suivant. Le curseur peut revenir en arrière, au champ précédent, par la pression simultanée de la touche Control et de la lettre « E » : « Control-E », en abrégé « ctl-E ».

Lorsque vous avez terminé avec le dernier champ, dBase II vous présente un nouveau cadre d'enregistrement vide.

Entrez les noms et adresses suivants. Nous allons les utiliser bientôt pour vous montrer la qualité des performances de dBase II.

```
BIDONOT,JULES       45 rue des Chaines,Nantes,44          44040
BLUM,GUY             5 rue de la Poste,Louviers,27        27421
ELECTRON,PAUL        2 rue des Pièces,Gallion,27          27940
FOUQUET,SEBASTIEN   68 rue de la Trésorie,Chartres,28     28000
LAMOTTE,THEOPHILE    4 rue de la Laiterie,Chambéry,73     73000
LEROY,JACQUES       20 rue des pistes,Chambéry,73         73000
MARTIN,DIEGO       151 rue des Peupliers,Roubaix,59       59057
POUPON,PIERRE       12 rue Beaudelaire,Angers,49          49044
```

Si vous faites des erreurs, il n'est pas possible de les corriger en revenant en arrière et d'écrire par dessus, lisez les deux pages suivantes sur EDIT *avant* de vous déplacer sur le prochain enregistrement.
Si accidentellement vous revenez au (.) de dBase, tapez :

```
.USE NOMS
.APPEND
```

et continuez votre saisie (ceci vous sera expliqué plus tard dans ce manuel).

Pour arrêter la saisie, après avoir entré le dernier code postal et tandis que vous êtes sur le premier caractère du premier champ de l'enregistrement suivant, pressez la touche <enter>. Si vous y aviez mis des informations ou y aviez déplacé le curseur, pressez simultanément la touche Control et la lettre « Q » : (**control-Q**).

dBase II quitte le mode d'entrée des données et présente son point (.) pour vous montrer qu'il attend vos commandes.

Si vous voulez vous arrêter maintenant, tapez simplement QUIT.

QUIT doit être tapé *toutes les fois* que vous avez terminé avec dBase II. Cette action vous permettra de fermer automatiquement et correctement vos fichiers; sinon, vous pourrez détruire vos bases de données.

COMMENT MODIFIER UNE BASE DE DONNEES AVEC « EDIT » et « BROWSE »

Si vous avez saisi des données erronées, vous pouvez les corriger rapidement et facilement en mode « EDITION PLEIN ECRAN ». Tapez :

```
.USE NOMS
.EDIT <numéro>
```

où « numéro » est le numéro d'un des enregistrements dans la base de données.

dBASE II affiche l'enregistrement entier et vous pouvez utiliser les commandes EDITION PLEIN ECRAN pour modifier tout ou partie des informations de votre enregistrement.
Pour obtenir l'enregistrement suivant, utilisez **CTRL-C**.
Pour revenir à l'enregistrement précédent, utilisez **CTRL-R**.

Essayons de taper EDIT 3.

```
RECORD 00003          DELETED
NOM        :ELECTRON PAUL    :
ADRESSE    :2 rue des Pièces      :
VILLE      :Gallion               :
DEPARTMENT:27:
COD:POSTAL:27940:
```

Si vous marquez un enregistrement pour l'effacer par **CTRL-U**, « DELETED » apparaît en haut de l'écran. En pressant CTRL-U à nouveau le mot « DELETED » disparaîtra et l'enregistrement vous sera restitué.

Si vous tapez **LIST** ou **DISPLAY**, vous verrez dans votre base de données un « * » précédant tous vos enregistrements à effacer.

```
.BROWSE FIELDS <liste de champs>
```

Cette commande est l'une des plus performantes de dBASE II pour la consultation et la modification des données.

BROWSE vous permet d'afficher simultanément tout ou partie du contenu de 19 enregistrements. Une fenêtre mobile de 30 colonnes vous permet de visualiser le contenu des différents champs de chaque enregistrement. **CTRL-B** déplace la fenêtre d'un champ vers la droite, **CTRL-Z** la déplace vers la gauche.

Les champs de plus de 80 caractères seront affichés sur 2 lignes ou plus, de façon à présenter l'ensemble de leur contenu.

Si vous précisez une liste de champs en séparant chaque nom de champ par une virgule, seul le contenu de ces champs sera affiché et ceci dans l'ordre de la liste.

Pour abandonner l'EDITION PLEIN ECRAN, faites **CTRL-Q**. La dernière modification ne sera pas prise en compte.

Pour sortir normalement et sauvegarder les corrections qui ont été faites, utilisez **CTRL-W**.

TOUCHES DE FONCTION

Cas général

CTRL X ou CTRL F	Déplace le curseur au champ suivant.
CTRL E ou CTRL A	Remet le curseur sur le champ précédent.
CTRL D	Déplace le curseur d'un caractère vers la droite.
CTRL S	Déplace le curseur d'un caractère vers la gauche.
CTRL V	Active/désactive l'insertion de caractères.
CTRL G	Efface le caractère sous le curseur.
DEL	Efface le caractère se trouvant à gauche du curseur.
CTRL P	Active/désactive l'imprimante.
CTRL Q	Revient en mode commande sous dBase sans modifier l'enregistrement, même si vous êtes en mode MODIFY.

Mode Modification (MODIFY)

CTRL T	Efface le champ où se trouve le curseur et remonte les champs suivants.
CTRL Y	Remet le champ en cours à blanc, mais laisse tous les champs tels qu'ils étaient.
CTRL N	Déplace le champ d'une position vers le bas afin de permettre l'insertion d'un nouveau champ à la position du curseur.

Mode Ajout (APPEND)

CTRL R	Ecrit l'enregistrement (sur disque) et passe à l'enregistrement suivant.
ENTER ou RETURN	Désactive le mode en cours (si le curseur se trouve sur le premier caractère de la première rubrique).
CTRL Q	N'écrit pas l'enregistrement sur disque et termine le mode en cours.

Mode édition (EDIT et BROWSE) (à ne pas utiliser avec APPEND)

CTRL C	Ecrit l'enregistrement sur disque et passe à l'enregistrement suivant.
CTRL R	Ecrit l'enregistrement sur disque et revient à l'enregistrement précédent.
CTRL U	Active/désactive l'effacement de l'enregistrement.

Mode édition (BROWSE)

CTRL B	Déplace la fenêtre d'un champ vers la droite.
CTRL Z	Déplace la fenêtre d'un champ vers la gauche.

INTRODUCTION AUX COMMANDES dBASE II

Les commandes dBASE II sont généralement des verbes. Vous les entrez quand le (.) de dBase apparaît.
Pour signifier à dBase II le nom du fichier sur lequel vous voulez travailler, tapez **USE <nom du fichier>**.
Pour visualiser l'enregistrement en cours, tapez **DISPLAY**.
Pour visualiser l'ensemble des informations du fichier, tapez **LIST**. (Pour arrêter ou commencer le défilement des informations, tapez **CTRL-S**.

Toutes les commandes dBASE peuvent être abrégées aux quatre premières lettres; mais si vous désirez plus de lettres, ces dernières doivent être correctes (**DISPLAY, DISP, et DISPLA** sont des commandes valides; **DISPRAY** ne l'est pas).

Si l'utilisateur a choisi la correction des commandes erronées en mode conversationnel (voir Installation de dBASE II), celles-ci sont alors vérifiées et les erreurs seront signalées à l'opérateur qui en fera la correction (sans être tenu de refrapper entièrement la ligne).

Supposons que nous avons frappé :

```
EDUT 3
```

```
EDUT 3
*** CETTE COMMANDE NE PEUT ETRE UTILISEE ***
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N)?Y
REMPLACER : U
PAR       : I
EDIT 3
*** DESIREZ-VOUS D'AUTRES CORRECTIONS (Y/N)?N
```

dBase II répète une commande qui lui est inconnue. Si vous décidez de la changer, vous n'avez pas à retaper la commande entière.
A la question REMPLACER : tapez essentiellement la partie erronée pour éviter toute ambiguïté, puis pressez la touche <ENTER>.
A la question PAR : tapez en remplacement l'information que vous voulez émettre. Dans cet exemple, nous avons simplement changé une lettre, mais vous trouverez ceci utile lorsque vous aurez à corriger des erreurs dans des longues lignes de commandes.

Astuce : La commande ERASE vous permet d'effacer complètement l'écran, de positionner le curseur au début de l'écran, en haut à gauche, afin de pouvoir travailler avec un écran net.

UTILISATION DES EXPRESSIONS ET DES OPERATEURS RELATIONNELS (LIST)

Une des plus puissantes caractéristiques de dBase II est la possibilité d'étendre les commandes sur mesure.

Vous pouvez ajouter des « phrases » et expressions à la plupart des commandes et définir plus tard ce que feront ces commandes. Ces dernières peuvent être saisies en majuscules et en minuscules; les lignes de commande peuvent avoir une longueur maximale de 254 caractères. Si la ligne est trop longue par rapport à la largeur de visualisation de l'écran, tapez un « ; » comme *dernier* caractère de la ligne (*pas d'espaces* après le « ; ». dBase II prendra la ligne suivante comme partie de la commande.

Puisque dBase II est une base de données relationnelle vous trouverez utiles les *opérateurs relationnels* suivants :

<	Inférieur à
>	Supérieur à
=	Egal à
< =	Inférieur ou égal à
> =	Supérieur ou égal à
< >	Différent de

Ils génèrent comme résultat une valeur logique (Vrai ou Faux). Si l'expression est Vraie, la commande sera exécutée; dans le cas contraire, la commande ne s'exécutera pas.

Nous avons vu plus haut que la commande LIST fera défiler tous les enregistrements d'un fichier (pour arrêter ou commencer le défilement, tapez **CTRL-S**). La commande complète se présente comme suit :

```
LIST [OFF] [FOR <expression>]
```

L'option OFF affiche l'enregistrement sans son numéro d'enregistrement.

L'option FOR permet de lister les enregistrements qui répondent à un critère de sélection défini dans < expression > . Tapez les lignes suivantes, en utilisant les apostrophes avant et après les informations données (nous aurons plus de détails sur les types de données au Chapitre II).

Exemple :

```
.USE Noms
.LIST
.LIST OFF
.LIST FOR Cod:Postal = '4'
.LIST OFF FOR Cod:Postal < '5'
.LIST FOR Nom='LAMOTTE'
```

Remarque : La sélection peut être réalisée sur une partie du contenu de la rubrique ; il n'est pas nécessaire de donner le nom complet de Mr LAMOTTE; toutefois s'il existe dans le fichier plusieurs LAMOTTE, nous pouvons spécifier le nom complet.

```
.Use noms
.list
00001  BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002  BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003  ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004  FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000
00005  LAMOTTE,THEOPHILE    4 rue de la Laiterie   Chambéry    7373000
00006  LEROY,JACQUES       20 rue des pistes       Chambéry    7373000
00007  MARTIN,DIEGO       151 rue des Peupliers    Roubaix     5959057
00008  POUPON,PIERRE       12 rue Beaudelaire      Angers      4949044

.list off
BIDONOT,JULES       45 rue des Chaines          Nantes      4444040
BLUM,GUY             5 rue de la Poste          Louviers    2727421
ELECTRON,PAUL        2 rue des Pièces           Gallion     2727940
FOUQUET,SEBASTIEN   68 rue de la Trésorie       Chartres    2828000
LAMOTTE,THEOPHILE    4 rue de la Laiterie       Chambéry    7373000
LEROY,JACQUES       20 rue des pistes           Chambéry    7373000
MARTIN,DIEGO       151 rue des Peupliers        Roubaix     5959057
POUPON,PIERRE       12 rue Beaudelaire          Angers      4949044

.list for cod:postal='2'
00002  BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003  ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004  FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000

.list for cod:postal<'7'
00001  BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002  BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003  ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004  FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000
00007  MARTIN,DIEGO       151 rue des Peupliers    Roubaix     5959057
00008  POUPON,PIERRE       12 rue Beaudelaire      Angers      4949044
```

La commande LIST permet d'afficher la structure d'une base nous précisant le nom, le type et la longueur d'une rubrique ainsi que la date de la dernière mise à jour de la base et le nombre d'enregistrements. La forme de la commande est :

LIST STRUCTURE

La commande LIST permet également de vérifier les noms des fichiers (.DBF) contenus dans une disquette . Voici la forme de la commande :
(Ne pas utiliser le « : » de CP/M).

LIST FILES ON <unité de disque>

```
.use noms
.list structure
STRUCTURE DU FICHIER   :NOMS.DBF
NOMBRE D'ENREGISTREMENTS : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP     NOM    TYP  DIM  DECIMALE(S)
001       nom,c,20
002       adresse,c,25
003       ville,c,25
004       department,c,2
005       cod:postal,c,5
**TOTAL**          00077

.list files
DATABASE            FILES     #RCDS     LAST UPDATE
NOMS                DBF       00008     00/00/00
FICCLE              DBF       00211     00/00/00
CHEQUES             DBF       00783     00/00/00
DEPENSE             DBF       00000     00/00/00
COMMANDE            DBF       00000     00/00/00
```

AFFICHAGE DE LA BASE AVEC LA COMMANDE DISPLAY

La commande DISPLAY est équivalente à LIST. Elle peut être utilisée de la façon suivante :

```
        [All]
DISPLAY [record n] [OFF] [FOR <expression>]
        [Next n]
```

Les options [All], [record] ou [Next n] permettent de travailler sur toute la base (All), sur l'enregistrement n de la base (record n) ou sur les n enregistrements suivant celui en cours (Next n).

```
DISPLAY All
DISPLAY record 3
DISPLAY Next 4

.display all
00001   BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002   BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003   ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004   FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000
00005   LAMOTTE,THEOPHILE    4 rue de la Laiterie   Chambéry    7373000
00006   LEROY,JACQUES       20 rue des pistes       Chambéry    7373000
00007   MARTIN,DIEGO       151 rue des Peupliers    Roubaix     5959057
00008   POUPON,PIERRE       12 rue Beaudelaire      Angers      4949044

.display record 3
00003   ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940

.display next 4
00001   BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002   BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003   ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004   FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000
```

Avec **LIST** l'option FOR permet de sélectionner, dans les enregistrements manipulés, ceux qui répondent à un critère bien défini par <expression>.
Les deux commandes suivantes sont équivalentes :

```
DISPLAY STRUCTURE  =  LIST STRUCTURE
DISPLAY FILES      =  LIST FILES
```

LIST et DISPLAY nous montrent les caractéristiques des fichiers sur une unité de disque en utilisant le CP/M « nom générique ». **DISPLAY FILES LIKE*.COM ON B** vous visualisera tous les fichiers .COM sur l'unité de disque B. Si vous n'êtes pas sûr, prenez votre manuel CP/M et utilisez la forme suivante :

DISPLAY FILES LIKE <Nom générique>

COMMENT SE POSITIONNER DANS UNE BASE (GO ou GOTO et SKIP)

Lorsque vous aurez constitué votre base de données sous dBASE II, vous pouvez vous déplacer rapidement et facilement d'un enregistrement à un autre.

```
USE Noms
GO TOP
DISPLAY
GO BOTTOM
DISPLAY
GOTO 5
DISPLAY
8
DISPLAY

.Use Noms
.go top
.display
00001   BIDONOT,JULES       45 rue des Chaines      Nantes      4444040

.go bottom
.display
00008   POUPON,PIERRE       12 rue Beaudelaire      Angers      4949044

.goto 5
.display
00005   LAMOTTE,THEOPHILE    4 rue de la Laiterie   Chambéry    7373000

.8
.display
00008   POUPON,PIERRE       12 rue Beaudelaire      Angers      4949044
```

La commande GO TOP ou GOTO TOP nous positionne sur le premier enregistrement du fichier.

La commande GO BOTTOM ou GOTO BOTTOM nous positionne sur le dernier enregistrement.

La commande GO 5, GOTO 5 ou 5 nous positionne sur l'enregistrement 5 de la base de données.

La commande SKIP ± nous permet de passer n enregistrements en avant (par rapport à l'enregistrement en cours) ou en arrière (-n)

```
DISPLAY
SKIP -3
DISPLAY
SKIP
DISPLAY
```

```
.display
00008  POUPON,PIERRE      12 rue Beaudelaire     Angers      4949044

.skip-3
enregistrement :00005

.display
00005  LAMOTTE,THEOPHILE   4 rue de la Laiterie  Chambéry    7373000

.skip
enregistrement :00006

.display
00006  LEROY,JACQUES      20 rue des pistes      Chambéry    7373000
```

LA COMMANDE ?

La commande ? vous permet d'utiliser le mode calcul de dBase II. Tapez simplement le signe ? suivi d'un espace et du nombre ou des fonctions mathématiques que vous voulez calculer et dBase vous répondra à la ligne suivante. Si vous tapez ? ? la réponse figurera sur la même ligne.

Tapez par exemple :

```
? 73/3.0000
? 73.00/3
? 73/3
```

```
.? 73/3.0000
24.3333
.? 73.00/3
24.33
.? 73/3
24
```

La commande ? affiche le résultat du calcul avec un nombre de décimales égal au plus grand nombre de décimales utilisées dans la commande.

La commande ? peut aussi être interprétée comme une question : « qu'est-ce... ? » : les <...> seront remplacés par une expression, une variable, un nom de champ, une

variable mémoire, une fonction de dBase II ou une liste de ces fonctions séparées par des virgules. Exemple :

```
USE Noms
6
? Cod:Postal
? Nom
SKIP
? Nom
GO BOTTOM
? Ville
```

```
.use noms
.6
.? cod:postal
73000
.? nom
LEROY,JACQUES
.? department
   73
.skip
ENREGISTREMENT : 00007
.? nom
MARTIN,DIEGO
.go bottom
.? ville
Angers
```

AJOUTER OU INSERER DES INFORMATIONS DANS UNE BASE (APPEND, INSERT)

L'opérateur peut ajouter des enregistrements à une base de données existante en frappant les commandes :

```
USE noms
APPEND
```

```
.use noms
.append
RECORD # :00009
NOM         :                    :
ADRESSE     :                         :
VILLE       :                         :
DEPARTMENT  :   :
COD:POSTAL  :       :
```

dBASE II affiche un numéro d'enregistrement correspondant au dernier numéro de l'enregistrement du fichier plus 1 ainsi que les différentes rubriques à saisir. Les rubriques seront saisies séquentiellement à partir de la première position du curseur. Si vous remplissez le champ entier avec des données, le curseur se déplace automatiquement au champ suivant. Sinon pressez la touche <ENTER>.

Si aucune donnée n'a été entrée dans le champ, pressez <ENTER> pour passer au champ suivant. Les champs de caractères seront à blanc, les champs numériques seront remplis de zéros. Lorsque vous entrez des données numériques, s'il n'y a pas de décimales, il n'est pas besoin de les taper : dBase II vous mettra automatiquement le point suivi du nombre de zéros nécessaires.

La commande INSERT permet d'insérer un enregistrement dans le fichier, à l'endroit où vous le désirez (exemple : vous voulez les avoir par ordre alphabétique). La commande est de la forme :

INSERT BEFORE BLANK

L'enregistrement créé sera inséré après l'enregistrement en cours. L'option BEFORE permet de l'insérer avant l'enregistrement en cours. Dans les deux cas, dBase II affiche le numéro de l'enregistrement et les différentes rubriques à saisir (voir CREATE ou APPEND). L'option BLANK insère un enregistrement vide.

Utiliser le mot INSERT seul positionne le nouvel enregistrement juste derrière le dernier enregistrement du fichier. En spécifiant BEFORE, le nouvel enregistrement sera mis avant l'enregistrement en cours. Dans tous les cas, la vitesse est la même que pour les commandes APPEND et CREATE. Si BLANK a été spécifié, un enregistrement vide sera inséré.

Ajoutez alphabétiquement les noms suivants dans le fichier <noms.dbf>.

```
LECHAT,CHARLES      5 avenue Lapatte,Montrond,42        42210
PERDREAU,RENE      12 rue de la Chasse,Romantin,41      41200
ROSE,ANATOLE       26 rue du Tonneau,Bordeaux,33        33200
```

```
USE Noms
5
INSERT BEFORE (entrer la première ligne ci-dessus)
APPEND        (entrer les 2 autres lignes)
```

La commande INSERT insère **un** enregistrement, lorsque vous aurez rempli le dernier champ, dBase reviendra au mode commande (.).

Pour sortir du mode APPEND, positionnez le curseur au début d'un champ nouveau puis pressez <enter> ou CTRL-Q.

Dans tous les cas, vous pouvez sortir d'un enregistrement en faisant CTRL-W, ce qui vous permet de sauvegarder tout ce qui a été saisi et de revenir au mode commande.

COMMENT EPURER UNE BASE DE DONNEES (DELETE, RECALL, PACK)

On peut effacer un enregistrement directement en mode EDIT. L'enregistrement est alors **marqué**; il est logiquement effacé, mais physiquement présent.

Pour effacer un enregistrement en cours, tapez DELETE.

Pour effacer plus d'un enregistrement, utilisons la forme DELETE <étendue> (où étendue a la même signification pour les autres commandes de dBase II telles que *ALL, RECORD N ou NEXT N*)

- Voici comment se présente une extension de la commande DELETE.

```
DELETE [etendue]  [FOR <expression>]
```

où expression est une condition ou un ensemble de conditions (Ceci sera développé dans le chapitre II).

```
        [All]
DELETE  [Next n] [FOR <expression>]
        [Record n]
```

La deuxième forme de la commande DELETE est la suivante :

```
DELETE FILE <unité de disque>:Noms.DBF
```

Cette commande détruit le fichier NOMS.DBF.

ATTENTION : **Cette commande est excessivement dangereuse, le fichier est complètement perdu.**

Nous précisons que la commande DELETE efface logiquement les enregistrements et non physiquement (contrairement à la commande DELETE FILE). Au lieu d'effacer les enregistrements, DELETE marque les enregistrements avec un « * ». Vous pouvez visualiser les enregistrements marqués par LIST ou DISPLAY mais dBase les ignorera et ne les comptera pas dans son traitement.

Les enregistrements peuvent être récupérés par la commande RECALL sous la forme suivante :

```
        [All]
RECALL  [Next n] [FOR <expression>]
        [Record n]
```

Toutefois, l'opérateur a la possibilité d'effacer physiquement les enregistrements qui ont été préalablement marqués par DELETE, en exécutant la commande PACK. La Commande PACK efface tous les enregistrements marqués d'un « * » et vous donne le nombre d'enregistrements contenus dans le fichier.

Note : si vous avez utilisé PACK, l'enregistrement est définitivement détruit.
Essayons de taper :

```
USE Noms
LIST
DELETE RECORD 2
DELETE RECORD 4
LIST
RECALL RECORD 4
LIST
PACK
LIST
```

L'exemple ci-dessous nous montre les quelques premiers enregistrements de notre fichier au fur et à mesure que nous exécutons ces commandes.

```
.List
00001   BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002   BLUM,GUY             5 rue de la Poste      Louviers    2727421
00003   ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004   FOUQUET,SEBASTIEN   68 rue de la Trésorie   Chartres    2828000
00005   LAMOTTE,THEOPHILE    4 rue de la Laiterie   Chambéry    7373000
00006   LECHAT,CHARLES       5 avenue Lapatte       Montrond    4242210

.delete record 2
00001 ENREGISTREMENT(S) EFFACE(S)

.delete record 4
00001 ENREGISTREMENT(S) EFFACE(S)

.list
00001   BIDONOT,JULES       45 rue des Chaines      Nantes      4444040
00002 *BLUM,GUY              5 rue de la Poste      Louviers    2727421
00003   ELECTRON,PAUL        2 rue des Pièces       Gallion     2727940
00004 *FOUQUET,SEBASTIEN    68 rue de la Trésorie   Chartres    2828000
00005   LAMOTTE,THEOPHILE    4 rue de la Laiterie   Chambéry    7373000
00006   LECHAT,CHARLES       5 avenue Lapatte       Montrond    4242210
```

```
.Recall record 4
00001 ENREGISTREMENT(S) RESTITUE(S)

.list
00001  BIDONOT,JULES       45 rue des Chaines     Nantes      4444040
00002 *BLUM,GUY             5 rue de la Poste     Louviers    2727421
00003  ELECTRON,PAUL        2 rue des Pièces      Gallion     2727940
00004  FOUQUET,SEBASTIEN   68 rue de la Trésorie  Chartres    2828000
00005  LAMOTTE,THEOPHILE    4 rue de la Laiterie  Chambéry    7373000
00006  LECHAT,CHARLES       5 avenue Lapatte      Montrond    4242210
.pack
COMPACTAGE TERMINE  00005 ENREGISTREMENT(S) TRANSFERE(S)
.list
00001  BIDONOT,JULES       45 rue des Chaines     Nantes      4444040
00002  ELECTRON,PAUL        2 rue des Pièces      Gallion     2727940
00003  FOUQUET,SEBASTIEN   68 rue de la Trésorie  Chartres    2828000
00004  LAMOTTE,THEOPHILE    4 rue de la Laiterie  Chambéry    7373000
00005  LECHAT,CHARLES       5 avenue Lapatte      Montrond    4242210
```

RESUME DU CHAPITRE I

Jusqu'à présent, nous avons vu ce qu'une base de données relationnelle aussi performante que dBase II peut nous apporter.

Vous pouvez créer maintenant une nouvelle base de données et commencer la saisie des informations en très peu de temps.

Les informations sont saisies en utilisant la commande APPEND. Une base de données peut être mise à jour avec la commande EDIT, listée avec les deux commandes DISPLAY et LIST. Les commandes GO, GOTO et SKIP permettent d'accéder directement à un enregistrement donné.

La commande ? permet d'utiliser le micro-ordinateur comme une machine à calculer et d'afficher les informations de la base, rubrique par rubrique.

Si vous voulez changer les informations, les commandes EDIT, DELETE, RECALL, et PACK sont faciles à utiliser.

Ayant présenté les commandes usuelles de dBASE, nous nous proposons d'introduire dans les chapitres suivants des commandes plus évoluées; pour cela, nous vous proposons de créer les bases de données suivantes.

```
.create
DONNEZ LE NOM DU FICHIER : DEPENSE
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP   NOM TYPE DIM DECIMALE(S)
001     D:Cheque,c,7
002     n:cheque,c,5
003     client,c,3
004     num:trav,n,3
005     nom,c,20
006     Descrip,c,20
007     montant,n,9,2
008     d:factu,c,7
009     n:factu,c,7
010     heures,n,6,2
011     num:emp,n,3
012
```

```
.create
DONNEZ LE NOM DU FICHIER : COMMANDE
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP   NOM  TYPE DIM  DECIMALE(S)
001     n:client,c,9
002     Article,c,20
003     quantite,n,4
004     prix,n,7,2
005     montant,n,9,2
006     Ret:comn,l,1
007     dat:comm,c,6
008
```

CHAPITRE II

MANIPULATIONS DE dBASE II

Dans ce chapitre, nous développerons l'utilisation des expressions modifiant les commandes de dBASE II. C'est la partie la plus importante de ce manuel.

Les commandes de dBASE II peuvent s'apprendre assez aisément puisqu'elles ressemblent à l'anglais. Apprendre une autre commande ne fait qu'augmenter votre vocabulaire et votre répertoire d'un autre mot.

Les expressions, combinées avec les commandes, vous donnent tous les éléments dont vous avez besoin pour manipuler vos données et exécuter des tâches spécifiques. Lorsque vous aurez appris comment utiliser les expressions, vous n'aurez plus que deux choses à apprendre sur la programmation pour être capable d'écrire des applications sous forme de fichiers de commandes. (Voir le chapitre III sur la façon de prendre les décisions et comment répéter une séquence de commandes).

COMMENT UTILISER LES EXPRESSIONS POUR LA SELECTION ET LE CONTROLE

Nous vous avons donné dans le chapitre I une courte introduction sur les expressions qui peuvent être utilisées avec les commandes de dBASE II.

Comme vous l'avez vu, il existe une façon puissante d'étendre vos commandes pour manipuler vos données facilement et rapidement. Plusieurs commandes de dBASE II peuvent être modifiées à l'aide de l'expression suivante :

<COMMAND> [FOR <expression>]

Cette extension vous donne une souplesse d'utilisation que vous n'obtiendrez pas avec d'autres systèmes. Des programmeurs expérimentés nous ont affirmé réaliser beaucoup plus rapidement leurs applications avec dBASE II qu'avec BASIC, COBOL, FORTRAN ou PL/1.

Pour pouvoir bénéficier de ces avantages, vous avez besoin de comprendre le fonctionnement des expressions et des opérateurs, de savoir comment combiner la modification des commandes dans un fichier afin d'exécuter répétitivement les mêmes tâches.

Rappel : Nous introduirons de nouvelles commandes, au fur et à mesure, dans le texte. Nous essaierons de vous expliquer à chaque fois un aspect particulier de ces commandes vous permettant ainsi de mieux comprendre leur action sur vos bases de données.

Voir le résumé de la fin de la Partie I et les définitions de la Partie II pour comprendre la fonction d'une commande.

CONSTANTES ET VARIABLES (STORE)

Les expressions de dBASE II sont utilisées pour vous faciliter la sélection et la manipulation des données dans votre base de données (voir ^**DISPLAY**^). Elles peuvent contenir des **constantes** et des **variables**.

Les ***constantes*** sont des données qui ne changent pas, qu'elles apparaissent dans une base de données ou à l'intérieur d'un programme. Elles ont des valeurs *littérales* car elles représentent exactement ce qu'elles sont. Les exemples sont les chiffres tels que 3 et les valeurs logiques comme V et F.

Les *caractères* et *chaînes de caractères* (l'espace ainsi que tous les caractères imprimables) peuvent aussi être des constantes, mais elles doivent être traitées d'une manière légèrement différente.

Une *« chaîne »* est simplement une suite de caractères pouvant contenir des espaces, des chiffres et des symboles. Ceux-ci peuvent être modifiés et traités comme des données. Une *« sous-chaîne »* est une partie de chaîne.

Si un caractère ou une suite de caractères est à traiter comme une *chaîne de caractères*, il faut l'encadrer par des apostrophes, des guillemets, ou des parenthèses, afin que l'ordinateur comprenne qu'il traite des caractères en temps que caractères.

Afin de mieux nous comprendre, appelez dBASE II sur votre ordinateur et USE <Noms>. Frappez :

```
dBASE
USE Noms
? "Nom"
? Nom
```

Pour répondre à la première commande du « ? », l'ordinateur répond NOM parce que c'est la valeur d'une constante. Si vous enlevez les apostrophes, l'ordinateur vérifie si ce mot est une commande. Sinon, il vérifie si c'est le nom d'une variable.

Les ***variables*** sont des données dont la valeur peut changer. Souvent, ce sont les noms des champs d'une base de données dont le contenu peut changer. Dans ce cas, l'ordinateur trouve dans notre fichier que nous avons un champ appelé <Nom>, aussi va-t-il nous donner les données de ce champ. Frappez :

```
SKIP 3
? Nom
        . use noms
        . ? 'Nom'
        Nom
        . ? Nom
        BIDONOT,JULES
        .skip 3
        ENREGISTREMENT : 00004
        . ? Nom
        FOUQUET,SEBASTIEN
```

Maintenant, frappez ^USE^. Puisque nous n'avons pas spécifié de nom de fichier, l'ordinateur ferme simplement tous les fichiers.

Si nous frappons à nouveau ^? **Nom**^, l'ordinateur nous répond que nous avons fait une erreur. Nous avons essayé d'utiliser une variable qui n'existait pas et nous n'utilisons pas de fichier ayant un tel nom de champ.

Les variables peuvent être des variables mémoire ou des noms de champs. dBASE II réserve une partie de la mémoire pour y stocker au maximum 64 variables, chacune ayant une longueur maximum de 254 caractères, mais avec un total maximum de 1 536 caractères pour toutes les variables.

La disponibilité de ces 64 cases vous permet de les remplir temporairement de données lorsque vous travaillez sur un problème.

Les noms de ces variables doivent répondre aux critères de dBASE II (commencer par une lettre, avoir une longueur de 10 caractères maximum, insertion possible des « : », utilisation possible des nombres mais non d'espaces).

Vous pouvez utiliser ces variables mémoire pour y stocker temporairement des données ou pour garder l'entrée des données séparée des champs de variables. Par exemple, lors d'une session, nous pouvons remplir la date dans une case (variable) appelée <Date>. Durant cette session, nous pouvons obtenir cette variable en demandant <Date>. Vous pouvez la placer dans n'importe quel champ de date et dans n'importe quelle base de données sans avoir à la réentrer.

Pour entrer des données (de type caractère, numérique ou logique) dans une variable mémoire, nous pouvons utiliser la commande **^STORE^**. La forme complète est :

STORE <expression> TO <variable mémoire>

Frappez les lignes suivantes :

```
STORE "Comment est-ce allé si loin ?" TO Message
STORE 10 TO Heures
STORE 17.35 TO Taux:Paie
? Taux:Paie*Heures
? Message
             .STORE "Comment est-ce allé si loin?" TO Message
             Comment est-ce allé si loin?
             .STORE 10 TO Heures
              10
             .STORE 17.35 TO Taux:paie
              17.35
             .? Taux:paie*Heures
                         173.50
             .? Message
             Comment est-ce allé si loin ?
```

A noter que nous utilisons les guillemets avant et après les chaînes de caractères (constantes) sur la première ligne, ceci en raison de l'utilisation d'apostrophes à l'intérieur de la chaîne.

Si ceci n'est pas encore clair, faites un essai avec ou sans guillemets, ou apostrophes, pour voir la différence entre les constantes et les variables. Pour commencer, frappez les lignes suivantes :

```
STORE 99 TO Variable
STORE 33 TO Autre
STORE Variable/Autre TO Troisième
STORE '99' TO Constante
? Variable/Autre
? Variable/3
? Constante/3
DISPLAY MEMORY
             . STORE 99 to Variable
               99
             . STORE 33 TO Autre
               33
             . STORE Variable/autre TO Troisième
               3
             . STORE '99' TO Constante
               99
             . ? Variable/Autre
                       3
             . ? Variable/3
                      33
             . ? Constante/3
             *** ERREUR DE SYNTAXE ***
                       ?
             ? CONSTANTE/3
             . DISPLAY MEMORY
             MESSAGE              (C)   Comment est-ce allé si loin ?
             HEURES               (N)    10
             TAUX:PAIE            (N)   17.35
             VARIABLE             (N)    99
             AUTRE                (N)    33
             TROISIEME            (N)    3
             CONSTANTE            (C)    99
             **TOTAL**  07 VARIABLES USED 00054 BITS USED
```

Le fait d'entrer une valeur dans une variable spécifie automatiquement à dBASE II le type des données. Donc, dès maintenant, vous ne pouvez mélanger les types de données (exemple : essayer de diviser une chaîne de caractères avec un nombre).

REGLE : Les chaînes de caractères qui apparaissent dans les expressions doivent être délimitées par des apostrophes ou par des guillemets. Les chaînes de caractères peuvent contenir tous les caractères imprimables y compris des espaces. Si vous voulez utiliser le signe & (« et commercial ») comme caractère, il doit être précédé et suivi d'un espace, car il est également utilisé pour une macro-fonction de dBASE II qui sera décrite plus loin.

La dernière commande présentée dans l'exemple est une autre forme de **^DISPLAY^** que vous trouverez très utile. (Vous pouvez aussi frapper **^LIST MEMORY^**).

Vous pouvez remettre à blanc une variable mémoire en frappant **^RELEASE <nom>^**, ou vous pouvez effacer toutes les variables mémoire en frappant **^RELEASE ALL^**.

Frappez les lignes suivantes (auparavant, faites éventuellement **^ERASE^** pour avoir un écran net) :

```
DISPLAY MEMORY
RELEASE Autre
DISPLAY MEMORY
RELEASE ALL
DISPLAY MEMORY
```

Conseil : Lorsque vous nommez les variables, essayez d'utiliser le nombre de caractères nécessaires à une bonne compréhension.

Conseil : Si vous n'utilisez que neuf caractères pour les champs de noms de vos bases de données, lorsque vous voulez les utiliser comme variables mémoire, vous pouvez les faire précéder d'un « M ».

LES OPERATEURS DE dBASE II

Les **opérateurs** servent aux manipulations que dBASE II exécute sur vos données. Certains de ces opérateurs vous sont familiers; d'autres peuvent vous demander un peu de pratique.

Les **opérateurs arithmétiques** sont les plus faciles. Ils génèrent des résultats arithmétiques.

( ) : les parenthèses sont utilisées pour grouper les termes.
* : utilisé pour la multiplication.
/ : utilisé pour la division.
+ : utilisé pour l'addition.
— : utilisé la soustraction.

Les opérateurs arithmétiques sont appliqués par ordre de priorité. Voici cet ordre : parenthèses, multiplication et division, addition et soustraction. Quand les opérateurs ont une priorité égale, ils sont appliqués de la gauche vers la droite. Voici quelques exemples :

17/33*72 + 8 = 45.09 (division, multiplication, puis addition)
17/(33*72 + 8) = 0,00644 (multiplication, addition, puis division)
17/33*(72 + 8) = 41.21 (division, addition, puis multiplication)

Les **opérateurs relationnels** donnent un résultat logique Booléen, c'est-à-dire de type vrai ou faux. Les différents opérateurs utilisés sont les suivants :

< : inférieur à
> : supérieur à
= : égal à
< > : différent de
< = : inférieur ou égal à
> = : supérieur ou égal à

Frappez les exemples suivants :

```
USE Noms
LIST FOR Cod:Postal <= '28000'
LIST FOR Adresse  > '123'
LIST FOR Nom = 'ROSE'

.LIST FOR Cod:Postal <='28000'
00003  ELECTRON,PAUL       2 rue des Pièces        Gallion    2727940
00004  FOUQUET,SEBASTIEN  68 rue de la Trésorerie  Chartres   2828000

.LIST FOR adresse > '123'
00007  MARTIN,DIEGO      151 rue des peupliers     Roubaix   5959057

.LIST FOR Nom='ROSE'
00010  ROSE,ANATOLE       26 rue du Tonneau        Bordeaux  3333200
```

Les **opérateurs logiques** étendent les possibilités de tri de données et de manipulation des enregistrements des bases de données. Ils génèrent des résultats logiques vrais ou faux. Voici leur liste par ordre de priorité à l'intérieur d'une expression. Exemple (.NOT. est mis avant .AND., etc.) :

( )	: les parenthèses servent pour le groupement des termes.
.NOT.	: est une négation booléenne.
.AND.	: est un et booléen.
.OR.	: est un ou booléen.
$	: est une sous-chaîne d'un opérateur logique (présence d'une chaîne dans une autre chaîne)

Exemple :

```
LIST FOR (job:num=730 .OR. Job:num=731);
          .AND. (Date:note >= '791001' .AND.;
                 Date:note <= '791031')
```

Cet exemple visualisera toutes les factures du mois d'octobre 1979 dont les numéros de travail sont 730 et 731 (notez comment la ligne de commande a été étendue avec des « ; »).

Si vous n'êtes pas habitué aux opérateurs logiques, partez du fait que ces opérateurs vous donneront un résultat qui sera vrai ou faux. Dans notre exemple, dBase II pose les questions suivantes pour chaque enregistrement :

1) Job :num est-il égal à 730 (Vrai ou Faux) ?
2) Job :num est-il égal à 731 (Vrai ou Faux) ?
3) Date :note est-elle supérieure ou égale à '791001' (Vrai ou Faux) ?
4) Date :note est-elle inférieure ou égale à '791031' (Vrai ou Faux) ?

dBASE II exécute alors trois tests logiques (.OR., .AND., .AND.) avant de décider de visualiser ou non l'enregistrement.

Les parenthèses utilisées dans les expressions logiques sont utilisées de la même manière que dans les expressions arithmétiques afin de clarifier les opérations et les relations. Pour le premier .AND., dBASE II sélectionnera uniquement les enregistrements fournissant une valeur « vraie » aux conditions posées dans les deux expressions comprises entre les parenthèses.

Dans la première expression, dBASE II vérifie d'abord le champ <Numéro-de-travail>. Si la valeur de ce champ est 730 ou 731, l'expression est vraie. Si le champ contient

d'autres valeurs, cette expression est fausse et l'enregistrement ne sera pas sélectionné. Si la première expression est vraie, dBase doit encore vérifier le contenu du champ Date :note pour évaluer la seconde expression. Si le contenu de ce champ est compris entre '791001' et '791031' inclus, cette expression est vraie et l'enregistrement sera sélectionné et visualisé, sinon la totalité de l'expression est fausse et dBASE II passera à l'enregistrement suivant pour procéder aux mêmes tests d'évaluation.

Essayons quelques-unes de ces commandes avec <Noms.DBF>. Frappez les lignes suivantes :

```
    USE Noms
    DISPLAY all FOR Cod:Postal > '4' .AND. Cod:Postal < '7'
    DISPLAY all FOR Nom < 'L'
    DISPLAY all FOR Adresse > '45' .AND. Adresse < '151'
    DISPLAY all FOR Adresse > '40' .OR. Adresse < '155'
. USE Noms
. DISPLAY all FOR Cod:Postal>'4' .AND. Cod:Postal<'7'
00007  MARTIN,DIEGO      151 rue des Peupliers     Roubaix    5959057

.DISPLAY all FOR Nom<'L'
00001  BIDONOT,JULES      45 rue des Chaines       Nantes     4444040
00002  ELECTRON,PAUL       2 rue des Pièces        Gallion    2727940
00003  FOUQUET,SEBASTIEN  68 rue de la Trésorerie  Chartres   2828000

. DISPLAY all FOR Adresse >'45' .AND. Adresse <'151'
00004  FOUQUET,SEBASTIEN  68 rue de la Trésorie    Chartres   280000

. DISPLAY all FOR Adresse <'40' .OR. Adresse <'155'
00001  BIDONOT,JULES      45 rue des Chaines       Nantes     4444040
00002  ELECTRON,PAUL       2 rue des Pièces        Gallion    2727940
00003  FOUQUET,SEBASTIEN  68 rue de la Trésorerie  Chartres   2828000
00004  LAMOTTE,THEOPHILE   4 rue de la Laiterie    Chambéry   7373000
00005  LECHAT,CHARLES      5 avenue Lapatte        Montrond   4242210
00006  LEROY,JACQUES      20 rue des Pistes        Chambéry   7373000
00007  MARTIN,DIEGO      151 rue des Peupliers     Roubaix    5959057
00008  POUPON,PIERRE      12 Rue Beaudelaire       Angers     4949044
00009  PERDREAU,RENE      12 rue de la Chasse      Romantin   4141200
00010  ROSE,ANATOLE       26 rue du Tonneau        Bordeaux   3333200
```

Notez ce qui se passe avec la dernière commande : tous les enregistrements ont été visualisés. Si vous n'êtes pas habitué aux opérateurs logiques, n'utilisez pas cette sélection non-sélective.

Le signe $ est une sous-chaîne d'un opérateur logique et est extrêmement utile de par ses capacités de recherche. Il est utilisé sous la forme suivante :

⟨sous-chaîne⟩ $ ⟨chaîne⟩

L'opérateur $ recherche la présence d'une sous-chaîne (à gauche) à l'intérieur d'une chaîne (à droite). Les termes peuvent être aussi bien des chaînes de variables que des chaînes de constantes. Pour en voir le fonctionnement, frappez les lignes suivantes :

```
    USE Noms
    LIST FOR 'EC' $ Nom
    LIST FOR '6' $ Adresse
    LIST FOR '5' $ Department
    ? 'al' $ 'Gallion'
    . GO 2
    . DISPLAY
    ? Department $ "27"
. USE Noms
. LIST FOR 'EC' $ Nom
00002  ELECTRON,PAUL       2 rue des Pièces        Gallion    2727940
00005  LECHAT,CHARLES      5 avenue Lapatte        Montrond   4242210

. LIST FOR '6' $ Adresse
00003  FOUQUET,SEBASTIEN  68 rue de la Trésorerie  Chartres   2828000
00010  ROSE,ANATOLE       26 rue du Tonneau        Bordeaux   3333200

.LIST FOR '5' $ Department
00007  MARTIN,DIEGO      151 rue des Peupliers     Roubaix    5959057

. ? 'al' $ 'Gallion'
.T.

.go 2
. display
00002  ELECTRON,PAUL       2 rue des Pièces        Gallion    2727940

. ? Department '27'
.T.
```

Avec cette fonction, nous aurions pu simplifier la structure de notre fichier de noms. Les départements par exemple auraient pu faire partie de l'adresse. Pour sélectionner un département spécifique, nous aurions pu simplement taper la ligne suivante (où XX est l'abréviation du département désiré) :

```
<COMMAND> FOR 'XX' $ Adresse
```

Les **opérateurs chaîne de caractères** génèrent des résultats de chaîne.

+ = concaténation de chaîne (exacte)
— = concaténation de chaîne (suppression des blancs)

Le mot concaténation signifie qu'une chaîne de caractères est accolée à la fin d'une autre chaîne de caractères. Frappez les lignes suivantes :

```
USE Noms
? Nom + Adresse
? Nom - Adresse
? 'Le nom dans cet enregistrement est ' + Nom;
     - ' et l'adresse est '+ Adresse
```

Les signes + et — relient deux chaînes. Le signe « + » relie les chaînes exactement telles qu'elles sont trouvées. Le signe « — » supprime les blancs inutiles se trouvant à droite d'une chaîne de caractères. Les blancs ne sont pas éliminés mais ne sont pas apparents entre les chaînes qui ont été reliées.

Si vous voulez éliminer les blancs inutiles, vous pouvez utiliser la fonction **TRIM**. Cette fonction est utilisée en frappant ^STORE TRIM (<**variable**>) TO <**variable**>^. Nous aurions pu frapper ^**STORE TRIM(Nom) TO (Nom)**^ pour éliminer tous les blancs suivant les caractères du nom. Pour éliminer tous les blancs dans notre exemple, nous aurions pu frapper ^**STORE TRIM(Nom — Adresse) TO Exemple**^.

Maintenant que vous connaissez l'utilisation des expressions et des opérateurs de dBASE II, nous allons continuer avec d'autres commandes dBASE II. Nous vous donnerons quelques exemples d'utilisation d'expressions et opérateurs puisque nous travaillons maintenant à développer des fichiers de commandes.

COMMENT CHANGER LA STRUCTURE D'UNE BASE VIDE (commande MODIFY)

ATTENTION : La commande MODIFY détruira votre base de données. Suivez, s'il vous plaît, très attentivement les instructions.

Lorsque votre base de données est vide, la commande **MODIFY** est le moyen le plus rapide et le plus facile pour ajouter, effacer, renommer, changer la dimension ou changer carrément la structure de votre base de données. *Cette commande détruira toutes les informations dans votre base de données, aussi ne l'utilisez pas si vous y avez déjà entré des données. Nous vous montrerons plus tard un moyen de modification plus sûr.*

<Depense.DBF> est un fichier encore vide. Aussi, allons-nous travailler avec ce fichier. Frappez les lignes suivantes :

```
USE Depense
LIST STRUCTURE
MODIFY STRUCTURE
^y^              (en réponse à la question)
   . use Depense
   . list structure
   STRUCTURE DU FICHIER : DEPENSE.DBF
   NOMBRE D'ENREGISTREMENT : 00000
   DATE DE LA DERNIERE MISE A JOUR : 00/00/00
   BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION

   CHAMP     NOM          TYPE    DIM       DECIMALE(S)
   001       CLIENT       C       004
   002       JOB:NUM      C       003
   003       DATE:NOTE    C       006
   004       FOURNISR     C       028
   005       DESCRIP      C       010
   006       HEURES       N       006        002
   007       EMP:NUM      C       002
   008       MONTANT      N       009        002
   009       NUM:NOTE     C       006
   010       CHECK:NUM    C       005
   011       CHECK:DATE   C       006
   **TOTAL**                    00086
```

dBASE II efface l'écran et liste les 16 premiers champs (ou moins) de la base de données. Utilisez **Ctrl-X** pour descendre d'un champ. Frappez simplement, sur l'ancien champ, le nouveau nom de champ. (Utilisez la barre d'espace pour effacer la lettre en trop).

Vous pouvez sortir de **MODIFY** de deux façons : **ctrl-W** change la structure sur disque puis reprend le cours nórmal des opérations de dBASE II, **ctrl-Q** abandonne et revient aux opérations normales de dBASE II sans faire les corrections. **ctrl-Q** vous ramène à l'ancienne structure de votre fichier sans détruire votre base de données, mais soyons prudent (voir page suivante).

COMMENT DUPLIQUER UNE BASE DE DONNEES ET LES STRUCTURES (commande COPY)

La duplication des fichiers sous dBASE II est directe; il ne vous est pas nécessaire de revenir au système. Frappez les lignes suivantes :

```
USE Noms
COPY TO Temp
USE Temp
DISPLAY STRUCTURE
LIST
   .use noms
   .copy to temp
   00010 ENREGISTREMENT(S) TRANSFERE(S)

   . use temp
   . display structure
   STRUCTURE DU FICHIER  : TEMP.DBF
   NOMBRE D'ENREGISTREMENTS : 00010
   DATE DE LA DERNIERE MISE A JOUR : 00/00/00
   BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
   CHAMP    NOM          TYPE    DIM   DECIMALE(S)

   001      NOM          C       020
   002      ADRESSE      C       025
   003      VILLE        C       025
   004      DEPARTMENT   C       002
   005      COD:POSTAL   C       005
   **TOTAL**                    00073
```

```
. list
00001  BIDONOT,JULES       45 rue des Chaines        Nantes     4444040
00002  ELECTRON,PAUL        2 rue des Pièces         Gallion    2727940
00003  FOUQUET,SEBASTIEN   68 rue de la Trésorerie   Chartres   2828000
00004  LAMOTTE,THEOPHILE    4 rue de la Laiterie     Chambéry   7373000
00005  LECHAT,CHARLES       5 avenue Lapatte         Montrond   4242210
00006  LEROY,JACQUES       20 rue des Pistes         Chambéry   7373000
00007  MARTIN,DIEGO       151 rue des Peupliers      Roubaix    5959057
00008  POUPON,PIERRE       12 Rue Beaudelaire        Angers     4949044
00009  PERDREAU,RENE       12 rue de la Chasse       Romantin   4141200
00010  ROSE,ANATOLE        26 rue du Tonneau         Bordeaux   3333200
```

Attention : Lorsque vous faites ^COPY^ vers un nom de fichier déjà existant, le nouveau fichier écrit par dessus l'ancien et vos anciennes données sont détruites.

COPY TO TEMP crée une nouvelle base de données appelée <Temp.DBF>. Ce fichier est identique au fichier <Noms.DBF>, même structure et mêmes données. La commande peut être :

COPY TO <nom-de-fichier> [STRUCTURE] [FIELDS liste]

Avec cette commande, vous pouvez *seulement* copier tout ou partie de la structure dans un autre fichier. Frappez les lignes suivantes :

```
USE Noms
COPY TO Temp STRUCTURE
USE Temp
DISPLAY STRUCTURE
    . use noms
    . copy structure to temp
    . use temp
    . display structure
    STRUCTURE DU FICHIER : TEMP.DBF
    NOMBRE D'ENREGISTREMENTS : 00000
    DATE DE LA DERNIERE A JOUR : 00/00/00
    BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
    CHAMP NOM          TYPE    DIM   DECIMALE(S)
    001   NOM          C       020
    002   ADRESSE      C       025
    003   VILLE        C       025
    004   DEPARTMENT   C       002
    005   COD:POSTAL   C       005
    **TOTAL**                 00073
```

Nous pouvons copier une partie de la structure en listant simplement les champs que nous voulons dans notre nouvelle base de données. Frappez :

```
USE Noms
COPY TO Temp STRUCTURE FIELDS Nom, Department
USE Temp
DISPLAY STRUCTURE
```

```
. use noms
. copy structure to temp fields nom, department
. use temp
. display structure
STRUCTURE DU FICHIER : TEMP.DBF
NOMBRE D'ENREGISTREMENTS : 00000
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP NOM          TYPE     DIM   DECIMALE(S)
001   NOM           C       020
002   DEPARTMENT    C       002
**TOTAL**                 00023
```

POUR LES PROGRAMMEURS AVERTIS : COPY peut aussi être utilisée pour permettre à un programme d'accéder à la structure d'une base de données. Frappez :

```
USE Noms
COPY TO Nouveau STRUCTURE EXTENDED
USE Nouveau
LIST
```

```
. USE noms
. copy to Nouveau structure extended
00006 ENREGISTREMENT(S) TRANSFERE(S)
```

```
. use nouveau
. display structure
STRUCTURE DU FICHIER : NOUVEAU.DBF
NOMBRE D'ENREGISTREMENTS :  00000
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
001   FIELD:NOM     C       010
002   FIELD:TYPE    C       001
003   FIELD:LEN     N       003
004   FIELD:DEC     N       003
**TOTAL**                 00018

  list
00001    NOM          C       20       0
00002    ADRESSE      C       25       0
00004    DEPARTMENT   C        2       0
00005    COD:POSTAL   C        5       0
00006    NUM:CLIENT   C        9       0
```

Les enregistrements de <Nouveau.DBF> décrivent la structure de la base de données <Noms> et un programme peut accéder directement à ces informations.

Un fichier qui a la même structure que <Nouveau.DBF> pourrait être inséré dans un programme pour que l'opérateur puisse entrer la structure de son fichier sans avoir à apprendre dBASE II. Ce programme pourrait alors lui créer sa base de données avec la commande suivante :

^CREATE <fichier-de-données> FROM <structure-de-fichier>^

COMMENT AJOUTER ET EFFACER DES CHAMPS DE DONNEES DANS UNE BASE

Au cours de vos applications avec dBASE II, vous voudrez probablement ajouter ou effacer des champs dans vos bases de données.

Le fait d'utiliser seulement **MODIFY STRUCTURE** pourrait détruire toutes les données de votre base, l'utilisation conjointe de **COPY** et d'**APPEND**, vous permettra d'ajouter et d'effacer des champs à volonté.

La technique consiste à copier la structure de la base de données que vous voulez changer dans un fichier temporaire, puis à effectuer vos modifications de structure dans ce dernier fichier. Lorsque vous avez terminé, vous transférerez les données de l'ancien fichier dans la nouvelle structure du nouveau fichier.

Comme exemple, nous allons utiliser notre fichier <Noms> et notre fichier <Commande>. A un certain moment, il peut être nécessaire de lister les commandes d'un certain client. Ceci peut être fait facilement en ajoutant un champ numéro de client dans le fichier <Noms>, nom de champ correspondant au champ du fichier <Commande>. Pour réaliser cette opération sans détruire les enregistrements qui ont déjà été saisis, frappez les lignes suivantes :

```
USE Noms
COPY TO Temp STRUCTURE
USE Temp
MODIFY STRUCTURE
^y^                        (en réponse au message)
```

Utilisez les caractéristiques de l'EDITION PLEIN ECRAN pour vous déplacer en bas, au premier champ blanc, et frappez les changements dans les colonnes appropriées (nom est « Num :Client », type de données est « C », longueur 9). Maintenant, frappez **ctrl-W** pour sauvegarder les corrections que vous avez faites et revenir au (.) de dBASE II.

^**DISPLAY STRUCTURE**^ pour vérifier les corrections. Si elles sont correctes, nous pouvons ajouter les données du fichier <Noms> en frappant :

```
APPEND FROM Noms
```

Nous aurions pu changer les dimensions des champs : la commande **APPEND** transfère les données dans les champs de noms correspondants.

```
. display structure
STRUCTURE DU FICHIER : TEMP.DBF
NOMBRE D'ENREGISTREMENTS : 00000
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP NOM          TYPE    DIM   DECIMALE(S)
001   NOM          C       020
002   ADRESSE      C       025
003   VILLE        C       025
004   DEPARTMENT   C       002
005   COD:POSTAL   C       005
006   NUM:CLIENT   C       009
**TOTAL**                00082
```

Notre nouveau fichier <Temp> devrait maintenant contenir le nouveau champ que nous avons voulu ajouter et toutes les données de l'ancien fichier. **DISPLAY STRUCTURE** puis **LIST** pour nous assurer qu'il n'y a pas d'erreur.

Si les données ont été transférées correctement, nous pouvons finir en frappant :

```
COPY TO Noms
USE Noms
```

La commande **COPY** écrit par dessus l'ancienne structure et les anciennes données. Après avoir visualisé et listé le nouveau fichier <Noms>, vous pouvez faire **DELETE FILE Temp.**

En résumé, la procédure peut être utilisée pour ajouter ou effacer des champs dans une base de données d'après la séquence suivante :

```
USE <ancien-fichier>
COPY TO <nouveau-fichier> STRUCTURE
USE <nouveau-fichier>
MODIFY STRUCTURE
APPEND FROM <ancien-fichier>
COPY TO <ancien-fichier>
```

```
. use Noms
. copy to temp structure
. use temp
. modify structure
LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES (Y/N) ? ^y^

. append from Noms
00010 ENGEGISTREMENT(S) TRANSFERE(S)

.copy to Noms
00010 ENREGISTREMENT(S) TRANSFERE(S)
. use noms
. display structure
STRUCTURE DU FICHIER : NOMS.DBF
NOMBRE D'ENREGISTREMENTS : 00010
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP  NOM            TYPE       DIM    DECIMALE(S)
001    NOM            C          020
002    ADRESSE        C          025
003    VILLE          C          025
004    DEPARTMENT     C          002
005    COD:POSTAL     C          005
006    NUM:CLIENT     C          009
**TOTAL                         00082
```

COMMENT TRAVAILLER AVEC CP/M ET DES FICHIERS « ETRANGERS »

Les informations de dBASE II peuvent être transcrites sous une autre forme afin d'être compatibles avec d'autres machines et d'autres systèmes (BASIC, PASCAL, FORTRAN, PL/1, etc.). dBASE II peut également lire les fichiers de données créés par ces systèmes.

Avec CP/M, le format standard des données (abrégé en **SDF** sous dBASE II), inclut un retour chariot et un interligne après chaque ligne de texte. Pour créer un fichier de données compatible au traitement de texte, vous utiliserez une autre forme de la commande **COPY**. Frappez :

```
USE Noms
COPY TO SysData SDF
```

Cette commande crée un fichier appelé <SysData.TXT>. Maintenant frappez **QUIT** pour quitter dBASE II et utilisez votre programme de traitement de texte pour examiner ce fichier. Vous constaterez que vous pouvez l'utiliser exactement comme si vous l'aviez créé sous CP/M.

Le format standard de données permet également à dBASE II de travailler avec des données provenant de fichiers CP/M. Cependant, *la structure des données doit correspondre à la structure des bases de données qui seront utilisées.*

Si nous avions utilisé un programme de traitement de texte pour créer un fichier appelé <NouvData.TXT>, nous aurions pu ajouter ces données au fichier <Noms.DBF> avec cette commande. A NOTER : les espaces séparant les données doivent correspondre à la structure des bases de données. Si le fichier <NouvData.TXT> contient les informations suivantes :

```
LATEMPETE,EMMANUEL    4 rue du Port          Brest       2929240
DUTRONC,JULES         5 rue César            Ladoux      2121550
LEROUGE,ALAIN        17 rue la Révolution    Montereau   7777130
BELLEVUE,ANATOLE      4 rue du Belvédère     Oxonnax     0101100

(20)                    (25)                  (25)        (2) (5)
```

nous pourrons ajouter ces informations au fichier <Noms> en frappant les lignes suivantes :

```
  USE Noms
APPEND FROM NouvData.TXT SDF
```

La procédure est la même si votre fichier « étranger » utilise différents délimiteurs.

Un format courant de fichier de données utilise des virgules entre les champs et des apostrophes autour des chaînes pour séparer les données. Pour créer ou utiliser ce type de fichier de données, utilisez le mot DELIMITED au lieu de SDF. Pour voir son fonctionnement, frappez :

```
^COPY TO Temp DELIMITED^
```

puis revenez au système et regardez vos données.

Si votre système a différents délimiteurs, vous pouvez spécifier ceux-ci dans la commande par ^DELIMITED [WITH <**pointeur**>] (ne pas frapper les symboles « < » et « > »). Si votre système utilise uniquement des virgules et rien autour des chaînes, utilisez ^DELIMITED WITH ,^.

Les formes complètes de COPY et APPEND sont les suivantes :

```
                                                    [SDF]
COPY [étendue] TO <nom-de-fichier> [FIELD list] [STRUCTURE] [FOR <expression>]
                                                    [DELIMITED  [WITH <pointeur>]]

      APPEND FROM <nom-de-fichier.TXT> [SDF]  [FOR <expression>]
                                       [DELIMITED [WITH <pointeur>]]
```

Ces deux commandes peuvent être sélectives grâce à l'utilisation d'expressions conditionnelles. L'étendue de COPY peut être spécifiée comme pour les autres commandes de dBASE II.

NOTE : Si dBASE II génère automatiquement des extensions pour les fichiers créés, vous devez spécifier l'extension « .TXT » au nom de fichier que vous ajoutez d'un autre système.

NOTE : Avec la commande APPEND, chaque champ utilisé dans <expression> doit exister dans la base de données pour permettre le transfert des données.

COMMENT RENOMMER LES CHAMPS DES BASES DE DONNEES AVEC COPY ET APPEND

Ainsi que nous l'avons déjà mentionné, **APPEND** transfère les données d'un fichier vers un autre dans des champs de noms correspondants. Si un champ de nom dans le fichier FROM n'est pas dans le fichier en USE, les données de ce champ ne seront pas transférées.

Cependant, la commande complète nous permet quand même de transférer seulement des données et nous pouvons utiliser cette caractéristique pour renommer les champs de nos bases de données. Si nous voulons renommer <num :client> en <cod :Client> dans notre fichier <Noms.DBF>, nous devons frapper :

```
USE Noms
COPY TO Temp SDF            (les données seulement dans Temp.TXT)
MODIFY STRUCTURE
APPEND FROM Temp.TXT SDF    (après avoir changé le champ de nom)
```

Maintenant lorsque vous faites **DISPLAY STRUCTURE**, le dernier champ s'appelle <cod :client>. N'oubliez pas de changer le nom du champ de <num :Client> dans votre fichier <Commandes> afin d'avoir des noms de champs correspondants.

```
. use noms
. copy to temp sdf
00014 ENREGISTREMENT(S) TRANSFERE(S)

. modify structure
LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES (Y/N) ? Y

. append from temp.TXT sdf
00015 ENREGISTREMENT(S) TRANSFERE(S)
```

Les données d'un fichier <.TXT> (créé par l'utilisation et l'option **SDF** ou **DELIMITED**) ont la même structure que celles du fichier original. Vous pouvez éditer un fichier <.TXT> avec votre traitement de texte mais ceci peut être dangereux.

Attention : Ne pas changer la position des champs ni leur dimension : les données sont sauvegardées par position et non par nom. Si vous changez la dimension d'un champ lors de la modification de la structure, vous détruirez vos données lorsque vous voudrez les ramener dans votre fichier.

Lorsque vous faites une copie des données vers un fichier <.TXT>, vous pouvez utiliser la commande complète pour spécifier *l'étendue, les champs* et *les conditions* (voir les explications précédentes).

COMMENT MODIFIER RAPIDEMENT VOS DONNEES (REPLACE, CHANGE)

Les modifications peuvent être faites rapidement pour tout ou partie des enregistrements en utilisant la commande suivante :

REPLACE [étendue] <champ> WITH <donnée> [, <champ> WITH <donnée>,...]
[FOR <expression>]

Cette commande est extrêmement puissante puisqu'elle remplace un champ que vous nommez, par tout ce que vous voulez y écrire. Vous pouvez remplacer plus d'un champ en mettant une virgule après la première combinaison. Les données peuvent être de nouvelles informations spécifiques y compris les blancs, ou un calcul à effectuer. Exemple :

```
REPLACE all Montant WITH Montant/1.06
```

Vous pouvez également faire des remplacements conditionnels par l'utilisation de FOR en spécifiant vos conditions dans <expression>.

Afin de vous en montrer le fonctionnement, nous avons besoin d'ajouter quelques données dans les deux fichiers de base de données <Noms.DBF> et <Commande.DBF>.

Frappez tout d'abord **USE Nom**, puis **EDIT 1**. Maintenant, entrez dans le champ <Cod :Client> **1001** en utilisant l'édition plein écran pour vous positionner à l'endroit voulu. Faites **ctrl-C** pour vous déplacer à l'enregistrement suivant. Le code client est entré avec une longueur de quatre chiffres. Le numéro de l'enregistrement sera les deux derniers chiffres (exemple : 1001, 1002, 1003, etc.).

Maintenant, frappez **USE Commande** et **APPEND** pour entrer les informations suivantes concernant le fichier Commande (ne pas frapper les titres des colonnes) :

```
(client) (arti)    (Qty)     (prix)
 1012    38567       5        .83
 1003    83899      34        .12
 1009    12829       7        .17
 1012    73833      23       1.47

USE Commande
REPLACE All Montant WITH Quantite*Prix
LIST

   . USE Commande
   . REPLACE All montant with quantite*prix
   00004 REMPLACEMENT(S)
   . list
   00001     1012    38567       5       0.83      4.15
   00002     1003    83899      34       0.12      4.08
   00003     1009    12829       7       0.17      1.19
   00004     1012    78833      23       1.47     33.81
```

La commande **REPLACE** peut également être utile dans un fichier de commandes pour mettre à blanc un enregistrement que vous avez ajouté dans un fichier. Dans vos programmes, les données de vos variables mémoire sont fréquemment utilisées pour remplir ces champs à blanc.

Pour un grand nombre d'enregistrements, les changements de champs peuvent s'effectuer rapidement par l'utilisation de :

CHANGE [étendue] FIELD <list> [FOR <expression>]

L'étendue est la même que pour les autres commandes de dBASE II. Il faut nommer au moins un champ, mais plusieurs champs peuvent être listés s'ils sont séparés par des

virgules. Cette commande sélectionne l'enregistrement correspondant à <expression> et le visualise ainsi que son contenu. Pour changer les données du champ, entrez-y de nouvelles informations. Pour le laisser tel qu'il était, appuyez sur la touche <Enter>. *Si vous voulez entrer des données dans un champ à blanc, faites un espace.*

Pour revenir à dBASE II, appuyez sur la touche ESCAPE.

```
. use Noms
. change field cod:client

ENREGISTREMENT: 00001

COD:CLIENT
CHANGE?^ ^(ENTRER UN ESPACE POUR CHANGER UN CHAMP VIDE )
TO      1001

COD:CLIENT :1001
CHANGE?<enter>

ENREGISTREMENT: 00002

COD:CLIENT :
CHANGE?
```

RAPPEL : Si vous avez un nombre d'enregistrements relativement petit, la commande BROWSE peut vous être utile (voir plus loin).

COMMENT ORGANISER VOS BASES DE DONNEES (SORT, INDEX)

Les données sont fréquemment entrées directement, comme nous l'avons fait pour notre fichier de données <Noms.DBF>. Cette méthode peut ne pas vous convenir, aussi dBASE II vous permet d'organiser vos bases de données par **SORT** et **INDEX**.

Les fichiers **INDEX** vous permettent de localiser rapidement vos enregistrements. Les fichiers peuvent être triés en ordre croissant ou décroissant. La commande complète est la suivante :

```
SORT ON <nom-de-champ> TO <nom-de-fichier> [DESCENDING]
```

Le <nom-de-champ> est la clé sur laquelle le fichier est trié et peut être de type caractère ou numérique, mais pas logique. Par défaut, le tri se fera par ordre croissant mais vous pouvez l'effectuer en ordre décroissant en spécifiant l'option DESCENDING.

Si vous voulez trier sur plusieurs clés, commencez par la clé la moins importante puis utilisez une série de clés qui vous mène jusqu'à la clé principale.

Afin de classer nos clients du fichier <Noms> par ordre alphabétique, frappez les lignes suivantes :

```
USE Noms
SORT ON Nom TO Temp
USE Temp
LIST
COPY TO Noms
```

```
. use noms
. sort on nom to temp
*** TRI TERMINE ***

. use temp
. list
00001  BELLEVUE,ANATOLE     4 rue du Belvédère  Oxonnax    0101100 1001
00002  BIDONOT,JULES       45 rue des Chaines    Nantes     4444040 1002
00003  DUTRONC,JULES        5 rue César          Ladoux     2121550 1003
00004  ELECTRON,PAUL        2 rue des Pièces     Gallion    2727940 1004
00005  FOUQUET,SEBASTIEN   68 rue la Trésorerie  Chartres   2828000 1005
00006  LAMOTTE,THEOPHILE    4 rue la Laiterie    Chambéry   7373000 1006
00007  LATEMPETE,EMMANUEL   4 rue du Port        Brest      2929240 1011
00008  LECHAT,CHARLES       5 avenue Lapatte     Montrond   4242210 1012
00009  LEROUGE,ALAIN       17 rue la Révolution  Montereau  7777130
00010  LEROY,JACQUES       20 rue des Pistes     Chambéry   7373000 1007
00011  MARTIN,DIEGO       151 rue des Peupliers  Roubaix    5959057 1008
00012  PERDREAU,RENE       12 rue de la Chasse   Romantin   4141200 1009
00013  POUPON,PIERRE       12 Rue Beaudelaire    Angers     4949044 1010
00014  ROSE,ANATOLE        26 rue du Tonneau     Bordeaux   3333200
. copy to noms
00014   ENREGISTREMENT(S) TRANSFERE(S)
```

ATTENTION : Ne jamais trier une base de données sur elle-même. Vous pourriez la détruire par une fausse manipulation.

Il vaut mieux faire votre tri sur un fichier temporaire, puis le recopier dans le fichier original.

Une base de données peut aussi être indexée afin qu'elle apparaisse triée. Voici la forme de la commande **INDEX** :

^INDEX ON <clé (variable/expression)> TO <index nom-de-fichier>^

Cette opération crée un nouveau fichier avec une extension .NDX. Seules les données comprises dans la clé sont triées. La clé peut être un nom de variable ou une expression complexe avec une longueur maximale de 100 caractères. Ce ne peut pas être un champ logique. Afin d'organiser notre base de données de noms par code postal, frappez :

```
USE Noms
INDEX ON Cod:postal TO Codes
USE Noms INDEX Codes
LIST
```

Afin d'indexer notre base de données sur trois clés, frappez :

INDEX ON Nom + Cod:Client + Department TO Composé

Les champs numériques utilisés de cette manière doivent être convertis en type caractère. Si le Cod :Client est un champ numérique de 5 positions avec 2 décimales, la fonction **STR** (décrite ultérieurement) exécute la conversion suivante :

INDEX ON Nom + STR(Cod:Client,5,2) + Department TO Composé

Afin de bénéficier de la vitesse d'un fichier indexé, nous devons le spécifier dans la commande **USE** :

USE <nom base-de-données> INDEX <nom-du-fichier indexé>

Les commandes de positionnement GO, GO BOTTOM, etc., utilisées avec un fichier indexé, vous positionnent sur l'index plutôt que sur la base de données. Exemple, la commande **GO BOTTOM** vous positionnera sur l'enregistrement correspondant au dernier index au lieu du dernier enregistrement de votre base de données.

Les corrections réalisées sur les champs correspondant à des clés quand vous exécutez les commandes **APPEND, EDIT, REPLACE** ou **PACK** de vos bases de données, sont répercutées dans le fichier indexé.

Après modification ou création d'enregistrements, il est nécessaire de mettre à jour votre fichier index. Pour réaliser cette opération, frappez la commande suivante : **SET INDEX TO <index fichier 1>, <index fichier 2>, <index fichier n>.**

Un fichier indexé vous permet d'utiliser la commande **FIND** pour localiser vos enregistrements en peu de secondes, même si vous avez une grande base de données.

COMMENT TROUVER L'INFORMATION QUE VOUS VOULEZ (FIND, LOCATE)

Si vous savez ce que vous voulez rechercher comme donnée, vous pouvez utiliser la commande **FIND**, à condition que votre base de données soit indexée et en USE.
Tapez simplement **FIND <chaîne de caractères>** sans apostrophes, où « chaîne de caractères » est tout ou partie du contenu d'un champ.
La chaîne peut être courte mais doit être assez longue pour la rendre unique. Tapez les lignes suivantes :

```
    USE Noms INDEX Cod:postal
    FIND 01
    DISPLAY
    FIND 27
    DISPLAY
    DIPLAY Next 2

. use noms index cod:postal
. find 01
. display
00001  BELLEVUE,ANATOLE     4 rue du Belvédère  Oxonnax    0101100 1001

. find 27
. display
00004  ELECTRON,PAUL        2 rue des Pièces    Gallion    2727940 1004

. display next 2
*** ENREGISTREMENT NON TROUVE ***
```

Si la clé n'est pas unique, dBASE II trouve le premier enregistrement correspondant à vos spécifications. Cet enregistrement peut être ou non celui que vous cherchez. Si aucun enregistrement n'existe sous la clé définie, dBASE II vous répondra **ENREGISTREMENT NON TROUVE.**

La commande **FIND** peut également être utilisée avec des fichiers qui ont été indexés avec des clés multiples. L'inconvénient d'une clé composée est qu'elle fonctionne de la gauche vers la droite et vous devez spécifier la clé composée complète pour accéder à vos données.
Lorsque vous recherchez un type de données spécifique, utilisez la commande :

LOCATE [étendue] [FOR <expression>]

Cette commande est utilisée pour rechercher des données spécifiques dans un fichier qui n'a pas été indexé sur la clé qui vous intéresse (exemple, le fichier a été indexé sur le code postal mais vous êtes intéressé par les départements, etc...).

LOCATE commençant sur le premier enregistrement, il n'est pas obligatoire d'en spécifier l'étendue si vous voulez effectuer la recherche sur la base de données entière. Pour rechercher une partie du fichier, utilisez **LOCATE Next <numéro>**. La recherche commence à l'enregistrement où se trouve le pointeur et contrôle le numéro de l'enregistrement suivant.

Si vos recherches portent sur des champs d'enregistrements dont les données sont des caractères, ces données doivent être délimitées par des apostrophes. Tapez les lignes suivantes :

```
USE Noms
LOCATE FOR Nom='POU'
DISPLAY
LOCATE FOR Cod:Postal>'4' .AND. Nom < 'P'
DISPLAY Nom, Cod:Postal
```

Si un enregistrement répond aux conditions de votre expression, dBASE II vous signale : **ENREGISTREMENT n.** Vous pouvez visualiser ou éditer l'enregistrement une fois qu'il a été localisé.

Dans l'éventualité où plusieurs enregistrements répondent à vos conditions, frappez **CONTINUE** pour avoir le numéro d'enregistrement suivant.

```
CONTINUE
CONTINUE
CONTINUE
```

Si dBASE II ne peut pas trouver votre enregistrement à l'intérieur de l'étendue définie, il vous dira : **FIN DE LOCALISATION** ou **FIN DE FICHIER.**

```
. use NOMS
. locate for nom='POU'
ENREGISTREMENT: 00013

. display
00013  POUPON,PIERRE      12 Rue Beaudelaire    Angers      4949044 1010

. locate for cod:postal >'5' .and. Nom <'M'
ENREGISTREMENT: 00006
. display Nom,Cod:Postal
00006  LAMOTTE,THEOPHILE    73000

. continue
ENREGISTREMENT: 00009

. continue
ENREGISTREMENT: 00010

. continue
*** FIN DE FICHIER ***
```

COMMENT EXTRAIRE LES INFORMATIONS DE LA BASE DE DONNEES (REPORT)

Les commandes **FIND** et **LOCATE** localisent des enregistrements individuels ainsi que des données. Mais pour la plupart des applications, vous souhaiterez peut-être un résumé des données qui réunisse les enregistrements remplissant une certaine condition. La commande **REPORT** vous en facilite la tâche.

Si vous utilisez une simple feuille de papier à l'imprimante, frappez d'abord **SET EJECT OFF** afin d'arrêter l'alimentation des feuilles. Maintenant choisissez la base de données sur laquelle vous désirez travailler pour obtenir des rapports et créer votre format d'état pour clients :

```
SET EJECT OFF
USE <nom-du-fichier>
REPORT
```

dBASE II vous pose une série de questions pour créer le format de l'état de sortie. Vous spécifiez ce que vous voulez dans l'état : quels sont les champs, quel est le titre de l'état, quels sont les titres des colonnes et quelles colonnes doivent être totalisées, etc. Vous avez par défaut 8 colonnes à partir du côté gauche de la page, 56 lignes par page, et une largeur de page de 80 caractères.

Vous allez maintenant essayer de créer un état à partir des fichiers <Noms.DBF> et <Commande.DBF> que vous avez créés sur votre disque de travail. Pour nos exemples, nous allons désormais utiliser le fichier <Depense.DBF> et d'autres bases de données qui font partie d'un système commercial déjà existant.

```
. use depense
. report
DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : Jobcouts
OPTIONS, M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR DE PAGE <RETURN>
DESIREZ-VOUS UN ENTETE ? (Y/N) Y
DONNEZ L'ENTETE : RESUME DES COUTS
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? n
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
DESIREZ-VOUS DES SOUS-TOTAUX DANS VOTRE RAPPORT (Y/N) ? n
COLONNE   LONGUEUR, CONTENU
001   10,D:Cheque
DONNEZ LE TITRE : DATE
002   22,Nom
DONNEZ LE TITRE : FOURNISSEUR
003   22,Descrip
DONNEZ LE TITRE : DESCRIPTION
004   12,Montant
DONNEZ LE TITRE : MONTANT
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
005   <enter>
         PAGE N. 00001
```

Lorsque vous avez terminé l'entière définition du contenu de votre état, appuyez sur la touche <enter> quand le curseur est au numéro de champ suivant. dBASE II commence immédiatement l'état pour vous montrer ce que vous avez spécifié et vous fait défiler l'entière base de données si vous le laisser faire. Pour arrêter le défilement de l'état, appuyez sur la touche **escape**.

dBASE II sauvegarde le format de cet état dans un fichier d'extension **.FRM** pour que vous puissiez l'utiliser à nouveau sans avoir à en redéfinir le format. La commande complète est la suivante :

REPORT FORM <nom-format> [étendue] [FOR <expression>] [TO PRINT]

En frappant :

```
REPORT FORM jobcouts FOR Num:trav=770
```

nous pouvons avoir un listing avec tous les coûts qui ont un numéro de travail 770, sans avoir à redéfinir le format.

```
. REPORT FORM jobcouts FOR num:trav='770'

PAGE N. 00001
                              RESUME DES COUTS

 DATE        FOURNISSEUR          DESCRIPTION          MONTANT

810113       TATAPE               SAISIES               177.00
810113       CIE COURRIER         ENVOIES               605.00
810113       MARSHALL, RENEE      SAISIES                37.10
810113       MARSHALL, RENEE      NETTOYAGE             200.00
810113       SHUTTERBUGS, INC     PHOTOGRAPHIE          565.00
810113       MAGIC TOUCH          RETOUCHE               56.00
**TOTAL**                                              1640.10
```

Nous pouvons également changer les informations du titre en frappant **SET HEADING TO chaîne de caractères** (jusqu'à 60 caractères et espaces, mais pas d'apostrophes). Si « étendue » n'est pas spécifiée, par défaut ce sera « tout ».

COMMENT TOTALISER ET COMPTER AUTOMATIQUEMENT (COUNT, SUM)

Pour certaines applications, vous ne souhaiterez peut-être pas visualiser les enregistrements en cours mais vous voudrez seulement savoir combien d'enregistrements répondent à un certain critère et quels sont les totaux pour une certaine condition.

Pour compter, utilisez la forme suivante :

COUNT [étendue] [FOR <condition>] [TO variable mémoire]

Le résultat du calcul sera stocké dans la variable mémoire créée lors de l'exécution de la commande. Pour obtenir les totaux, utilisez la forme suivante :

SUM champ(s) [étendue] [FOR condition] [TO variable(s) mémoire]

Vous pouvez lister au maximum 5 champs numériques pour les totaux de vos bases de données en USE. Si vous avez plus d'un champ à totaliser, les noms des champs doivent être séparés par des virgules. Les enregistrements à totaliser peuvent être limités par l'utilisation de étendue et/ou des expressions conditionnelles après FOR.

Si des variables mémoire sont utilisées (séparées par des virgules), souvenez-vous que les totaux sont stockés par position. Vous pouvez remettre les variables mémoire à blanc par la commande RELEASE.

```
. USE depense
. COUNT FOR Montant <100 TO petit
COMPTEUR = 00067

. SUM Montant FOR num:trav=770 TO Cout
 1640.10

. display memory
PETIT                   (N)              67
COUT                    (N)            1640.10
**TOTAL**   02 VARIABLES USED   00012 BYTES USED
```

COMMENT TOTALISER LES DONNEES ET ELIMINER LES DETAILS (TOTAL)

La commande TOTAL fonctionne de la même façon que le sous-total de la commande REPORT. La seule différence est que les résultats seront placés dans la base de données plutôt qu'imprimés. Voici la commande :

TOTAL ON <clé> TO <base-de-données> [FIELDS list] [FOR conditions]

NOTE : Afin de pouvoir utiliser cette commande, les informations qui proviennent des bases de données doivent être pré-triées ou indexées.

Exemple :

```
USE Depense
INDEX ON Num:Trav TO Travails
USE Depenses INDEX Travails
TOTAL ON Num:Trav TO Temporaire FIELDS Montant FOR Num:Trav > 699;
                                                .AND. Num:Trav < 800
USE Temporaire
LIST
```

La nouvelle base de données possède un numéro de travail comme entrée et un total de tous les coûts pour ce numéro de travail. Nous avons cependant un problème avec cette nouvelle base qui n'a que deux champs utiles.

Nous pouvons régler ce problème par une ligne de commande supplémentaire. La commande TOTAL transfère tous les champs si la base de données citée n'existe pas, mais utilise la structure d'une base de données existante. Dans la commande citée plus haut, nous aurions pu limiter les champs de notre nouvelle base de données en la créant d'abord avant d'utiliser la commande TOTAL.

COPY TO Temporaire FIELDS Num:Trav, Montant

De ce fait, quand nous exécutons la commande **TOTAL** to <Temporaire>, la nouvelle base de données contient uniquement le numéro de travail et les totaux.

Cette méthode peut aussi être utilisée pour totaliser une partie des quantités, les comptes crédit ou autres informations. (Rappel : les informations doivent toujours être triées ou indexées).

```
. USE Depense
. INDEX ON num:trav TO Travails
00093 ENREGISTREMENT(S) INDEXE(S)

. USE Depense INDEX Travails
. TOTAL ON num:trav TO Temp FIELDS Montant FOR num:trav > 699;
                                          .AND. num:trav < 800
00025 ENREGISTREMENT(S) TRANSFERE(S)

. USE Temp

. LIST
00011  810129   3148   SML    779    138.00   TATAPE          SAISIES
810129  2633         0.00      0
00012  810129   3152   SML    782     59.49   MAGIC TOUCH     RETOUCHE
TONE        810129    429        0.00      0
00013  810129   3148   SMM    784     46.00   TATAPE          SAISIES
810129  3003         0.00      0
00014  810129   3148   DOC    786    251.00   TATAPE          SAISIES
810129  2764         0.00      0
                            (listing partiel)
```

RESUME DU CHAPITRE II

Dans ce chapitre, nous vous avons montré une grande partie de ce que vous pouvez faire avec dBASE II.

Nous avons vu que les différents opérateurs arithmétiques, relationnels et chaîne peuvent être utilisés pour modifier les commandes de dBASE; ils permettront de contrôler toutes vos données.

Puisque la structure des données est fondamentale pour tout système de base de données, nous avons vu différentes méthodes nous permettant de modifier la structure de nos fichiers avec ou sans données.

Nous vous avons également expliqué comment entrer, modifier et trouver une information spécifique. Nous avons passé en revue quelques commandes globales telles que COUNT, SUM, REPORT et TOTAL.

Dans la section suivante, nous allons vous montrer comment on écrit un fichier de commandes (programme) sous dBASE II.

CHAPITRE III

CONCEPTION D'UN FICHIER DE COMMANDES

Si vous avez compris comment s'écrivent les expressions, vous êtes maintenant capable d'écrire des programmes.

Voici **les quatre structures de base pour la programmation** :

- Séquence
- Choix/Décision
- Répétition
- Procédures

Vous avez sans doute remarqué que dBASE II exécute vos commandes séquentiellement dans l'ordre où vous les avez entrées. Dans cette section, nous allons vous expliquer comment faire vos choix (**IF...ELSE**), comment demander à l'ordinateur de répéter une séquence de commandes (**DO WHILE..**), et comment utiliser des sous-fichiers de commandes (procédures).

COMMENT MONTER UN FICHIER DE COMMANDES (écriture d'un programme)

Les commandes qui vous ont été présentées jusqu'à présent sont très puissantes et peuvent accomplir de très grandes tâches. Toutefois, nous n'avons vu qu'une partie des capacités de dBASE II.

Lorsque vous créez un fichier de commandes, vous faites déjà de la programmation. Vous allez voir qu'avec dBASE II, c'est beaucoup plus facile que vous ne le pensez. Les fichiers de commandes ont une extension <**CMD**>.

Pour créer un fichier de commandes, vous frappez les commandes que vous voulez exécuter dans un fichier créé avec un éditeur ou un traitement de texte. dBASE II exécute séquentiellement une à une les commandes en commençant par le début jusqu'à la fin de la liste. dBASE II considère le retour chariot comme la fin d'une ligne de commande. Dans certains cas, l'ordre séquentiel peut être interrompu puisque, dans nos commandes, nous avons demandé à l'ordinateur d'aller exécuter d'autres procédures sous certaines conditions. Plus de précisions vous seront données ultérieurement.

Mais à présent, créons un fichier de commandes appelé <Test>. Vous pouvez faire cette opération en utilisant un éditeur de texte ou un traitement de texte, mais il est plus facile de le faire sous dBASE II en frappant :

```
MODIFY COMMAND Test
```

dBASE II vous présente alors un écran blanc sur lequel vous pouvez déjà commencer à écrire le programme ci-dessous.

La fin d'une ligne indique la fin d'une commande à moins que vous n'utilisiez des « ; ». Frappez la liste de commandes telle que ci-dessous :

```
USE Noms
COPY Structure TO Temp FIELDS Nom, CodPostal
USE Temp
APPEND FROM Noms
COUNT FOR Nom = 'P' TO P
DISPLAY MEMORY
? 'Nous venons de créer brillamment notre premier fichier de commandes.'
```

Lorsque vous avez terminé, faites **ctrl-W** pour sauvegarder votre fichier et revenir au (.) de dBASE II. Frappez maintenant :

```
Do Test
```

Si vous avez frappé le programme exactement de la façon dont il a été imprimé, le programme ne s'exécutera pas. Maintenant frappez à nouveau **MODIFY COMMAND Test** et insérez un deux-point (:) pour corriger le champ de nom <Cod :Postal>.

Lorsque vous aurez à programmer vous-même de plus grands fichiers de commandes, vous trouverez utile cet éditeur incorporé à dBASE II car il vous permettra d'écrire, de corriger et de changer vos programmes sans avoir à revenir chaque fois au système. Cet éditeur incorporé peut contenir 5.000 caractères environ.

Le fichier de commandes que vous venez de frapper est en lui-même insignifiant mais vous montre toutefois comment vous pouvez exécuter une séquence de commandes à partir d'un fichier avec une commande.

L'exécution de cette commande est semblable à celle que vous utilisez avec les fichiers COM sous votre système d'exploitation.

Si vous n'êtes pas sous dBASE II, vous pouvez frapper dBASE Test directement à partir du système pour exécuter votre fichier de commandes.

ASTUCE : Vous pouvez renommer le fichier principal de dBASE II en **DO.COM**. De cette façon, vous pouvez frapper **DO <nom-de-fichier>** que vous soyez sous votre système ou sous dBASE II. Afin d'exécuter cette opération avec CP/M, frappez **REN DO.COM dBASE.COM**.

COMMENT FAIRE LES' CHOIX ET LES DECISIONS (IF..ELSE)

Sous dBASE II, **les choix et décisions** se font avec **IF..ELSE..ENDIF**. Ce choix a la signification suivante : **IF** J'ai faim, je mange, **(OR) ELSE** je ne mange pas.

LES DECISIONS SIMPLES. Si une décision simple est à prendre, vous pouvez laisser tomber le **ELSE** et utiliser la forme suivante :

```
IF condition [.AND. cond2 .OR. cond3 ...]
   do cette commande
   [cmd2]
ENDIF
```

La condition peut être une série d'expressions contenant un maximum de 254 caractères. Cette condition peut être évaluée logiquement comme étant vraie ou fausse. Utilisez les opérateurs logiques pour les lier ensemble. En utilisant notre fichier <Depense>, nous pouvons prendre les décisions en frappant :

```
IF Num:Trav = '730' .AND. Montant > 99.99;
       .OR.    Fournisseur = 'MAGIC TOUCH';
       .OR.    Date:factu > '791231'
   do cette commande
   [cmd 2]
   [ ... ]
 ENDIF
```

Si *toutes* les conditions sont remplies, l'ordinateur exécute en séquence les commandes listées entre le IF et le ENDIF puis la phrase suivant le ENDIF. Si aucune condition n'est vérifiée, l'ordinateur passe à la première commande suivant le ENDIF.

DEUX CHOIX. Si l'action dépend de deux choix, utilisez le IF..ELSE de la façon suivante :

```
IF condition(s)
   do commande(s) 1
ELSE
   do commande(s) 2
ENDIF
```

En fonction de la réponse, l'ordinateur exécute soit le premier soit le deuxième jeu de commandes, puis passe à la commande suivant le ENDIF.

CHOIX MULTIPLES. Fréquemment, vous avez à prendrre des décisions d'après une série d'alternatives. Exemple, dans un menu où vous devez choisir une ou plusieurs procédures différentes à exécuter. Dans ce cas, utilisez la construction IF..ELSE..IF.

Cette construction est la même que IF..ELSE décrite précédemment, mais nous utilisons plusieurs niveaux. C'est ce qu'on appelle un IF imbriqué.

```
IF conditions 1
   do commandes 1
ELSE
   IF conditions 2
      do commandes 2
   ELSE
      IF conditions 3
         do commandes 3
      ELSE
         .
         .
         .
      ENDIF 3
   ENDIF 2
ENDIF 1
```

Pour que votre programme s'exécute correctement, faites impérativement correspondre à chaque IF un ENDIF.

CONSEIL : Après un ENDIF, dBASE II ne lit pas les mots écrits sur la même ligne. Aussi, pouvez-vous ajouter à côté du ENDIF les remarques que vous voulez.

COMMENT REPETER UN TRAITEMENT (DO WHILE..)

La répétition est l'un des avantages majeurs des ordinateurs. La machine peut exécuter éternellement la même tâche sans s'ennuyer, ni faire des erreurs par monotonie. La répétition peut se faire avec l'instruction **DO WHILE** :

```
DO WHILE conditions
   do commande(s)
ENDDO
```

L'exécution des commandes se fera si les conditions que vous avez spécifiées sont logiquement vraies.

CONSEIL : Rappelez-vous que l'exécution des commandes doit modifier éventuellement la réponse aux conditions, sinon votre programme bouclera éternellement. Si vous savez combien de fois la procédure doit être répétée, vous pouvez utiliser la structure suivante :

```
STORE 1 TO Index                    * Le compteur commence à 1
DO WHILE Index < 11                 * Nous traitons 10 enregistrements
   IF Article = ' '                 * Il n'y a pas de données,
      SKIP                          * laissons passer cet enregistrement
      LOOP                          * nous revenons en arrière pour
                                    * exécuter le DO WHILE
   ENDIF blanc                      * sans exécuter Proced:A
   DO Proced:A                      * Exécutons le Proced:A.CMD
   STORE Index+1 TO Index           * Incrémente notre compteur d'1
ENDDO dix fois
```

Dans cet exemple, si le champ <Article> contient des données, l'ordinateur exécute le fichier de commandes appelé Proced :A.CMD, puis revient à son point de départ en augmentant la valeur de la variable Index de 1, et teste ensuite si cette valeur est inférieure à 11. L'ordinateur va exécuter le DO WHILE dix fois. Quand le compteur

dépassera 10, l'ordinateur sortira de la boucle et exécutera l'instruction qui suit immédiatement le **ENDDO**.

L'instruction **LOOP** est utilisée pour interrompre une séquence et pour dire à l'ordinateur de revenir en arrière, là où commence la commande **DO WHILE**.

Dans notre cas, si le champ <Article> est à blanc, l'enregistrement ne sera pas traité car, par la commande LOOP, nous revenons en arrière, au **DO WHILE Index < 11**. L'enregistrement à blanc ne sera pas compté puisque nous avons sauté la ligne de commande où nous ajoutions 1 au compteur.

LOOP pose un problème car il court-circuite le déroulement du programme, d'où la difficulté de suivre la logique du programme. La meilleure solution est d'éviter l'utilisation de cette instruction.

LES PROCEDURES (les fichiers de commandes annexes)

La possibilité de créer des procédures standard dans un langage simplifie énormément la programmation.

En BASIC, ces procédures sont appelées « sub-routines ». En Pascal et PL/I, elles sont appelées « Procédures ». En dBASE II, il existe des fichiers de commandes qui peuvent être appelés par un programme écrit par vous. Dans l'exemple précédent, nous avons fait appel à une procédure en utilisant *Do Proced :A*. « Proced :A » est un fichier de commandes avec une extension .CMD. Le contenu de ce fichier de commandes aurait pu être :

```
IF Statut = F
   DO Payfone
ELSE
   IF Statut = T
      DO Paytele
   ELSE
      IF Statut = M
         DO Payloyer
      ENDIF
   ENDIF
ENDIF
RETURN
```

Cette fois encore nous pouvons appeler des procédures qui peuvent elles-mêmes appeler d'autres procédures. 16 fichiers de commandes peuvent être ouverts en même temps, aussi si un fichier est en USE, nous pouvons avoir 15 autres fichiers ouverts. Certaines commandes utilisent des fichiers supplémentaires (REPORT, INSERT, COPY, SAVE, RESTORE et PACK utilisent un fichier supplémentaire; SORT en utilise deux).

Un fichier est fermé quand la fin du fichier est atteinte ou quand la commande RETURN figure dans un fichier de commandes. La commande **RETURN** redonne le contrôle au fichier de commandes appelant.

La commande **RETURN** n'est pas toujours nécessaire mais, dans la pratique, RETURN doit être inséré à la fin de tous vos fichiers de commandes.

CONSEIL : Notez que dans notre exemple, les lignes de commande sont décalées. Ceci n'est pas obligatoire mais rend le fichier de commandes plus lisible, surtout si vous avez des structures imbriquées à l'intérieur d'autres structures. L'utilisation des lettres majuscules pour les commandes et des lettres minuscules pour les variables peut aussi améliorer (donc clarifier) la présentation de votre programme.

COMMENT ENTRER LES DONNEES INTERACTIVEMENT (WAIT, INPUT, ACCEPT)

Pour plusieurs applications, les fichiers de commandes doivent accéder à des données supplémentaires provenant de l'opérateur plutôt que d'utiliser uniquement ce qui est dans la base de données.

Votre fichier de commandes doit indiquer à l'opérateur le genre d'informations qu'il doit entrer. Un bon exemple est un menu de fonctions dans lequel on doit choisir une fonction. Les commandes sont les suivantes :

WAIT [TO variable mémoire]

Cette commande arrête le traitement du fichier de commandes et attend l'entrée d'un caractère au clavier avec un message ***** VEUILLEZ ENTRER VOTRE CHOIX.** Le traitement continue lorsqu'on appuie sur une touche. Si une variable est également

spécifiée, le caractère entré y est stocké. Si l'entrée est un caractère non imprimable, (par exemple <enter>), un espace est entré dans la variable.

INPUT ['message'] TO variable mémoire

Cette commande accepte toutes sortes de données en provenance du terminal et l'introduit dans la variable mémoire citée créant ainsi cette variable si elle n'existait pas déjà. Si l'option message rapide a été choisie (une simple ou double apostrophe est utilisée), le message apparaît sur le terminal suivi d'un point, pour montrer où la donnée doit être frappée. Le type de donnée de la variable (caractère, numérique ou logique) est déterminé par le type de données entrées.

Les chaînes de caractères doivent être entrées, encadrées d'apostrophes.

ACCEPT ['message'] TO variable mémoire

Cette commande accepte toutes les données caractères sans limitation. Elle est très utile pour l'entrée de longues chaînes.

Conseils pour l'utilisation de ces commandes

WAIT peut être utilisée pour les entrées rapides et réagit instantanément à l'entrée, mais ne doit pas être utilisée quand une entrée erronée peut nuire à votre base de données.

ACCEPT est utile pour les longues chaînes de caractères car elle ne demande pas aux chaînes d'être délimitées par des apostrophes.

INPUT accepte les données numériques, logiques ou caractères; peut être utilisée comme ACCEPT.

COMMENT PLACER DES DONNEES ET DES MESSAGES (@..SAY.GET)

Les commandes ?, ACCEPT et INPUT peuvent être utilisées pour placer les messages à l'écran pour l'opérateur.

Si votre terminal a un support X-Y pour le positionnement du curseur, une autre commande de dBASE II vous permet de positionner votre message et d'obtenir vos données à n'importe quel endroit choisi par vous à l'écran :

@ <coordonnées> [SAY <'message'>]

Les coordonnées sont les rangs et colonnes de la console avec 0,0 comme position supérieure en haut à gauche. Si vos coordonnées sont spécifiées comme étant « 9,34 », notre message commencera au 10ème rang de la 35ème colonne.

Note : Si vous installez la demi-intensité ou la vidéo inversée, le message sera en demi-intensité ou en vidéo inversée. Si vous voulez changer ce fait, refaites la procédure d'installation et utilisez l'option « Modify/Change ».

La commande **SAY..** est optionnelle car elle peut aussi effacer n'importe quelle ligne ou une partie d'une ligne d'écran. Appelez dBASE II et frappez :

```
ERASE
@ 20,30 SAY 'Quoi?'
@  5,67 SAY 'Ici...'
@ 11,11 SAY "C'est tout."
@ 20, 0
@  5, 0
@ 11,16
```

La commande ne se limite pas à l'affichage d'un message, elle peut aussi servir à visualiser la valeur d'une expression à une ou plusieurs variables. La forme est la suivante :

@ <coordonnées> [SAY <expression>]

```
USE Noms
@ 13,9 SAY Cod:Postal
@ 13,6 SAY Ville
SKIP 3
@ 23, 5 SAY Nom + ', ' + Adresse
```

La commande peut être étendue afin d'afficher les valeurs des variables utilisées (les variables mémoire ou les champs) à la position écran désirée. (Les variables utilisées par **GET** et **SAY** doivent exister avant d'être appelées, sinon vous obtiendrez une erreur).

@ <coordonnées> [SAY <expression>] [GET <variable>]

```
ERASE
USE Noms
@ 15, 5 SAY 'Departement' GET Department
@ 10,17 GET Cod:postal
@  5, 0 SAY 'Nom' GET Nom
```

Cette séquence positionne les valeurs des variables à différents endroits de l'écran. Afin de remplir de données les variables figurant sur votre écran, à l'endroit de votre choix, frappez :

READ

Le curseur se positionne de lui-même au premier champ saisi. Vous pouvez maintenant entrer de nouvelles données ou laisser le champ tel qu'il est en appuyant sur la touche <enter>. Lorsque vous quittez ce champ, le curseur se positionne sur la deuxième variable que vous avez entrée.

Changez les données dans les deux champs qui restent. Lorsque vous en avez terminé avec le dernier champ, vous revenez sous dBASE II. Maintenant, frappez **DISPLAY**. L'enregistrement contient maintenant de nouvelles données.

Comme vous avez pu vous en apercevoir, **GET** fonctionne un peu comme les commandes **INPUT** et **ACCEPT**. **GET** vous permet également d'entrer plusieurs variables.

Une base de données peut avoir une douzaine ou deux champs (au maximum, 32) pour n'importe quelle procédure d'entrée de données. Plutôt que d'utiliser la commande **APPEND**, qui vous listera tous les champs de vos bases de données à l'écran, vous pouvez utiliser **APPEND BLANK** pour créer un enregistrement avec des champs vides puis **GET** les données que vous voulez.

Pour vous habituer aux fichiers de commandes, veuillez créer un fichier appelé <Essai.CMD> avec les commandes suivantes :

```
ERASE
? 'Cette procedure vous permet d'ajouter de nouveaux enregistrements'
? 'au fichier Noms.DBF. Nous allons maintenant ajouter le nom et le '
? 'code postal '
?
? 'Tapez S pour arrêter la procedure,'
? '<enter> pour continuer.'
WAIT TO Continue
USE Noms
DO WHILE Continue <> 'S' .AND. Continue <> 's'
   APPEND BLANK
   ERASE
   @ 10, 0 SAY "NOM" GET Nom
   @ 10,30 SAY "CODE POSTAL" GET cod:postal
   READ
   ?
   ?  '  S pour arrêter la procedure,'
   ? '<enter> pour continuer.'
   WAIT TO Continue
ENDDO
RETURN
```

Si vous êtes revenu sous CP/M, frappez **dBASE Essai** (ou **DO Essai** si vous avez renommé le fichier dBASE.COM ainsi nous vous l'avons suggéré).

Maintenant entrez quelques données dans plusieurs enregistrements. Lorsque vous aurez terminé, **LIST** le fichier pour visualiser ce que vous avez ajouté.

Comme vous avez pu le voir, l'entrée des données est simple et non encombrante.

Pour personnaliser votre écran, vous pouvez positionner vos champs de variables et de messages à l'endroit désiré.

NOTE : Vous devez utiliser la commande **ERASE** ou **CLEAR GETS** après avoir exécuté 64 **GET's**. Utilisez CLEAR GETS si vous ne voulez pas changer d'écran.

UN FICHIER DE COMMANDES QUI RESUME CE QUE NOUS AVONS APPRIS

Avant de commencer à lire, vous pouvez exécuter le programme pour voir ce qu'il fait. Frappez **dBASE Exemple** si vous êtes sous CP/M ou **DO Example** si vous êtes sous dBASE II. Après avoir exécuté ce programme, vous pouvez revenir au manuel. Ce programme reprend la plupart des commandes que nous avons vues et comprend d'assez larges commentaires.

```
************************  EXEMPLE.CMD ******************************
* Ce fichier de commandes amène un écran de message à l'utilisateur
* et accepte des données dans une variable mémoire, puis exécute les
* procédures choisies par l'utilisateur. Bien que ce ne soit qu'un
* fragment de programme, ce fichier de commandes va s'exécuter.
*     Nous n'avons pas encore écrit de procédures qui soient appelées
* par le menu, mais nous pouvons voir l'ordinateur exécuter quelques
* actions qui nous montrerons le fonctionnement de celles-ci.
*     Normalement, dBASE II nous montre le résultat des commandes à
* l'écran. Ce fait pourrait être gênant, aussi nous allons faire
* SET TALK OFF.

SET TALK OFF
USE Dépense
ERASE

* C'est une bonne habitude de nettoyer son écran avant d'y visualiser
* toute nouvelle donnée.
```

```
* La fonction de substitution de display peut être utilisée pour entrer
* les informations dans l'écran de la façon suivante :

?
?
? '          MENU DE DEPENSE '
?
?
? '          0 = fin'
? '          1 = Résumé des comptes débits'
? '          2 = Entrée de nouvelles factures'
? '          3 = Entrée des paiements effectués'
?
? '          Choississez un numéro'
WAIT TO Choix
ERASE

* Puisque nous n'avons pas encore développé de procédures pour ces trois
* articles, l'ordinateur nous visualisera différents commentaires qui
* dépendent de l'alternative choisie dans le menu.

    IF Choix = '1'
   @ 0,20 SAY 'Un'
ELSE
   IF Choix = '2'
      @ 1,20 SAY 'Deux'
   ELSE
      IF Choix = '3'
         @ 2,20 SAY 'Trois'
      ELSE
         @  7,20
         @  8,20 SAY ' TOUT CARACTERE AUTRE QUE 1, 2, OU 3 ARRETERA '
         @  9,20 SAY ' CE PROGRAMME APRES IMPRESSION DE CE MESSAGE '
         @ 10,20 SAY ' REMARQUEZ QUE LES NOMBRES SONT ENTOURES PAR '
         @ 11,20 SAY ' DES APOSTROPHES DANS LES PHRASES "IF" '
         @ 12,20 SAY ' LA CAUSE EST QUE LA COMMANDE WAIT ACCEPTE'
         @ 13,20 SAY ' UNIQUEMENT LES ENTREES EN CARACTERES'
         @ 14,20 SAY '
      ENDIF 3
   ENDIF 2
ENDIF 1
```

```
* A chaque IF doit correspondre un ENDIF. Nous avons également
* mis une étiquette après chaque ENDIF pour indiquer à quel IF
* il correspond, ainsi sommes-nous certain d'avoir terminé toutes
* nos boucles.
?
?
?
?
INPUT 'Voulez-vous continuer (Oui ou Non)?' TO Decision
ERASE
IF Decision                    :
   INPUT "D'accord, donnons rapidement un numéro"  TO Numéro
ELSE
   @ 10,20 SAY "  POURQUOI PAS ?  "
   WAIT
ENDIF
ERASE
@ 10,20 SAY " DESOLE, JE NE SUIS PAS PRET . AU REVOIR"

* La boucle DO WHILE garde pendant quelques secondes le dernier
* message à l'écran pour vous permettre de le lire avant que le
* que le programme ne se termine.
* Afin d'en modifier la durée, nous pouvons changer soit la limite
* (100), soit le pas (+ 1).

 STORE 1 TO X
 DO WHILE X < 100
   STORE X + 1 TO X
 ENDDO
 ERASE
 RETURN
```

Si vous voulez exécuter ce programme à nouveau. Essayez toutes les alternatives puis essayez d'entrer des données définitivement erronées. Vous verrez comment fonctionne le programme et comment dBASE II détecte les erreurs.

Bien que ce ne soit qu'un fragment de programme, <exemple.CMD> nous montre malgré tout pas mal de choses :

1. L'utilisation fréquente de **ERASE** est une très bonne technique pour « nettoyer son écran ».

2. L'utilisation des alinéas offre un programme plus clair à l'utilisateur. C'est pour la même raison que nous utilisons les lettres majuscules et minuscules.
3. Le point d'interrogation « ? » peut être utilisé pour insérer des lignes d'espaces visualisables à l'écran et pour montrer les chaînes de caractères entre apostrophes ou crochets.
4. La commande **WAIT** attend l'entrée d'un simple caractère avant de laisser le programme continuer à s'exécuter. L'entrée doit être traitée comme un caractère.
5. La commande **INPUT** attend et accepte toutes sortes de données mais les caractères et les chaînes de caractères doivent être délimités par des guillemets, des apostrophes ou des crochets. Pour ne pas gêner l'ordinateur, si vous avez une apostrophe dans votre message, vous devez utiliser les guillemets ou les crochets pour définir vos chaînes de caractères.
6. Vous n'avez pas à définir vos variables à l'avance. Nommez-en une au moment où vous en avez besoin. Vous pouvez avoir un maximum de 64 variables actives au même moment.
7. Les valeurs logiques peuvent être traitées en style télégraphique. Ainsi dans le programme, IF Decision est équivalent à « IF Decision = T ».
8. Le **RETURN** à la fin du programme n'est pas nécessaire. Mais il y figure pour faire face à l'éventualité qui ferait de ce programme une sous-procédure appelée par un autre fichier de commandes.

COMMENT TRAVAILLER AVEC DES BASES DE DONNEES MULTIPLES (PRIMARY, SECONDARY,SELECT)

Comme nous avons pu le voir lorsque vous avez commencé à travailler avec dBASE II, vous frappez **USE <nom-de-fichier>** pour indiquer à dBASE le nom du fichier qui vous intéresse, et commencer le traitement d'entrée des données, d'édition, etc...

Pour travailler sur une base de données différente, frappez **USE <Nouveau-Fichier>**. dBASE II ferme alors le premier fichier et ouvre le second sans que vous en soyez concerné. De cette façon, vous pouvez travailler avec un bon nombre de fichiers, que ce soit de votre terminal ou dans vos fichiers de commandes. Vous pouvez aussi fermer un fichier sans en ouvrir un autre en frappant **USE**.

Lorsque vous faites **USE** un fichier, dBASE II « se réembobine », revient au début et se positionne sur le premier enregistrement du fichier. Dans la plupart des cas, c'est exactement ce que vous voulez. Cependant, dans certains cas, vous aimeriez accéder à d'autres fichiers sans perdre votre place dans le premier.

dBASE II est un langage évolué, il vous permet de travailler sur deux fichiers ouverts simultanément : **PRIMARY** et **SECONDARY**. Vous passerez de l'un à l'autre des fichiers à l'aide de la commande **SELECT**.

Lorsque vous commencez, vous êtes placé automatiquement sur le **PRIMARY**. Pour travailler avec une autre base de données sans perdre votre position dans la première, frappez **SELECT SECONDARY** puis **USE** <**nouveau fichier**>. Pour revenir à la position de travail de départ, frappez **SELECT PRIMARY**, puis continuez avec cette base de données.

Les deux fichiers peuvent être utilisés indépendamment. Les commandes qui déplacent des données ou opèrent sur des enregistrements fonctionnent uniquement sur le fichier en **USE**.

Cependant, les informations peuvent être transférées d'un fichier vers un autre en utilisant le **P.** et **S.** comme préfixes pour les variables. Si vous travaillez dans le secteur PRIMARY, utilisez le S comme préfixe de la variable dont vous avez besoin dans le secteur SECONDARY. Si vous êtes dans le secteur SECONDARY, utilisez le P. comme préfixe pour les variables dont vous avez besoin dans votre secteur PRIMARY.

QUELQUES GENERALITES PRATIQUES SUR LES COMMANDES ET FONCTIONS

MODIFY COMMAND <**nom-de-fichier**> vous laisse modifier directement le fichier de commandes de dBASE II en utilisant la forme normale de l'édition plein écran.

BROWSE vous visualise au maximum 19 enregistrements avec autant de champs que la largeur de l'écran peut en contenir. Utilisez **ctrl-B** pour le défilement des données vers la droite et **ctrl-Z** vers la gauche.

CLEAR remet toutes les variables à blanc et ferme tous les fichiers.

RESET est utilisée après un changement de disquette pour recommencer à zéro avec le système d'exploitation. Lisez la description détaillée dans le dictionnaire des commandes (Partie II) avant d'utiliser cette commande.

* vous permet d'insérer des commentaires dans un fichier de commandes, mais ces commentaires ne seront pas visualisés lors de l'exécution du fichier de commandes. Ainsi, le programmeur peut ajouter des explications concernant son programme sans troubler l'opérateur. Il *doit* y avoir au moins un espace entre le symbole et le commentaire. Les commentaires ne doivent pas figurer sur une ligne de commandes. NOUS REPETONS : Les commandes et les commentaires doivent être sur des lignes séparées.

REMARK permet de stocker des commentaires dans les programmes, mais contrairement à la commande *, visualise ces commentaires à l'exécution du programme. Il doit y avoir au moins un espace entre le mot et la remarque, cette dernière ne peut figurer sur une ligne de commande.

RENAME <ancien-fichier> TO <nouveau-fichier> permet de changer le nom des fichiers du répertoire CP/M. *Ne pas essayer de renommer un fichier en USE.*

QUIT TO <système.COM liste de fichier> vous permet de terminer avec dBASE II et de revenir automatiquement à l'exécution d'un autre fichier de commandes .COM sous CP/M. Chaque fichier .COM cité doit être encadré par des apostrophes et séparé des autres fichiers cités par des virgules.

Vous pouvez également utiliser la commande ? pour appeler les fonctions suivantes :

est la **fonction numéro d'enregistrement**. Elle vous donne le numéro de l'enregistrement en cours.

* est la **fonction enregistrement à effacer**. La valeur est vraie si l'enregistrement est marqué pour effacement, sinon elle est fausse.

EOF est la **fonction fin de fichier**. La valeur est vraie si la fin du fichier en USE a été atteinte, et fausse si elle ne l'a pas été.

COMMENT ECRIRE ET STRUCTURER VOS FICHIERS DE COMMANDES

Commencez par éteindre votre ordinateur avant d'écrire un fichier de commandes.

Ce conseil est valable car nombreux sont les programmeurs qui font cette erreur : ils commencent immédiatement à coder une solution avant même d'avoir une idée claire de ce qu'ils désirent faire.

Enfin, voici brièvement quelques conseils :

— Commencez par définir votre problème en phrases normales.

— Entrez maintenant dans les détails. Quelles sortes d'entrées allez-vous avoir ? Quelle sera la forme de vos sorties et de vos états ?

— Ensuite, essayez de cerner les exceptions. Quelles seront les conditions de début ? Qu'arrivera-t-il s'il manque un enregistrement ?

Lorsque vous aurez terminé la définition de ce que vous voulez faire, décrivez les détails en anglais modifié. Le texte est appelé « pseudocode ». Dans les grandes lignes, votre programme peut ressembler aux lignes suivantes :

```
Utiliser la base de données Cout
Imprimer les factures impayées du mois dernier
Faire un chèque pour chaque facture impayée.
```

En ajoutant un peu plus de détails, il ressemblera à ceci :

```
USE Cout
Impression des factures impayées du mois dernier en utilisant le fichier
    RESUME.FRM.
Commencer le traitement du début de la base de données jusqu'à la fin.
Si la facture n'a pas été réglée
    Payer cette facture
    Saisie de cette facture dans la base de données
Faire cette opération pour chaque enregistrement.
```

Avec un petit effort, les lignes précédentes peuvent être traduites dans un fichier de commandes de la façon suivante :

```
USE Cout
* Impression des impayés du mois de Decembre, 1985.
REPORT FORM Resumé FOR date:note >= '851201' .AND.
   date:note <= '851231' TO PRINT
GOTO TOP                    * Aller au 1er enregistrement
DO WHILE .NOT. EOF          * Répèter l'opération jusqu'à la fin

   IF Check:num=' '         * La facture n'a pas été payée
      DO Ecriture           * faire un chèque puis
      DO MAJ                * mise à jour des enregistrements
   ENDIF
SKIP                        * Aller à l'enregistrement suivant
ENDDO
```

Cet exemple n'est qu'une approche pour résoudre la plupart des problèmes. La première étape consiste à avoir une vue d'ensemble, et définir ce qui est et ce qui n'est pas. Nous entrons ensuite progressivement dans les détails en résolvant d'abord ceux qui sont les plus faciles, laissant les détails les plus compliqués de côté pour les résoudre ultérieurement.

Vous pouvez toujours tester une partie du programme en utilisant ce que les programmeurs appellent un « **module** ». Exécutez un fichier de commandes que vous avez cité dans un programme et entrez trois articles. Un message vous fera savoir que le programme a atteint WAIT et RETURN.

dBASE II exécute ces fichiers de procédures, vous donne le message, puis revient et continue avec le reste du programme après que vous ayiez appuyé sur n'importe quelle touche.

CHAPITRE IV

LES FONCTIONS MULTI-FICHIERS

COMMENT ETENDRE VOTRE POUVOIR DE CONTROLE AVEC LES FONCTIONS

Les fonctions sont des opérations spéciales utilisées avec des expressions pour exécuter des traitements difficiles ou impossibles à réaliser avec des opérations arithmétiques, logiques ou chaînes.

Ces fonctions sont appelées en frappant un point d'interrogation « ? », un espace et la fonction. Elles peuvent être appelées du terminal ou à l'intérieur des fichiers de commandes.

NOTE : Les parenthèses qui vous sont montrées ci-dessous doivent être utilisées. Rappelez-vous qu'une chaîne est simplement une suite de caractères incluant des espaces, des nombres et des symboles. Elles sont manipulées comme des données. Une « sous-chaîne » fait partie d'une chaîne spécifique.

!(< variable/chaîne >)
Le point d'exclamation vous permet de changer tous les caractères en minuscules de 'a' à 'z' dans une chaîne ou une variable de chaîne en majuscules. Tout autre caractère dans la chaîne ne sera pas affecté. Ceci simplifiera par la suite notre recherche des données puisque nous savons que toutes les données sont stockées en majuscules, quelle que soit la façon dont elles ont été entrées.

TYPE(< expression >)
TYPE est la fonction type de données. Les champs peuvent être de type C, N, ou L suivant les types d'expressions de données, caractères, numériques ou logiques.

INT(< variable/expression >)
C'est la fonction de nombres entiers. Elle arrondit un nombre fractionnaire en supprimant la partie décimale. Le terme à l'intérieur des parenthèses (vous *devez* utiliser les parenthèses) peut être un nombre, un nom de variable ou une expression complexe. EXEMPLES : INT(123.86) vous donne un champ de 123, tandis que INT(-123.86) vous donne un champ de -123. INT(Montant) vous donnera 2,333 et non $2,333.75.

Pour arrondir au plus près un nombre entier (plutôt que de le tronquer), utilisez la forme suivante : INT(valeur + 0.5). La valeur comprise entre les parenthèses est d'abord déterminée puis la fonction INT va s'exécuter.
La fonction INT peut également être utilisée pour arrondir la valeur de n'importe quel nombre avec des chiffres décimaux. Exemple : IN(valeur*10 + 0.5)/10 arrondit la décimale la plus proche à cause de l'ordre de priorité des opérations (parenthèses, entier, puis division). Pour arrondir à deux décimales, utilisez « 100 », pour trois décimales utilisez « 1000 », etc...

VAL(< variable/chaîne/sous-chaîne >)
Cette fonction convertit une chaîne de caractères ou une sous-chaîne de caractères en un nombre. Ce nombre sera signé et aura un point décimal.
Exemple : VAL('123') vous donnera le nombre 123.
Vous pouvez également utiliser cette fonction avec l'opérateur de sous-chaîne VAL($(< chaîne >)).

STR(< expression/variable/nombre > , < longueur > , < décimales >)
Cette fonction convertit les nombres ou le contenu d'une variable numérique en une chaîne de caractères en spécifiant la longueur ainsi que le nombre de décimales. La longueur spécifiée *doit* être assez grande pour contenir au moins tous les nombres ainsi que le point décimal. Si la valeur numérique est plus courte que le champ spécifié, la partie du champ restant est remplie de blancs. Si la précision décimale n'est pas spécifiée, elle est supposée nulle.
Cette fonction est utilisée assez souvent dans les systèmes de comptabilité pour simplifier la visualisation. Les nombres sont convertis en chaînes puis joints à d'autres chaînes de caractères pour la visualisation.

LEN(< variable/chaîne >)
Cette fonction vous donne le nombre de caractères que vous avez dans une chaîne citée. Cette opération peut être utile quand le programme doit décider de la longueur d'une variable mémoire à allouer à une information et ceci sans aucune intervention de l'opérateur. Cependant, si un champ de variable de caractère est utilisé, cette fonction garde la dimension de ce champ et non la longueur du contenu.

$(< expression/variable/chaîne > , < début > , < longueur >)
Ceci est une fonction de sous-chaîne. Cette fonction sélectionne les caractères d'une

chaîne ou d'une variable de caractères commençant à la position spécifiée et continuant pour une longueur spécifique.
Exemple : Si nous avions une variable appelée <Date> dont la valeur est '10823', la fonction **$(Date,5,2)** donnerait 23. Pour convertir ces numéros en nombre, nous pourrions utiliser **VAL($(Date,5,2))**.
A ne pas confondre avec les opérateurs de sous-chaîne logique décrits en Chapitre II.

@(<variable1/chaîne1>, <variable2,chaîne2>)
C'est une fonction de recherche de sous-chaîne : « Où se trouve la chaîne 1 dans la chaîne 2 ? ». Cette fonction nous dira où commence la première chaîne ou variable de chaîne dans la seconde chaîne ou variable de chaîne. Si la première chaîne n'existe pas, une valeur de « 0 » est retournée.
Nous utilisons cette fonction pour trouver où commence une chaîne spécifique afin de pouvoir utiliser la fonction de sous-chaîne. Nous pouvons aussi utiliser cette fonction pour trouver l'existence d'une chaîne.
(Si vous voulez seulement savoir si une chaîne se trouve à l'intérieur d'une autre chaîne, vous pouvez utiliser l'opérateur de chaîne relationnel : *Chaîne1$Chaîne2*, Chapitre II).

CHR(<nombre>)
Cette fonction vous donne l'équivalence du nombre en caractères ASCII. Suivant la manière dont votre terminal utilise le code ASCII, ? CHR(12) peut effacer votre écran. ? CHR(14) peut vous donner la vidéo inversée et ? CHR(15) annule la vidéo inversée. D'autres fonctions peuvent être utilisées pour contrôler les périphériques utilisés par votre système, par exemple une imprimante.
Afin d'obtenir une ligne soulignée sur votre imprimante, essayez de joindre une chaîne de caractères, un retour de chariot et un soulignement avec la commande suivante : **? 'chaîne' + CHR(13) + ' '**. Vous pouvez même écrire un fichier de commandes utilisant la fonction LEN pour trouver la longueur de la chaîne et la souligner.

&
C'est une fonction de macro-substitution. Lorsque ce symbole précède un nom de variable mémoire, dBASE II remplace le nom par le contenu de la variable mémoire. Les données doivent être des caractères. Cette fonction peut être utilisée lorsqu'une expression complexe doit être utilisée fréquemment pour passer d'un paramètre à un autre entre les fichiers de commandes (ou dans un fichier de commandes où la valeur du paramètre sera fournie à l'exécution du programme).
Cette fonction peut être utilisée pour obtenir le nom de la base de données désirée.
Exemple :

```
? 'Quel fichier désirez-vous revoir ?'
ACCEPT TO Base-de-données
USE &Base-de-données
```

Elle peut également être utilisée comme abréviation d'une commande. Exemple : **STORE 'Delete Record' TO D**. La commande **&D 5** effacera l'enregistrement 5 lors de l'exécution du programme.
Si une macro-commande n'est pas suivie d'une chaîne de variables valide, elle ne sera pas exécutée. Ce qui signifie que vous n'aurez pas de message d'erreur si vous vous trompez.

FILE(< « nom-de-fichier »/variable/expression >)
Produit une valeur vraie si le fichier existe sur le disque, et fausse si le fichier n'existe pas. Si vous utilisez un nom de fichier spécifique, encadrez-le d'apostrophes. Le nom d'une chaîne de variables ne nécessite pas d'apostrophes. Vous pouvez également utiliser la chaîne d'expression suivante : **FILE(« B : » + Base-de-données)**. Cette expression vous dira si le fichier stocké dans la variable mémoire < Base-de-données > se trouve sur l'unité de disque B.

TRIM
Elimine les blancs inutiles à droite d'une chaîne de variables. Elle est utilisée de la façon suivante :

```
STORE TRIM (<variable>) TO <nouvelle-variable>
```

RANK (< chaine >)
Retourne la valeur décimale du premier caractère d'une chaîne. Cette fonction correspond à la fonction ASC de BASIC.

COMMENT CHANGER LES CARACTERISTIQUES DE dBASE II

dBASE II a un certain nombre de commandes contrôlant le fonctionnement de l'interactivité de votre système. Vous pouvez changer ces paramètres à n'importe quel moment, ou les installer une fois pour toute au début de vos fichiers de commandes et les laisser tels quels. Dans certaines applications, vous pourriez n'avoir besoin que des valeurs par défaut.

Les paramètres sont changés dans vos fichiers de commandes ou par l'utilisation de la commande SET.

Dans la liste ci-dessous, les valeurs par défaut sont en gras.

SET TALK **ON**	visualise les résultats des commandes à la console.
OFF	pas de visualisation.
SET PRINT ON	liste les sorties sur l'appareil que vous désignez.
OFF	pas de listage.
SET CONSOLE **ON**	liste les sorties sur votre console.
OFF	pas de listage.
SET SCREEN **ON**	met en marche l'opération full screen pour les commandes APPEND, EDIT, INSERT et CREATE.
OFF	désactive l'opération full screen.
SET FORMAT TO SCREEN	envoie les sorties des commandes '@' à l'écran.
SET FORMAT TO PRINT	envoie les sorties des commandes '@' à l'imprimante.
SET FORMAT TO <fichier .FMT>	utilise les formats créés précédemment par les commandes APPEND, EDIT, INSERT et CREATE.
SET MARGIN TO <nnn>	pose la marge gauche de votre imprimante. (« nnn » < 254)
SET RAW ON	affiche et liste vos enregistrements sans espaces entre les champs.
OFF	affiche et liste vos enregistrements avec un espace entre les champs.
SET HEADING TO <chaîne>	change les titres de la commande REPORT.
SET ECHO ON	toutes les commandes de vos fichiers de commandes sont visualisées sur votre console au fur et à mesure qu'elles sont exécutées.
OFF	pas de visualisation.
SET EJECT ON	permet l'alimentation des feuilles avec la commande REPORT.
OFF	pas d'alimentation de feuilles.
SET STEP ON	arrête le programme après l'exécution complète de chaque commande, afin de trouver les erreurs.
OFF	continuation normale de l'opération.

SET DEBUG ON	envoie les sorties des commandes ECHO et STEP uniquement à l'imprimante.
OFF	envoie les sorties des commandes ECHO et STEP à l'écran.
SET BELL **ON**	met en route un signal sonore quand le champ est plein
OFF	pas de signal sonore.
SET COLON **ON**	les points sont utilisés pour délimiter l'entrée des variables à l'écran.
OFF	pas de délimitation.
SET CONFIRM ON	attend après <enter> avant de quitter la variable pendant le full screen editing.
OFF	quitte la variable quand le champ est plein.
SET CARRY ON	transfère les données de l'enregistrement précédent dans le nouvel enregistrement lorsque vous êtes en mode APPEND.
OFF	vous présente un enregistrement blanc lorsque vous êtes en mode APPEND.
SET INTENSITY **ON**	permet la double intensité pour les opérations full screen.
OFF	désactive la double intensité.
SET LINKAGE ON	permet la visualisation des bases de données devant être liées : maximum 64 champs et 2 000 octets par enregistrement. Les préfixes P. et S. doivent être utilisés lorsque les champs de noms sont semblables dans les deux bases de données.
OFF	désactive la liaison.
SET EXACT ON	exige pour une comparaison que tous les caractères compris dans les deux chaînes soient exactement les mêmes.
OFF	permet une longueur de chaîne différente. Exemple : 'ABCD' = 'AB' sera vrai. (Cette fonction affecte aussi la commande FIND).
SET ESCAPE **ON**	permet à la clé <escape> d'abandonner l'exécution du fichier de commandes.
OFF	désactive la clé <escape>.

SET ALTERNATE TO <nom-de-fichier> crée un fichier avec une extension .TXT pour sauvegarder tout ce qui a été fait à l' écran. Pour commencer la sauvegarde, frappez SET ALTERNATE ON.

Vous pouvez changer le fichier que vous avez sauvegardé en frappant **SET ALTERNATE TO <nouveau-fichier>**.

Pour arrêter, frappez SET ALTERNATE OFF.

COMMENT FUSIONNER DEUX BASES DE DONNEES (UPDATE)

Les données peuvent être transférées d'un fichier de données à un autre avec la commande suivante :

```
UPDATE FROM <fichier-de-données> ON <clé> [ADD <list champ>]
                                          [REPLACE <list champ>] [RANDOM]
                                          [REPLACE <champ> WITH <champ>]
```

Note : Les deux fichiers de données doivent être triés ou indexés sur une 'clé' si l'option RANDOM n'est pas utilisée.

Sans l'option RANDOM, les deux fichiers reviennent à leur point de départ et leurs champs de clé sont comparés. Si les clés sont identiques, les données de la base de données FROM seront soit *ajoutées* numériquement aux données du fichier en USE, soit *utilisées* pour remplacer les données dans le fichier en cours par les champs spécifiés dans la liste des champs. Si les champs ne sont pas semblables, ces enregistrements ne seront pas traités. Cette commande peut être utilisée par exemple pour la mise à jour des stocks.

Si les champs des deux bases de données ont le même nom, vous pouvez lister les champs dans lesquels les données doivent être remplacées.
Si les champs ont des noms différents, vous pouvez utiliser la commande REPLACE.

Si la clause RANDOM est utilisée, le fichier de données devant être mis à jour devra obligatoirement être indexé sur le champ de <clé>. L'organisation du <fichier de données> d'origine peut être quelconque. A chaque lecture d'un enregistrement dans le <fichier de données> d'origine, une commande FIND est émise pour localiser et mettre à jour l'enregistrement du fichier en cours d'utilisation.

COMMENT ASSEMBLER DES BASES DE DONNEES ENTIERES (JOIN)

JOIN est l'une des plus puissantes commandes de dBASE II. Elle peut combiner deux bases de données (les fichiers en USE sur les secteurs PRIMARY et SECONDARY) pour créer un troisième fichier. La forme de la commande est la suivante :

JOIN TO <nouveau-fichier> ON <expression> [FIELD <list>]

La commande positionne dBASE II sur le premier enregistrement du fichier en USE du secteur PRIMARY et évalue chacun des enregistrements dans le fichier en USE du secteur SECONDARY. Toutes les fois que les champs d'expression sont vrais, un enregistrement est ajouté à la nouvelle base de données. Si vous êtes sur le secteur PRIMARY lorsque vous frappez la commande JOIN, le préfixe S. doit précéder le nom de la variable du fichier en USE dans le secteur SECONDARY. Si vous êtes dans le secteur SECONDARY, faites précéder le nom de votre variable du fichier en USE du secteur PRIMARY du préfixe P.
Cette opération est répétée jusqu'à la fin de la comparaison de tous les enregistrements.

Note : Cette opération peut être très longue si vos deux bases de données sont grandes. Il se peut aussi qu'elle ne soit pas du tout complète si les contraintes sont trop grandes. Deux fichiers, avec 1 000 d'enregistrements chacun, créeront une base de données de 1 000 000 d'enregistrements si les expressions de JOIN sont toujours vraies, alors que dBASE II est limité à 65 535 enregistrements dans une simple base de données.

Pour utiliser cette commande, frappez la séquence d'instructions suivante :

```
USE Stock
SELECT SECONDARY
USE Commmande
JOIN TO Nouvfich FOR P.Part:Num=Part:Num;
        FIELD Client,Article,Montant,Coût
```

Ceci va créer une nouvelle base de données appelée <NouvFich.DBF> avec quatre champs : Client, Article, Montant et Coût. La structure de ces champs (type de données, dimensions) est la même que celle des deux bases de données rassemblées. (A noter que le préfixe P. est utilisé pour appeler une variable d'une zone de travail non en USE).

COMMENT EDITER ET FORMATER EN PLEIN ECRAN (TEXT, ENDTEXT,@..SAY..GET..PICTURE, .FMT)

TEXT
Tout texte que vous entrez sera envoyé directement à l'écran ou à l'imprimante sans aucun traitement de commandes. Vous devez cependant utiliser des ';' pour la continuation de vos lignes et un **ENDTEXT** doit figurer à la fin du texte.

dBASE II dispose d'une série de commandes puissantes pour le formatage : vous pouvez positionner l'information exactement là où vous le voulez. Vous avez vu cette action dans notre programme <Exemple.CMD>.

@ <coordonnées> SAY ['message'] GET <variable>

Cette commande est capable de positionner les messages et les variables ainsi que leurs valeurs à n'importe quel endroit spécifié à l'écran. Lorsque nous listons une série de commandes suivie d'un **READ**, nous sommes capables de contrôler le format de l'écran entier. Vous pourriez vouloir créer et exécuter le fichier de commandes suivant :

```
STORE "    " TO MDate
STORE "     " TO MBalance
STORE "     " TO MLigne
@ 5,5 SAY "Set date JJ/MM/AA " GET MDate
@ 10,5 SAY "Quelle est la balance? " GET MBalance
READ
ERASE
@ 5,5 SAY "Devons-nous faire une evaluation?" GET MValeur
READ
```

La commande peut également être utilisée sans la phrase SAY, comme suit : @ <coordonnées> GET <variable> (le READ figurera plus tard dans le fichier de commandes). Ceci vous visualisera uniquement les points délimitant la longueur des champs des variables.

ASTUCE : En mode SCREEN, les numéros de lignes ne sont pas tenus d'être en ordre, mais dans la pratique vous devez les ordonner puisqu'ils sont utilisés par l'ordre PRINT du formatage.

Cette commande peut également être étendue pour un formatage spécial tel que :

@ <coordonnées> SAY [expression] GET <variable> [PICTURE <format>]

L'option de la phrase **PICTURE** est remplie pour l'utilisation du format des symboles listés ci-dessous :

```
@ 5,1 SAY "La date d'aujourd'hui est " GET Date PICTURE '99/99/99'
```

Ceci vous donnera :

```
La date d'aujourd'hui est :  /  /  :
```

Nous avions initialisé la variable Date à blanc. Dans cet exemple, seuls les nombres peuvent être entrés.

Les **symboles de la fonction GET** sont :

9 ou # accepte uniquement les nombres en entrée.
A accepte les caractères alphabétiques.
! convertit l'entrée des caractères en majuscules.
X accepte n'importe quel caractère.
**$** imprime '$' à l'écran.
***** imprime '*' à l'écran.

Avec la commande « *@* », vous pouvez formater rapidement et facilement votre menu ainsi que l'entrée des écrans.

Si vous utilisez uniquement les commandes *@*...SAY...GET et les commentaires (précédés d'un *, vous pouvez sauvegarder ce format dans un fichier avec une extension .FMT. Le fichier de format .FMT peut contenir des instructions spéciales qui permettront à l'opérateur d'entrer des données uniquement dans les champs visés.

Pour utiliser le fichier .FMT, tapez :

SET FORMAT TO <nom-du-fichier>

Ainsi, lorsque vous utilisez les commandes APPEND, EDIT ou INSERT, ce format vous sera visualisé.

Le programme **ZIP** incorporé à dBASE II vous permet de dessiner facilement et rapidement votre écran en positionnant vos messages et champs de données à l'endroit que vous désirez. En quelques secondes, **ZIP** vous crée soit un fichier .CMD (fichier de commandes) pour la saisie des données ou les sorties des informations, soit un fichier .FMT (fichier de format) que vous pouvez utiliser avec dBASE II.

COMMENT FORMATER UNE PAGE (SET FORMAT TO PRINT, @..SAY..USING)

Lorsque vous dites **SET FORMAT TO PRINT**, la commande « *@* » envoie les informations à l'imprimante et non à l'écran.
Les phrases GET et PICTURE sont ignorées, et la commande READ ne peut être utilisée.

Les données qui doivent être imprimées sur les chèques, sur les commandes d'achat, sur les factures, ou autres formats standard, peuvent être d'abord organisées à l'écran avec la commande suivante pour les imprimer ensuite exactement comme vous le voulez :

@ coordonnées SAY variable/expression/'chaîne' [USING format]

Pour imprimer, les coordonnées doivent être en ordre. Les lignes doivent être incrémentées dans l'ordre (exemple : imprimez la ligne 7 avant la ligne 9, etc.). Il en est de même pour les colonnes, la colonne 15 devant être imprimée avant la colonne 63, etc.

Cette commande peut donner la valeur courante d'une variable citée, le résultat d'une expression ou un message de chaîne littérale.

Si la phrase **USING** est incluse, cette commande spécifie quels seront les caractères à imprimer et où ils devront apparaître sur la page. Les symboles sont les suivants :

9 ou # imprime seulement les chiffres.
A imprime les caractères alphabétiques.
X imprime n'importe quel caractère imprimable.
**$** imprime un chiffre ou un '$' à la place des espaces gauches.
* imprime un chiffre ou un '*' à la place des espaces gauches.

La commande **10,50 SAY Heures*Taux USING '$$$$$$$.99'** pourrait être utilisée aussi bien à l'écran qu'à l'imprimante puisqu'il n'y a pas de phrase GET. Pour Heures = 8 et Taux = 12.73, il sera imprimé ou visualisé **$$$$101.84**.

COMMENT DESSINER ET IMPRIMER UN FORMAT

Pour dessiner un format, utilisez la configuration de votre imprimante. Notre impression a horizontalement 10 caractères au pouce et, verticalement, 6 lignes au pouce.

Dans notre fichier de commandes « Menu de dépenses », nous pourrions avoir un autre choix d'article appelé « 4 = Ecriture des chèques », aussi allons-nous exécuter une partie du fichier commande Ecriture des chèques.

Pour commencer, nous avons à entrer la date. Les lignes de commande suivantes acceptent la date d'une variable appelée MDate et vérifient, éventuellement, si celle-ci est correcte.

```
ERASE
SET TALK OFF
STORE "  " TO MDate
STORE T TO NoDate
DO WHILE NoDate
@ 5,5 SAY "Mettre date sous forme MM/JJ/AA" GET MDate PICTURE "99/99/99"
READ
   IF VAL($(MDate,1,2)) < 1;
        .OR.  VAL($(MDate,1,2)) > 12;
        .OR.  VAL($(MDate,4,2)) < 1;
        .OR.  VAL($(MDate,4,2)) > 31;
        .OR.  VAL($(MDate,7,2)) <> 81
      STORE "  " TO MDate
      @ 7,5 SAY "**** DATE INCORRECTE, VEUILLEZ LA RE-ENTRER ****"
      STORE T TO NoDate
   ELSE
      STORE F TO NoDate
   ENDIF
ENDDO nous avons maintenant la date correcte
ERASE
```

Dans cet exemple, nous avons initialisé la valeur de MDate à 8 blancs, aussi la commande @..SAY visualisera :

```
Mettre date sous forme  JJ/MM/AA : / / :
```

Lorsque la date est entrée, le IF va vérifier si le mois est compris entre 1 et 12, si le jour est compris entre 1 et 31, et si l'année est égale à 81. Cette vérification a été faite en trois étapes :

— La fonction de sous-chaîne $ prend les deux caractères représentant le mois, le jour ou l'année (exemple, le mois commence à la 4e position et occupe deux caractères)

— La fonction VAL convertit cette sous-chaîne en nombre entier

— Le nombre entier est ensuite comparé avec les valeurs allouées. Si la valeur n'est pas trouvée, MDate se remet à nouveau à blanc et un message d'erreur sera visualisé. Lorsque la date est trouvée, le programme continue son exécution.

L'impression du chèque peut être faite dans la portion du programme suivant.

```
@ 8,3 SAY dialog * Une variable de caractères va mettre le montant dans
                 * une écriture. Cette dernière sera remplie dans une
                 * autre procèdure appelée Ecriture2. Pour l'instant
                 * nous faisons :
                 * STORE 'Traitement' TO Dialog
                 * RETURN
@ 11,38 SAY Num:vendr
@ 11,50 SAY MDate
@ 11,65 SAY Montant
@ 13,10 SAY Vendeur
@ 14,10 SAY Adresse
@ 15,10 SAY Ville:Etat
@ 15,35 SAY Cod:postal
@ 17,10 SAY Qui
```

Vous pouvez vérifier la sortie de vos données à l'écran avant de les imprimer. Vous passez du mode SCREEN to PRINT avec la commande SET.

Pas de problème pour la longueur des formats : une page à imprimer peut avoir jusqu'à 255 lignes de longueur. Pour réinitialiser votre ligne de compteur, utilisez la commande **EJECT** avec l'imprimante choisie.

RESUME DU CHAPITRE IV

dBASE II est un système si puissant qu'il possède un très grand nombre de commandes et de techniques pour répondre aux besoins de vos bases de données. Il vous permet d'obtenir plus facilement de plus amples informations qu'avec tous les autres systèmes de base de données ou fichiers couramment utilisés sur d'autres micros-ordinateurs.

Le moyen le plus facile pour apprendre cette technique est de frapper nos exemples sur votre ordinateur. Vous pouvez les changer afin de les adapter à vos besoins.

Commencez par utiliser la commande CREATE pour créer vos bases de données. En plus du fichier Dépense.DBF, nous avons créé d'autres structures de bases de données que vous pourrez trouver utiles (voir le Chapitre VI).

Vous pouvez vérifier nos autres structures de bases de données pour voir comment elles sont utilisées dans les programmes. Dans toutes nos bases de données, nous essayons de garder des champs de noms et des structures individuelles semblables afin de pouvoir fusionner nos fichiers plus tard pour d'autres traitements. Les données d'un fichier pourront s'ajuster dans le champ correspondant d'un autre fichier, et comme le nom est commun, le transfert sera direct.

Vous pouvez maintenant travailler avec les fichiers de commandes du Chapitre VI. La plupart des commandes de dBASE II ont été utilisées et les fichiers peuvent être exécutés tels qu'ils sont écrits.

Le premier fichier de commandes est le menu principal pour le système avec des sous-fichiers sélectionnés par la frappe d'un nombre. Les programmes deviennent de plus en plus compliqués, aussi devez-vous aller voir les programmes utilitaires à la fin du Chapitre VI pour ne pas vous emmêler avec le reste des programmes.

Pour écrire ces fichiers de commandes, nous avons utilisé exactement les procédures qui vous ont été recommandées précédemment : d'abord définir le problème dans ses grandes lignes, entrer progressivement dans les détails, les écrire dans un langage ordinaire, puis les transformer en pseudocode.

Lorsque nous avons un problème à résoudre, mais dont nous ne connaissons pas encore la solution, nous écrivons simplement une procédure quelconque que nous reprendrons plus tard.

Les identificateurs sortent du pseudocode mais sont légèrement modifiés pour être ajustés à l'intérieur des 10 caractères alloués. Cette correction n'en modifie pas le sens. Les commentaires sont répartis dans les fichiers comme documentation.

CHAPITRE V

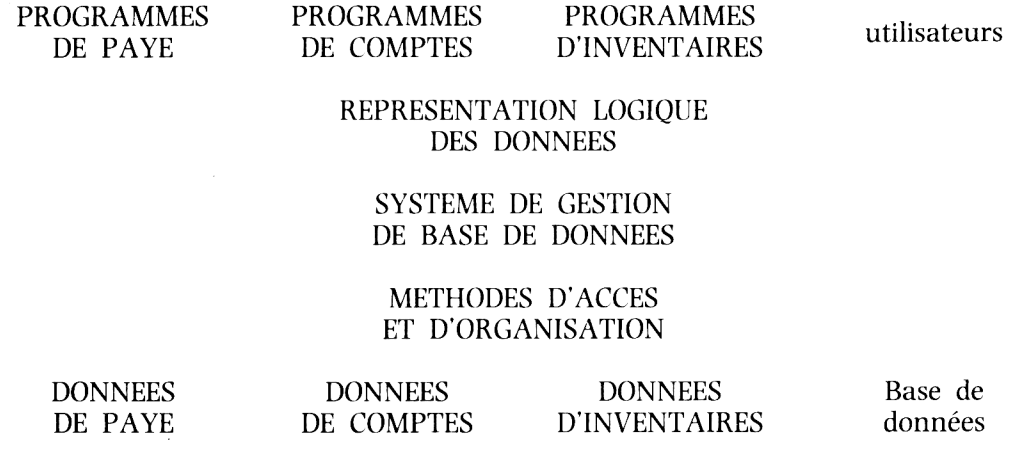
LES BASES DE DONNEES RELATIONNELLES

LE CONCEPT DE BASE DE DONNEES RELATIONNELLE

Dans un système de gestion de base de données (SGBD) tel que dBase II, le traitement (création, interrogation, modification, suppression) et la maintenance des données (réorganisation) se font de manière considérablement différente des systèmes de gestion de fichiers.

Dans les systèmes de gestion de fichiers, les programmes d'application sont conçus en fonction de l'organisation physique des données. On dit qu'ils sont « dépendants des données », c'est-à-dire que si l'on change quelque chose concernant ces données (organisation, type, ajout de nouvelles données, codification...), des modifications parfois importantes devront être apportées dans tous les programmes d'application qui les utilisent. Or, ces changements sont souvent nécessaires pour satisfaire à une évolution. De plus, une même donnée pouvant se trouver sous une forme différente dans plusieurs fichiers, il faudra, lors d'une modification, effectuer des mises à jour successives de tous les fichiers afin d'éviter une incohérence.

Ce problème de dépendance est en grande partie résolu par les systèmes de gestion de bases de données. Un SGBD réunit les données en évitant la duplication et selon une organisation connue de lui-même.



L'utilisateur ignore cette représentation physique. Il ne considère, pour concevoir ses applications, qu'une représentation « logique » qui décrit uniquement la nature des données (par exemple, les bulletins de paye, les états d'inventaires) et les liens logiques existants entre eux.

De ce fait, les programmes sont conçus dans des langages de haut niveau indépendamment des contraintes de l'organisation et de la structure des données. Ils précisent uniquement l'objet du traitement (création, interrogation, modification) et non comment accéder aux données et assurer effectivement le traitement. C'est le SGBD qui, connaissant l'organisation et la méthode d'accès en cours d'utilisation, va établir les cheminements nécessaires, exécuter les programmes correspondants et fournir les résultats demandés.

Si des modifications interviennent ultérieurement au niveau des données (par exemple, ajout de nouveaux types), ou au niveau de leur structuration, les programmes déjà écrits ne seront plus affectés. Le SGBD assurera la correspondance de ces derniers avec le nouvel ensemble de données.

La *représentation logique* des données peut revêtir une des trois formes : hiérarchique, en réseau ou relationnelle.

Les représentations hiérarchiques impliquent une programmation des applications qui dépend de la hiérarchie choisie pour structurer les données et qui comporte de nombreuses contraintes.

Les représentations en réseau ont moins de contraintes, mais la manipulation des données demeure assez complexe et encore proche du cheminement.

La représentation relationnelle, adoptée en dBASE II, est la plus simple et la plus dépendante. Les données sont représentées dans des tables à deux dimensions. Chaque colonne correspond aux valeurs d'un champ, chaque ligne à un enregistrement composé d'une suite de valeurs de chaque champ.

Col. 1	Col. 2	Col. 3	Col. 4	Col.5
N° facture	Fournisseur	Description	montant	Numéro
2386	Graphiques	Impressions	23.56	BBQ-747
78622	Sculptures	Litho	347.42	TFS-901

Chaque valeur est une valeur simple (ni rang, ni ensemble). L'ordre des enregistrements est quelconque mais chaque enregistrement est unique.

INTRODUCTION A L'ORGANISATION D'UNE BASE DE DONNEES

Lorsque vous créez votre base de données, vous voulez accéder à vos données dans un ordre bien défini.

Pour certaines bases de données, l'ordre selon lequel vous entrez vos données sera l'ordre de sortie de ces dernières. Cependant, dans la plupart des cas, vous désirerez organiser vos données différemment.

Avec dBase II, vous pouvez organiser vos données avec les commandes **SORT** ou **INDEX**. (Voir le Chapitre II).

La commande **SORT** déplace des enregistrements entiers pour les trier en ordre ascendant ou descendant, selon les champs que vous avez définis (nom, code postal etc...). Ces champs sont appelés **clés**.

La recherche des données est relativement lente puisque les bases de données triées sont lues séquentiellement.

La commande **INDEX** permet de créer vos bases de données en utilisant uniquement les clés qui vous intéressent plutôt que la base de données entière. Une **clé** est un champ de base de données (ou combinaison de champs) compris dans un enregistrement. Dans un système de stocks, le numéro peut être l'objet et le montant disponible, coût, location, etc... Pour un fichier de personnel, les noms ou numéros des employés seraient probablement les meilleures clés.

Avec une base de données indexée, seules les clés sont organisées avec un pointeur pour l'enregistrement auquel elles appartiennent. dBase II utilise une structure appelée B*-trees pour les index. Que le fichier soit grand ou de taille moyenne, la commande FIND prend deux secondes pour s'exécuter.

Si voulez organiser vos données sur plusieurs champs différents, pour différentes applications, vous pouvez créer plusieurs fichiers d'index (un pour chaque champ) et utiliser l'index approprié au moment voulu. Vous pouvez avoir plusieurs fichiers d'index triés par nom du fournisseur, par numéro de client, par code postal, ou par n'importe quelle autre clé dans une seule base de données.

Les nouvelles entrées sont automatiquement ajoutées au fichier indexé utilisé.

Un autre avantage des bases de données indexées est la rapidité de localisation des données qui vous intéressent.

LES FICHIERS DE DONNEES DE DBASE II

DBASE II a été conçu pour intégrer 65 535 enregistrements. Vu la dimension actuelle des supports magnétiques d'un micro, ce nombre est relativement important.

Les enregistrements sous DBASE II font 1 000 caractères au maximum, et peuvent avoir 32 champs (254 caractères par champ).

12 43 48 ...etc... 1 000

Dans notre exemple, chaque enregistrement contient 5 champs et l'enregistrement total a une longueur de 58 caractères.

Numéro facture	Fournisseur	Description	Montant	Numéro travail	
1	9 10	28 29	43 44	51 52	58

LE TYPE DES INFORMATIONS

Comme nous l'avons vu plus tôt, chaque champ doit contenir un seul type de données, nous citons ci-dessous les types utilisés par dBase :

Caractère : toute chaîne de caractères ASCII, y compris les nombres entiers, les symboles et les espaces.

Numérique : les valeurs négatives et positives sont comprises entre : 1.8×10^{63} et 1.8×10^{-63}. L'exactitude est de 10 nombres.

Logique : Les valeurs logiques sont des champs d'un caractère et leurs valeurs sont ou Vraie ou Fausse (O/N). dBase reconnaît les caractères **T**, **t**, **Y** et **y** comme **VRAI** alors que les caractères **F**, **f**, **N**, et **n** sont reconnus comme **FAUX**.

Nom de champ : Afin que dBASE identifie les différents champs et retrouve vos données, vous devez attribuer un nom à chaque champ. Un nom de champ est composé d'un maximum de 10 caractères dont le premier est impérativement une lettre. Les éventuels neuf suivants seront des lettres ou des chiffres ainsi que le séparateur « : ».

Exemples :

A	correct
A123456789	correct
Nom :Réf	correct (les minuscules ou majuscules sont autorisées).
A123,B456	incorrect (la virgule n'est pas autorisée).
Client :	incorrect (les « : » ne sont pas utilisés en tant que séparateur).

CONSEIL : Afin de faciliter la compréhension du nom, utilisez le maximum de caractères permis (exemple : num :ref est plus significatif que num). L'utilisation d'un maximum de 9 caractères par nom de champ facilitera les manipulations de variables mémoire, le libellé de ces dernières pourra être le libellé des champs précédé, par exemple, de la lettre « M » (mémoire).

ENCORE UN CONSEIL : Lorsque vous écrivez vos fichiers de commandes, utilisez les lettres majuscules pour les commandes de dBase II et les lettres minuscules pour les champs, les variables ou autres données dont vous avez le contrôle.

LES TYPES DE FICHIERS DE DBASE II

Les **noms de fichiers** sont limités à une longueur de 8 caractères avec, après le point, 3 caractères d'extension. Vous pouvez insérer les « : » dans le nom du fichier mais alors vous ne pourrez utiliser ce fichier que sous dBase II : CP/M stockera le nom du fichier avec les « : », mais ne le reconnaîtra pas lorsque vous voudrez utiliser une fonction telle que PIP.

Un fichier dBase II est simplement une collection d'informations. dBase II opère avec les 6 différents types de fichiers décrits ci-dessous :

.DBF Fichiers de données. C'est dans ce fichier que seront stockées vos données. L'extension .DBF est automatiquement ajoutée par dBase II lorsque vous créez un nouveau fichier avec la commande CREATE. Ces fichiers peuvent contenir jusqu'à 65 535 enregistrements et sont incompatibles avec un traitement de texte.

.FRM Fichiers de rapport. Ils sont créés par la commande REPORT (qui génère des états de sortie standard). Ces fichiers contiennent la structure d'un état à imprimer. Ils

peuvent être modifiés par un traitement de texte mais cette pratique est hasardeuse et déconseillée.

.CMD Fichiers de programmes. Ils contiennent une séquence de commandes DBASE II exécutant les fonctions que vous désirez d'une façon répétitive (ex : un système de paye complet et complexe). Ces fichiers peuvent être modifiés avec un éditeur ou un traitement de texte.

.NDX Fichiers indexés. Ils sont créés par la commande INDEX. Ces fichiers indexés permettent de localiser rapidement les données.

.MEM Fichiers mémoire. Ces fichiers sont créés automatiquement lorsque vous sauvegardez les résultats de vos calculs, les constantes, ou variables avec la commande SAVE. Vous pouvez sauvegarder 64 variables (chacune d'une longueur de 254 caractères) et les ramener en mémoire avec la commande RESTORE.

.TXT Fichiers de texte. Ces fichiers sont créés par les commandes SET ALTERNATE TO, SET ALTERNATE ON, ou par la commande COPY...SDF. Ils mémorisent toute information provenant de votre terminal sur disque. Ces informations peuvent être plus tard éditées, imprimées et/ou sauvegardées.

.FMT Fichiers d'état. Ces fichiers peuvent contenir uniquement des instructions @...SAY et des commentaires précédés par un *. Utilisez ces fichiers pour le formatage de vos écrans avec les commandes APPEND, INSERT et EDIT.

RESUME DES OPERATIONS DBASE

Les opérateurs arithmétiques

Ils génèrent des résultats arithmétiques.

( ) : parenthèses pour ensemble
* : multiplication
/ : division
\+ : addition
— : soustraction

Les opérateurs relationnels

Ils génèrent des résultats logiques.

<	: inférieur à
>	: supérieur à
=	: égal à
< >	: non égal à
< =	: inférieur ou égal à
> =	: supérieur ou égal à

Les opérateurs logiques

Ils génèrent des résultats logiques.

( )	: parenthèses pour ensemble
.NOT.	: négation
.AND.	: ET
.OR.	: OU
$	: opérateur de sous-chaîne logique. (ex : la chaîne 1 est-elle dans la chaîne 2 ?)

Les opérations sur les chaînes de caractères

+	: concaténation des chaînes (rapprochement)
—	: concaténation des chaînes (avec élimination des blancs)

RESUME DES FONCTIONS DBASE II

: numéro de l'enregistrement actif.

* : marque d'enregistrement à effacer.

EOF : fin du fichier.

!(<variable/chaine>) : conversion en majuscules.

TYPE(<expression>) : donne le type de l'information (N, C ou L).

INT(<variable/expression>) : partie entière.

VAL(<variable/chaine/sous-chaine>) : conversion d'une chaîne de caractères en valeur numérique.

STR(<expression/variable/nombre>), <longueur> , <decimaux>) : conversion d'une valeur numérique en chaîne de caractères.

LEN(<variable/chaine>) : longueur d'une chaîne.

$(<expression/variable/chaine> , <début> , <longueur>) : choix d'une sous-chaîne.

@(<variable1/chaine1> , <variable2/chaine2>) : recherche d'une sous-chaîne.

CHR(<nombre>) : change la valeur d'un nombre en ASCII.

& : appel à une macro.

FILE(<nom du fichier/var/exp>) : permet de savoir si un fichier existe.

TRIM(<chaine>) : enlève les blancs inutiles à droite d'une chaîne.

RANK(<chaine>) : donne la valeur ASCII d'un caractère. (Cette commande n'est pas fonctionnelle sur les versions antérieures à la version 2.4).

RESUME DES COMMANDES DBASE

Les symboles <...>, encadrant un commentaire, définissent la précision que devra donner l'utilisateur. Les crochets [...] définissent des précisions optionnelles. Dans le cas où plusieurs précisions optionnelles sont possibles, elles seront présentées les unes au-dessous des autres.

? <expression> [,liste>] : affiche sur l'écran une expression (ou une liste d'expressions séparées par des virgules).

@ <coordonnée> [SAY <expression> [USING picture]] [GET <variable> [PICTURE 'picture']] : formate une image à l'écran ou une sortie sur imprimante.

ACCEPT ['message'] TO <variable> : entre une chaîne de caractères à partir de l'écran (apostrophes entourant la chaîne non nécessaires).

APPEND BLANK
APPEND FROM <nom du fichier> [SDF] [FOR <expression>]
[DELIMITED] [FOR <expression>]

ajoute des données à une base.

CANCEL : abandonne l'exécution d'un fichier de commandes.

CHANGE [étendue] FIELD <liste> [FOR <expression>] : change les informations d'une base.

CLEAR : referme les fichiers de dBase et efface les variables mémoire.

CONTINUE : continue une recherche (voir commande LOCATE).

COPY [étendue] TO <nom de fichier> [SDF] **[STRUCTURE] [FIELD <liste>] [FOR] <expression> [DELIMITED [WITH 'délimiteur']]**

copie des données d'une base à partir d'un autre fichier.

COPY TO <nom du fichier> STRUCTURE EXTENDED : crée un nouveau fichier dont les enregistrements montrent la structure de l'ancien fichier (voir CREATE <nouveau fichier> FROM <ancien fichier>).

COUNT [étendue] [FOR <expression>] [TO <variable>] : compte le nombre d'enregistrements répondant à un certain critère.

CREATE [<nom du fichier>] : crée une nouvelle base de données.

CREATE <nouveau fichier> FROM <ancien fichier> : crée un nouveau fichier dont la structure des données est définie dans l'ancien fichier. (Voir la commande COPY STRUCTURE EXTENDED).

DELETE [étendue] [FOR <expression>] : marque un enregistrement pour effacement.

DELETE FILE <nom du fichier> : efface un fichier.

DISPLAY [étendue] [FOR <expression>] [OFF] : liste les champs que vous sélectionnez.

DISPLAY STRUCTURE : liste la structure d'une base de données.

DISPLAY MEMORY : liste le contenu des variables mémoire.

DISPLAY FILES [ON unité de disque] : liste le répertoire du disque.

DISPLAY STATUS : montre les fichiers en cours, les fichiers indexés et les clés ainsi que les paramètres mis par SET.

DO <nom du programme> : exécute un fichier de commandes.

DO WHILE <expression> : exécute une séquence de commandes répétitivement.

EDIT : permet d'altérer une donnée dans une base.

EDIT [numero] : altère l'enregistrement n.

EJECT : permet un saut de page à l'imprimante.

ELSE : alterne l'exécution de la commande IF.

ENDDO : termine une commande DO WHILE.

ENDIF : termine une commande IF.

ENDTEXT : termine une commande TEXT.

ERASE : efface l'écran.

FIND <clé> : recherche un enregistrement dans une base de données indexée.

GO ou GOTO [enregistrement n] ou [TOP] ou [BOTTOM] : se positionne sur l'enregistrement n, au début du fichier ou à la fin du fichier.

HELP [<verbe de commande>] : explique brièvement la fonction de cette commande.

IF <expression> : exécution conditionnelle d'une séquence.

INDEX ON <clé> TO <nom du fichier> : crée un fichier indexé.

INPUT ['message'] TO <variable> : permet d'affecter la saisie d'une variable mémoire de l'utilisateur par l'écran.

INSERT [BEFORE] ou **INSERT [BLANK]** : insère un enregistrement dans la base de données.

JOIN TO <nom du fichier> FOR <expression> [FIELDS <list>] : fusionne deux bases de données.

LIST : liste les enregistrements.

LOCATE [étendue] [FOR <expression>] : recherche une information répondant aux conditions de l'<expression>.

LOOP : instruction de saut dans une boucle DO WHILE.

NOTE ou * : commentaires dans le programme.

MODIFY COMMAND <nom de fichier> : permet de modifier en direct les fichiers de commandes.

MODIFY STRUCTURE : change la structure d'une base de données. Détruit les données de cette base.

PACK : élimine physiquement les enregistrements marqués d'une *.

QUIT [TO fichier.COM ou commandes de CP/M] : termine une session DBASE, et enchaîne l'exécution d'un autre programme (chaque commande doit être entourée par des apostrophes et séparée par des virgules).

READ : permet l'édition plein écran et accepte les données (voir « @ » et GET).

RECALL [étendue] FOR <expression> : restitue les enregistrements marqués pour effacement (cette commande annule l'effet de DELETE).

RELEASE [<variable>[,liste]] ou [ALL] : efface les variables mémoire.

REMARK : commentaires du programme qui apparaissent pendant l'exécution de celui-ci (contraire de NOTE ou @).

RENAME <ancien fichier> TO <nouveau fichier> : change le nom d'un fichier.

REPLACE [étendue] <champ> WITH <expression>[,<champ> WITH <expression>] : change les données d'une base. (Attention, faites d'abord une sauvegarde de votre base de données si vous n'êtes pas sûr de vous).

REPORT [étendue] [FORM <nom du fichier>] [TO PRINT] [FOR <expression>] : génère des états à imprimer.

RESET : signale à CP/M que la disquette a été échangée.

RESTORE FROM <nom du fichier> : rappelle les variables mémoire sauvegardées dans un fichier (SAVE) en détruisant les variables existantes.

RETURN : termine un programme appelé pour revenir au programme appelant.

SAVE TO <nom du fichier> : sauvegarde les variables mémoire pour utilisation future.

SELECT [PRIMARY] ou [SECONDARY] : permet de changer du secteur primaire au secteur secondaire et vice versa.

SET paramètre [ON] ou [OFF] ou [TO <condition, nom du fichier>] : reconfigure les opérations de dBase.

SKIP ± n : permet de revenir à l'enregistrement précédent ou de passer au suivant par saut de n.

SORT ON <clé> TO <nom du fichier> [ASCENDING]
[DESCENDING]
génère une base de données triée par rubriques.

STORE <expression> TO <variable> : affecte une valeur dans une variable.

SUM [étendue] <rubriques[,liste]> TO <variable> [FOR <expression>] : totalise les rubriques d'une base.

TEXT : visualise un texte sans formatage spécial, tant que ENDTEXT n'est pas trouvé.

TOTAL TO <nom du fichier> ON <clé> [FIELDS <rubrique>[,liste] : génère un fichier avec des sous-totaux dans les enregistrements.

UPDATE FROM <nom du fichier> ON <clé> [ADD <rubriques[,liste]>]
[REPLACE <rubriques [,liste]>] :
modifie une base de données à partir d'une autre base.

USE <nom du fichier> INDEX <nom du fichier> : ouvre une base de données indexée pour utilisation future.

USE : ferme les bases actives.

WAIT [TO <variable>] : le programme s'arrête et attend une réponse de l'opérateur.

LES COMMANDES DBASE GROUPEES PAR SOUS-ENSEMBLES

Les structures des fichiers

CREATE : crée et définit une nouvelle structure de base de données.

CREATE <nouveau fichier> FROM <ancien fichier> : crée un nouveau fichier dont la structure des enregistrements est décrite dans l'ancien fichier.

USE <ancien fichier>.
COPY TO <nouveau fichier> STRUCTURE :
la combinaison de ces 2 commandes crée un nouveau fichier dont la structure est la même que l'ancien fichier.

USE <ancien fichier>.

COPY TO <nouveau fichier> STRUCTURE EXTENDED : crée un nouveau fichier qui contient comme données la structure de l'ancien fichier.

LIST STRUCTURE ou DISPLAY STRUCTURE : liste la structure d'un fichier.

MODIFY STRUCTURE : modifie la structure d'un fichier mais détruit toute les données du fichier.

COMMENT CHANGER LA STRUCTURE D'UNE BASE DE DONNEES REMPLIE

USE <ancien fichier>
COPY TO <nouveau fichier>
USE <nouveau fichier>
MODIFY STRUCTURE
APPEND FROM <ancien fichier>
COPY TO <ancien fichier>
USE <nouveau fichier>
DELETE FILE <nouveau fichier>

COMMENT RENOMMER UN CHAMP REMPLI DE DONNEES DANS UNE BASE

USE <ancien fichier>
COPY TO <nouveau fichier> SDF
MODIFY STRUCTURE
APPEND FROM <nouveau fichier>.TXT SDF
DELETE FILE <nouveau fichier>

MANIPULATION DES FICHIERS

USE <nom du fichier> : ouvre les fichiers.

USE <nouveau fichier> : active un nouveau fichier et désactive l'ancien.

USE : désactive tous les fichiers.

RENAME <ancien nom> TO <nouveau nom> : La base renommée ne doit pas être en USE.

COPY TO <nouveau fichier> : copie la base (crée une sauvegarde).

CLEAR : ferme les fichiers et remet à blanc les variables mémoire.

SELECT [PRIMARY] [SECONDARY] : permet l'ouverture simultanée de deux fichiers. Les données sont transferées avec un préfixe P. et S.

DISPLAY FILES [ON <unité de disque>] : affiche le contenu d'un disque.

DISPLAY FILES LIKE <nom générique> [ON <unité de disque>] : affiche les fichiers particuliers du disque.

QUIT : termine une session DBASE et ferme tous les fichiers.

REORGANISATION D'UNE BASE

SORT ON <clé> TO <nouveau fichier>
INDEX ON <clé> TO <nouveau fichier>
Pouvons utiliser des clés pour ces deux commandes.

FUSION DE BASES

COPY TO <nouveau fichier> : crée une copie du fichier en USE.

APPEND FROM <autre fichier> : ajoute à une base des enregistrements provenant d'une autre base.

UPDATE FROM <autre fichier> ON <clé> : Met à jour des informations d'une base à partir d'une autre base (les 2 bases de données doivent être triées).

JOIN : crée un troisième fichier à partir de deux autres fichiers.

EDITION, MISE A JOUR D'UNE BASE

DISPLAY, LIST, BROWSE : visualisent les enregistrements.

DELETE : efface (logiquement) un enregistrement.

RECALL : restitue un enregistrement logiquement effacé (contraire de DELETE).

PACK : compacte une base en effaçant physiquement les enregistrements effacés logiquement.

EDIT : vous laisse changer les informations d'un enregistrement.

REPLACE <champ WITH données> : remplace une ou plusieurs rubriques d'un enregistrement.

@ <coordonnées> GET <variable>
READ
vous permet de changer les variables visualisées.

INSERT [BEFORE] [BLANK] : insère un enregistrement dans une base.

MODIFY COMMAND : modifie un programme.

UTILISATION DES VARIABLES

LIST MEMORY ou DISPLAY MEMORY : listent les variables mémoire.

STORE <**valeur**> TO <**variable**> : charge ou change une variable mémoire.

RELEASE <**variables**> : efface les variables.

SAVE MEMORY TO <**nom du fichier**> : sauve le contenu des variables mémoire sur disque (extenxion .MEM).

RESTORE FROM <**nom de fichier**> : charge en mémoire les variables précédemment sauvegardées sur disque. (Les éventuelles variables mémoire existantes sont détruites lors du chargement).

SAISIE INTERACTIVE

WAIT : arrête le défilement de l'écran, continue avec une clé.

WAIT TO <**variable**> : entre un caractère dans une variable mémoire.

INPUT ['**message**'] TO <**variable**> : entre une information alphanumérique dans une variable mémoire (l'information doit être entourée d'apostrophes).

ACCEPT ['**message**'] TO <**variable**> : même fonction que INPUT, à la seule différence que les apostrosphes ne sont pas requises.

@ <**coordonnée**> SAY ['**message**'] GET <**variable**> [PICTURE]
READ :
visualise les variables mémoire et les remplace par de nouvelles valeurs.

RECHERCHE D'INFORMATIONS

SKIP [+ <**expression**>] : positionne l'enregistrement + n (à partir de l'enregistrement courant).

GO [TO] <**numéro**>, GO TOP, GO BOTTOM : va au nième enregistrement, au début du fichier ou à la fin du fichier.

FIND : recherche une information dans une base indexée.

LOCATE FOR <**expression**>
CONTINUE
recherche séquentiellement une information.

LA SORTIE DE L'INFORMATION

?, LIST, DISPLAY : montre les expressions, les enregistrements, les variables et les structures.

REPORT FORM <**nom de l'état**> : génère des rapports.

@ <**coordonnée**> **SAY** <**variable/expression**> : sortie formatée de l'information à l'écran ou à l'imprimante.

TEXT : visualise un bloc de texte sans formatage spécial.

LA PROGRAMMATION

DO <**nom du programe**> : Exécute un programme.

IF <condition>
ELSE
ENDIF
Exécute conditionnellement une séquence du programme.

DO WHILE <conditions>
ENDDO
Boucle conditionnelle.

DO CASE
CASE <expression>
<commandes>
CASE <expression>
<commandes>
OTHERWISE
<commandes>
ENDCASE
choix multiples.

CHAPITRE VI

GESTION DE STOCK

UN EXEMPLE DE GESTION DE STOCK

L'application de gestion de stock décrite aux pages suivantes a été écrite dans le but de vous familiariser avec les différentes commandes de dBASE II.
Cette application utilise deux fichiers :

— un fichier articles
— un fichier demandes d'achat

Elle vous montrera comment enchaîner différents menus, comment utiliser simultanément deux fichiers de données en lecture ou en écriture, les différentes possibilités de contrôle de saisie et de recherche d'un enregistrement.

Les différents fichiers de commandes de cette application comportent de nombreux commentaires et explications sur les instructions utilisées.

FICHIERS DE REPORT

Vous trouverez ci-dessous les différents fichiers de report utilisés dans l'application définie à la première page de ce chapitre.

ETATMINI.FRM

```
M=0,W=71
Y
PIECES NECESSITANT UNE COMMANDE DE REACTUALISATION
```

```
N
Y
N
12,PIECE
<REFERENCE;____________
25,LIBELLE
<DESIGNATION;_________________________
8,QTE:STOCK
>STOCK;________
N
8,MINI:STOCK
>MINIMUM;________
N
7,QTE:RESTE
>RESTE;_______
N
10,(MINI:STOCK-QTE:STOCK-QTE:RESTE)*PRIX:UHT
>COUT CDE;__________
Y
```

SELESTOC.FRM

```
W=71,M=0
Y
ETAT DU STOCK SELON VOTRE SELECTION
N
Y
N
15,PIECE
<REFERENCE;_______________
20,LIBELLE
<DESIGNATION;____________________
6,QTE:STOCK
>STOCK;______
N
9,PRIX:UHT
>PRIX UHT;_________
N
12,QTE:STOCK * PRIX:UHT
>PRIX THT;___________
Y
```

SELECTDA.FRM

```
W=70,M=0
Y
LISTE DES COMMANDES AVEC DES DELAIS DE LIVRAISON NON RESPECTES
N
Y
N
12,DA:NUM
<NUMERO DA;__________
14,CMD:NUM
<NUMERO CDE;_____________
15,EMETTEUR
<EMETTEUR;_____________
15,FOURNI
<FOURNISSEUR;_____________
12,DELAI
>N. SEMAINE;__________
N
```

EMETDA.FRM

```
M=0,W=72
Y
LISTE NOMINATIVE DE DEMANDES D'ACHAT
N
Y
N
12,DA:NUM
<NUMERO D.A.;__________
9,DA:DATE
<DATE;_________
13,PIECE
<REFERENCE;___________
5,QTE:CMD
>QTE;_____
N
```

```
9,PRIX:UHT
>PRIX:UHT;_________
N
11,PRIX:UHT * QTE:CMD
>PRIX THT;__________
Y
12,EMETTEUR
<EMETTEUR;__________
```

DADATE.FRM

```
W=50,M=2
Y
ETAT DES DEMANDES D'ACHAT EMISES DANS UNE PERIODE DONNEE
N
Y
N
10,DA:NUM
<NUMERO;_________
11,DA:DATE
<DATE EMIS.;_________
6,QTE:CMD
>QTE.;______
N
10,PRIX:UHT
<PRIX UHT;_________
N
12,QTE:CMD * PRIX:UHT
PRIX THT;__________
Y
```

STRUCTURES DE FICHIERS

STRUCTURE DU FICHIER : STOCK.DBF

CHAMP	NOM	TYP	DIM	DECIMALE(S)
001	ATELIER	C	005	
002	SECTION	C	005	
003	MACHINE	C	025	
004	LIBELLE	C	035	
005	PIECE	C	012	
006	DA:NUM	C	010	
007	QTE:RESTE	N	005	
008	QTE:STOCK	N	006	
009	PRIX:UHT	N	006	002
010	MINI:STOCK	N	004	
** TOTAL ***			00117	

(Indexé sur PIECE to PIECE.NDX)

STRUCTURE DU FICHIER : DA.DBF

CHAMP	NOM	TYP	DIM	DECIMALE(S)
001	DA:NUM	C	010	
002	DA:DATE	C	008	
003	CMD:NUM	C	012	
004	CMD:DATE	C	008	
005	TYPE	C	015	
006	EMETTEUR	C	025	
007	SECTION	C	005	
008	FOURNI	C	040	
009	PIECE	C	012	
010	QTE:CMD	N	005	
011	IMPUTATION	C	025	
012	DELAI	N	002	
013	PRIX:UHT	N	009	002
** TOTAL **			00177	

(Indexé sur DA:NUM to DA.NDX)

FICHIER DE COMMANDE INITIAL

INITIAL.CMD

```
****************** FICHIER DE COMMANDE INITIAL *****************
* Ce fichier de commande est le premier a être exécuté. Il permet *
* la saisie et le contrôle de la date du jour avant d'enchaîner *
* le traitement sur le menu principal : MENU2                    *
*****************************************************************

* SET TALK OFF INTERDIT DE RENVOYER VERS L'ECRAN LES DIFFERENTS
* MESSAGES GENERES PAR DBASE II LORS D'UN TRAITEMENT
SET TALK OFF

* SET COLON OFF PERMET DE NE PAS ENCADRER PAR : ":", LES
* ZONES DE SAISIE ( INSTRUCTION "GET" )
SET COLON OFF
* EFFACEMENT DE L'ECRAN
ERASE
@ 10,25 SAY "Donnez la date "
* AFFECTATION DE LA VARIABLE : DATE
STORE "        " TO DATE
*****************************************************************
* Toutes les instructions situées entre DO WHILE CONTDATE = "0" et *
* ENDDO seront exécutées tant que la variable CONTDATE sera égale *
* à "0". Pour sortir d'une boucle de ce type, il faut : soit modi- *
* fier le contenu de la variable -- dans notre cas, modifier le *
* contenu de CONTDATE --, soit utiliser une commande de sortie *
* comme QUIT ou RETURN, soit lancer un nouveau fichier de com- *
* mande. Ce sera cette dernière possibilité que nous utiliserons *
* dans notre exemple : DO MENU2.                                 *
*****************************************************************
```

```
STORE "0" TO CONTDATE
DO WHILE CONTDATE="0"
* SAISIE DE LA DATE
@ 10,40 GET DATE PICTURE "99/99/99"
READ
* VERIFICATION DE LA DATE SAISIE
STORE VAL($(DATE,1,2)) TO J
STORE VAL($(DATE,4,2)) TO M
STORE VAL($(DATE,7,2)) TO AN
STORE INT(AN/4) TO CALCUL
IF (AN/4)-CALCUL=0
STORE "OUI" TO BISSEXTILE
ELSE
STORE "NON" TO BISSEXTILE
ENDIF

IF AN<81 .OR. M<1 .OR. M>12 .OR. J<1 .OR. J>31 .OR. (( M=4 .OR. M=6 .OR.;
M=9 .OR. M=11) .AND. J>30 ) .OR. ( M=2 .AND. J>29) .OR. (BISSEXTILE="NON";
 .AND. M=2 .AND. J>28)
@ 22,0 SAY "Date incorrecte, appuyez sur une touche"
SET CONSOLE OFF
WAIT
SET CONSOLE ON
@ 22,0 SAY "                                                  "
ELSE
* CALCUL DU NUMERO DE SEMAINE
STORE INT(((M-1)*30+J+6)/7) TO SEMAINE
* ANNULATION DES VARIABLES MEMOIRE INUTILES
RELEASE J,M,AN,CONTDATE,BISSEXTILE,CALCUL
* AFFICHAGE DU DEUXIEME MENU
DO MENU2
ENDIF
* FIN DU MENU INITIAL
ENDDO INITIAL
```

FICHIER DE COMMANDE MENU2.CMD

MENU2.CMD

```
********************** MENU PRINCIPAL *********************************
* Ce menu est le menu principal de l'application. Il vous proposera un *
* choix de trois sélections primaires. Selon votre reponse, il exécu- *
* tera le traitement correspondant en enchainant des menus secondaires *
***********************************************************************
STORE '0' TO CONTMENU2
DO WHILE CONTMENU2 = '0'
STORE 'F' TO REPONSE2
* EFFACEMENT DE L'ECRAN, AFFICHAGE DU MENU, SAISIE DE LA REPONSE
ERASE
@  6, 0 SAY "+-------------------------------------------------------------;
-----------------+"
@  7, 0 SAY "I                                                             ;
                 I"
@  8, 0 SAY "I                  Quel traitement desirez-vous effectuer ?   ;
                 I"
@  9, 0 SAY "I                                                             ;
                 I"
@ 10, 0 SAY "I Modification ou consultation des pieces detachees ======> 1 ;
                 I"
@ 11, 0 SAY "I                                                             ;
                 I"
@ 12, 0 SAY "I Modification ou consultation des demandes d'achat ======> 2 ;
                 I"
@ 13, 0 SAY "I                                                             ;
                 I"
@ 14, 0 SAY "I Fin de traitement                                  ======> F ;
                 I"
@ 15, 0 SAY "I                                                             ;
                 I"
@ 16, 0 SAY "+-------------------------------------------------------------;
-----------------+"
@ 17, 0 SAY "+----------------------+"
@ 18, 0 SAY "I Reponse -->           I"
@ 19, 0 SAY "+----------------------+"
@ 18,14 GET REPONSE2
```

```
READ
* FIN DU TRAITEMENT SI LA REPONSE EST : "F"
IF !(REPONSE2) = 'F'
QUIT
ELSE
* TRAITEMENT DU STOCK SI LA REPONSE EST : "1"
        IF REPONSE2 = '1'
        DO TRASTOCK
        ELSE

* TRAITEMENT DES DEMANDES D'ACHAT SI LA REPONSE EST : "2"
                IF REPONSE2 = '2'
                DO TRADA
                ELSE
* SI LA REPONSE EST DIFFERENTE DE "1", "2" OU "F" -> MESSAGE D'ERREUR
                        @ 22,0 SAY 'VOTRE SELECTION EST INVALIDE, VALIDEZ'
* ARRET DE L'AFFICHAGE A L'ECRAN, POUR NE PAS AFFICHER LE MESSAGE
* D'ATTENTE
                        SET CONSOLE OFF
                        WAIT
* ANNULATION DE LA COMMANDE D'ARRET DE L'AFFICHAGE, APRES LA SAISIE
* D'UN CARACTERE
                        SET CONSOLE ON
                ENDIF 2
        ENDIF 1
ENDIF F
ENDDO MENU2
```

FICHIER DE COMMANDE TRASTOCK.CMD

TRASTOCK.CMD

```
********** TRAITEMENT DES PIECES DETACHEES **********
* Ce fichier de commande vous propose différentes options de traitement *
* au sein du fichier des pièces détachées. Selon votre sélection, il *
* lancera le traitement correspondant. La réponse de defaut : "F" vous *
* permettra de revenir au menu principal : MENU2                      *
*************************************************************************
```

```
STORE '0' TO CONTSTOCK
DO WHILE CONTSTOCK = '0'

* Effacement de l'ecran et affichage des differentes options
ERASE
STORE 'F' TO REPONSE3
@  1,27 SAY "TRAITEMENT DES PIECES DETACHEES"
@  5, 0 SAY "+-----------------------------------------------------------------;
-----------------+"
@  6, 0 SAY "I                                                                 ;
                  I"
@  7, 0 SAY "I                            SELECTIONNEZ VOTRE OPTION            ;
                  I"
@  8, 0 SAY "I                                                                 ;
                  I"
@  9, 0 SAY "I                                                                 ;
                  I"
@ 10, 0 SAY "I  AJOUT DE NOUVEAUX ENREGISTREMENTS -> 1                         ;
                  I"
@ 11, 0 SAY "I                                                                 ;
                  I"
@ 12, 0 SAY "I  MODIFICATION D'ENREGISTREMENTS    -> 2                        ;
                  I"
@ 13, 0 SAY "I                                                                 ;
                  I"
@ 14, 0 SAY "I  ENTREE/SORTIE DE PIECE DU STOCK   -> 3                         ;
                  I"
@ 15, 0 SAY "I                                                    REPONSE      ;
                  I"
@ 16, 0 SAY "I  VISUALISATION D'ENREGISTREMENTS   -> 4                         ;
                  I"
@ 17, 0 SAY "I                                                                 ;
                  I"
@ 18, 0 SAY "I  SORTIE D'ETAT SUIVANT SELECTION   -> 5                         ;
                  I"
@ 19, 0 SAY "I                                                                 ;
                  I"
@ 20, 0 SAY "I  RETOUR AU MENU INITIAL            -> F                         ;
                  I"
@ 21, 0 SAY "+-----------------------------------------------------------------;
-----------------+"
@ 15,60 GET REPONSE3
```

```
* Saisie de la reponse
READ

* Si la reponse est : "f" ou "F" --> retour au menu principal (MENU2)
IF !(REPONSE3) = 'F'
RELEASE CONTSTOCK,REPONSE3,REPONSE4,MODESTOCK,IMPRIMANTE
RETURN

* Si la reponse est differente de "f" ou de "F", nous lancerons
* le traitement correspondant

ELSE

 IF REPONSE3 = '1'

********** CREATION D'UN NOUVEL ENREGISTREMENT **********
* Le traitement correspondant a cette option est la  creation *
* d'un nouvel enregistrement. Le fichier de commande STOCKSCR *
* permet l'affichage de la presentation d'un  enregistrement. *
* Ce fichier ( STOCKSCR ), sera utilise pour tous les traite- *
* ments qui necessitent l'affichage de la  presentation  d'un *
* enregistrement. Pour eviter toute ambiguite,  on  affichera *
* le mode en cours en haut a droite de l'ecran. Pour  obtenir *
* ce resultat,  il  suffira  d'affecter le mode en cours a la *
* variable : MODESTOCK,  avant d'executer le fichier de  com- *
* mande STOCKSCR. Lors de son execution, if affichera le con- *
* tenu de la variable :  MODESTOCK  en  haut et a  droite  de *
* l'ecran.                                                    *
*************************************************************

 STORE 'AJOUT' TO MODESTOCK
 DO STOCKSCR

* Execution du fichier de commande permettant la creation  de
* nouveaux enregistrements.
 DO AJSTOCK
 USE
 ELSE
   IF REPONSE3 = '2'
   STORE "MODIFICATION" TO MODESTOCK
   DO STOCKSCR
```

```
* Execution du fichier de commande permettant la modification
* d'un enregistrement.
   DO MODISTOC
   USE
   ELSE
     IF REPONSE3 = '3'
     STORE "ENT/SOR PIECE" TO MODESTOCK
     DO STOCKSCR

* Execution du fichier de commande permettant l'entree ou la
* sortie de piece(s) du stock.
     DO ENTSORT
     USE
     ELSE
       IF REPONSE3 = "4"
       STORE "VISUALISATION" TO MODESTOCK
       DO STOCKSCR

* Execution du fichier de commande permettant la consultation
* de piece(s).
       DO VISTOCK
       USE
       ELSE
         IF REPONSE3 = "5"

* Execution du fichier de commande permettant la generation
* de rapport(s).
         DO RESTOCK
         USE
         ELSE
```

```
* Si la selection ne correspond a aucune des options proposees,
* le message suivant sera affiche.
        @ 22,0 SAY 'AUCUN TRAITEMENT NE CORRESPOND A VOTRE SELECTION, ;
VALIDEZ'
           SET CONSOLE OFF
           WAIT
           SET CONSOLE ON
         ENDIF 5
       ENDIF 4
     ENDIF 3
   ENDIF 2
 ENDIF 1
ENDIF F
STORE '0' TO CONTSTOCK
ENDDO TRASTOCK
```

FICHIER DE COMMANDE STOCKSCR.CMD

STOCKSCR.CMD

```
ERASE
@  0, 0 SAY "                                                              +----;
---------------+"
@  1, 0 SAY "                                                              I Mode;
               I"
@  1,65 SAY MODESTOCK
@  2, 0 SAY "+-------------------------------------------------------------+----;
---------------+"
@  3, 0 SAY "I                          GESTION DU STOCK DE PIECES DETACHEES;
                I"
@  4, 0 SAY "I                                                                 ;
                I"
@  5, 0 SAY "I  +-----------------------------------------------------------;
------------+  I"
@  6, 0 SAY "I  I ATELIER          SECTION         MACHINE                    ;
             I  I"
```

```
@  7, 0 SAY "I  I                                                          ;
                I  I"
@  8, 0 SAY "I  I DESIGNATION                                              ;
                I  I"
@  9, 0 SAY "I  I                                                          ;
                I  I"
@ 10, 0 SAY "I  I PIECE                                                    ;
                I  I"
@ 11, 0 SAY "I  +-------------------------------------------------------;
-------------+  I"
@ 12, 0 SAY "I                        DERNIERE COMMANDE                    ;
                I"
@ 13, 0 SAY "I  +-------------------------------------------------------;
----------------+"
@ 14, 0 SAY "I  I No D.A.              date D.A.          No Cde            ;
     du         I"
@ 15, 0 SAY "I  I                                                          ;
                I"
@ 16, 0 SAY "I  I Qte commandee                 reste a livrer             ;
                I"
@ 17, 0 SAY "I  +-------------------------------------------------------;
----------------+"
@ 18, 0 SAY "I                                    Quantite      prix UHT   ;
  prix total    I"
@ 19, 0 SAY "I  Qte MINI                        +------------+------------+--;
----------------+"
@ 20, 0 SAY "I              ETAT DU STOCK =>  I            I            I  ;
                I"
@ 21, 0 SAY "+---------------------------------+------------+------------+--;
----------------+"
```

FICHIER DE COMMANDE AJSTOCK.CMD

AJSTOCK.CMD

```
********** CREATION DE NOUVEAUX ENREGISTREMENTS **********
* OUVERTURE DE LA BASE DE DONNEES DES PIECES EN STOCK. NOUS  UTILISONS *
* UN FICHIER INDEX : "PIECE". CE FICHIER CONTIENT TOUS LES NUMEROS  DE *
* REFERENCE ORGANISES DANS L'ORDRE CROISSANT AINSI QUE LA POSITION  DE *
* L'ENREGISTREMENT CORRESPONDANT DANS LE FICHIER PRINCIPAL. CE FICHIER *
* SERA MIS A JOUR AUTOMATIQUEMENT LORS DE L'AJOUT DE NOUVEAUX ENREGIS- *
* TREMENTS.                                                           *
************************************************************************

SET EXACT ON
USE STOCK INDEX PIECE
STORE '0' TO CONTAJ
DO WHILE CONTAJ = '0'
STORE '              ' TO MPIECE
* EFFACEMENT DES DIFFERENTES ZONES DE SAISIE A L'ECRAN
@ 6,13 SAY  '        '
@ 6,31 SAY  '        '
@ 6,47 SAY  '                              '
@ 8,17 SAY  '                                        '
@ 16,50 SAY '              '
@ 20,3  SAY '              '
@ 20,35 SAY '              '
@ 20,48 SAY '              '
@ 20,61 SAY '                    '
* SAISIE DU NUMERO DE REFERENCE
@ 10,11 GET MPIECE
READ
* SI AUCUN NUMERO DE REFERENCE N'EST SAISIE, ==> FIN DE SAISIE
IF MPIECE = ' '
RELEASE MPIECE,MATELIER,MSECTION,MMACHINE,MLIBELLE
RELEASE MQTE:RESTE,MQTE:STOCK,MPRIX:UHT,MPRIX:THT
SET EXACT OFF
RETURN
ELSE
* RECHERCHE DE L'EXISTENCE DU NUMERO DE REFERENCE SAISIE DANS LE FICHIER
  FIND &MPIECE
```

```
* SI CETTE REFERENCE EXISTE LA VARIABLE INTERNE : '#', CONTENANT LE NUMERO
* DE L'ENREGISTREMENT EN COURS SERA DIFFERENTE DE ZERO
  IF # > 0
  @ 22,0 SAY 'La piece '
  @ 22,9 SAY MPIECE
  @ 22,21 SAY ' est deja enregistree, VALIDEZ'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
  @ 22,0 SAY '
  ELSE
* SI CETTE REFERENCE N'EXISTE PAS --> SAISIE DES DIFFERENTS PARAMETRES
  STORE '      ' TO MATELIER
  STORE '      ' TO MSECTION
  STORE '                          ' TO MMACHINE
  STORE '                                  ' TO MLIBELLE
  STORE 0 TO MQTE:RESTE
  STORE 0 TO MQTE:STOCK
  STORE 0.00 TO MPRIX:UHT
  STORE 0 TO MMINI:STOC
  @ 6,13 GET MATELIER
  @ 6,31 GET MSECTION
  @ 6,47 GET MMACHINE
  @ 8,17 GET MLIBELLE
  @ 16,51 GET MQTE:RESTE
  @ 20,3  GET MMINI:STOC
  @ 20,35 GET MQTE:STOCK
  @ 20,48 GET MPRIX:UHT
  READ
* CALCUL DU MONTANT TOTAL DU STOCK POUR CETTE PIECE
  STORE MPRIX:UHT * MQTE:STOCK TO MPRIX:THT
  @ 20,67 SAY MPRIX:THT
```

```
* ENREGISTREMENT DES INFORMATIONS SAISIES
  APPEND BLANK
  REPLACE ATELIER WITH MATELIER
  REPLACE SECTION WITH MSECTION
  REPLACE MACHINE WITH MMACHINE
  REPLACE LIBELLE WITH MLIBELLE
  REPLACE PIECE WITH MPIECE
  REPLACE QTE:RESTE WITH MQTE:RESTE
  REPLACE QTE:STOCK WITH MQTE:STOCK
  REPLACE PRIX:UHT WITH MPRIX:UHT
  REPLACE MINI:STOCK WITH MMINI:STOC
  ENDIF
ENDIF
ENDDO AJSTOCK
```

FICHIER DE COMMANDE MODISTOC.CMD

MODISTOC.CMD

```
*************** MODIFICATION(S) D'ENREGISTREMENT(S) ***************
* CE FICHIER DE COMMANDE VOUS PERMETTRA DE MODIFIER CERTAINES    *
* DES RUBRIQUES CONSTITUANT UNE PIECE.                           *
* LES RUBRIQUES COMPLEMENTAIRES A UN NUMERO DE DEMANDE D'ACHAT   *
* ( DATE DE LA DEMANDE D'ACHAT, DATE DE LA COMMANDE, NUMERO DE   *
* LA COMMANDE, QUANTITE COMMANDEE ) NE FIGURANT PAS - CELA SERAIT*
* INUTILE - DANS L'ENREGISTREMENT D'UNE PIECE, NOUS UTILISERONS  *
* SIMULTANEMENT DEUX BASES DE DONNEES : 1. BASE STOCK            *
*                                       2. BASE DA               *
* CE QUI NOUS PERMETTRA PAR LE BIAIS DE LA BASE : "DA" D'AFFICHER*
* LES RENSEIGNEMENTS COMPLEMENTAIRES DE LA PIECE EN COURS DE     *
* MODIFICATION.                                                  *
* dBASE II PERMET DE GERER SIMULTANEMENENT DEUX BASES DE DONNEES.*
* APRES L'AFFECTATION DE LA BASE PRIMAIRE ET DE LA BASE SECONDAIRE*
* IL SUFFIRA DE SELECTIONNER LA BASE CONCERNEE ( SELECT PRIMARY OU*
* SELECT SECONDARY ) POUR EFFECTUER UN QUELCONQUE TRAITEMENT.    *
* L'UTILISATION DE CETTE FONCTION PERMET DE RESERVER EN MEMOIRE  *
* CENTRALE UN EMPLACEMENT A CHACUNE DES BASES.                   *
******************************************************************
```

```
SET EXACT ON
* DEFINITION DE LA BASE PRIMAIRE
SELECT PRIMARY
USE DA INDEX DA

* DEFINITION DE LA BASE SECONDAIRE
SELECT SECONDARY
USE STOCK INDEX PIECE

STORE "NON" TO EFFACEMENT
STORE '0' TO CONTMODIF
DO WHILE CONTMODIF = '0'

* SELECTION DE LA BASE SECONDAIRE -> STOCK
SELECT SECONDARY
STORE "N" TO REPONSE
STORE '              ' TO MPIECE
* EFFACEMENT DES DIFFERENTES ZONES DE SAISIE A L'ECRAN
@ 6,13 SAY  '        '
@ 6,31 SAY  '        '
@ 6,47 SAY  '                                '
@ 8,17 SAY  '                                              '
@ 14,13 SAY '            '
@ 14,36 SAY '   /  /   '
@ 14,53 SAY '                '
@ 14,71 SAY '   /  /   '
@ 16,20 SAY '         '
@ 16,51 SAY '             '
@ 20,3  SAY '               '
@ 20,39 SAY '          '
@ 20,48 SAY '                '
@ 20,61 SAY '                    '
* SAISIE DU NUMERO DE REFERENCE
@ 10,11 GET MPIECE
READ
* SI AUCUN NUMERO DE REFERENCE N'EST SAISIE, ==> FIN DE MODIFICATION
IF MPIECE = '  '
RELEASE MPIECE,REPONSE
```

```
* SI UN EFFACEMENT A ETE EXECUTE LE CONTENU DE LA VARIABLE EFFACEMENT
* EST : "OUI". DANS CE CAS ON DETRUIRA DEFINITIVEMENT LES ENREGISTRE-
* MENTS PRECEDEMMENT MARQUES ( DELETE).
IF EFFACEMENT = "OUI"
@ 22,0 SAY "Prise en compte des modifications en cours."
@ 23,0 SAY "Veuillez patienter quelques instants ....."
PACK
ENDIF
SET EXACT OFF
RETURN
ELSE
* RECHERCHE DE L'EXISTENCE DU NUMERO DE REFERENCE SAISIE DANS LE FICHIER
* DES PIECES DETACHEES
  FIND &MPIECE
  IF # = 0 .OR. *
  @ 22,0 SAY 'La piece '
  @ 22,9 SAY MPIECE
  @ 22,21 SAY ' ne figure pas dans le fichier, VALIDEZ'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
  @ 22,0 SAY '
  ELSE

* SELECTION DE LA BASE PRIMAIRE -> DA
  SELECT PRIMARY
  STORE S.DA:NUM TO MDA:NUM
  FIND &MDA:NUM
        IF # = 0
        @ 14,36 SAY "  /  /  "
        @ 14,71 SAY "  /  /  "
        ELSE
        @ 14,13 SAY P.DA:NUM
        @ 14,36 SAY DA:DATE
        @ 14,53 SAY CMD:NUM
        @ 14,71 SAY CMD:DATE
        @ 16,20 SAY P.QTE:CMD
        ENDIF
  @ 16,51 SAY S.QTE:RESTE
  @ 20,39 SAY S.QTE:STOCK
```

```
   STORE S.ATELIER TO MATELIER
   STORE S.SECTION TO MSECTION
   STORE S.MACHINE TO MMACHINE
   STORE S.LIBELLE TO MLIBELLE
   STORE S.MINI:STOCK TO MMINI:STOC
   STORE S.PRIX:UHT TO MPRIX:UHT
   @ 6,13 GET MATELIER
   @ 6,31 GET MSECTION
   @ 6,47 GET MMACHINE
   @ 8,17 GET MLIBELLE
   @ 20,3  GET MMINI:STOC
   @ 20,48 GET MPRIX:UHT
   READ
 * CALCUL DU MONTANT TOTAL DU STOCK POUR CETTE PIECE
   STORE MPRIX:UHT * QTE:STOCK TO MPRIX:THT
   @ 20,67 SAY MPRIX:THT
   SELECT SECONDARY
 * SI AUCUN PARAMETRE N'A CHANGE --> PROPOSITION EFFACEMENT
      IF MATELIER=ATELIER .AND. SECTION=MSECTION .AND. MACHINE=MMACHINE ;
      .AND. LIBELLE=MLIBELLE .AND. MINI:STOCK=MMINI:STOC .AND. ;
      PRIX:UHT=MPRIX:UHT
      @ 22,0 SAY "Voulez-vous l'effacer ? " GET REPONSE
      READ
      @ 22,0 SAY "                                        "
         IF REPONSE = "O"
         DELETE
         STORE "OUI" TO EFFACEMENT
         ENDIF
      ELSE
 * ENREGISTREMENT DES INFORMATIONS MODIFIEES
      REPLACE ATELIER WITH MATELIER
      REPLACE SECTION WITH MSECTION
      REPLACE MACHINE WITH MMACHINE
      REPLACE LIBELLE WITH MLIBELLE
      REPLACE PIECE WITH MPIECE
      REPLACE PRIX:UHT WITH MPRIX:UHT
      REPLACE MINI:STOCK WITH MMINI:STOC
      ENDIF
   ENDIF
ENDIF
ENDDO MODISTOC
```

FICHIER DE COMMANDE ENTSORT.CMD

ENTSORT.CMD

```
********** ENTREE OU SORTIE DE PIECES DU STOCK **********
* CE FICHIER PERMET DE PRENDRE EN COMPTE L'ENTREE OU LA SORTIE DE PIECES *
* DU STOCK, MAIS SANS CREER AUCUN DOCUMENT.                              *
*************************************************************************

SET EXACT ON
@ 0,0 SAY  "FIN = VALIDEZ"
* DEFINITION DE LA BASE PRIMAIRE
SELECT PRIMARY
USE DA INDEX DA

* DEFINITION DE LA BASE SECONDAIRE
SELECT SECONDARY
USE STOCK INDEX PIECE

STORE "0" TO CONTENTSOR
DO WHILE CONTENTSOR = "0"
SELECT SECONDARY
STORE "            " TO MPIECE
@ 10,11 GET MPIECE
READ

* SI AUCUN NUMERO DE REFERENCE N'EST SAISIE --> RETOUR AU MENU TRASTOCK
IF MPIECE = " "
RELEASE CONTENTSOR
SET EXACT OFF
RETURN
ELSE
SELECT SECONDARY
FIND &MPIECE
```

```
* SI LA REFERENCE SAISIE N'EXISTE PAS --> EFFACEMENT DE TOUTES LES ZONES
* DE MODIFICATIONS DE LA REFERENCE PRECEDENTE, ET AFFICHAGE D'UN MESSAGE
* D'ERREUR
     IF # = 0
     @ 6,13 SAY "      "
     @ 6,31 SAY "      "
     @ 6,47 SAY "                          "
     @ 8,17 SAY "                                  "
     @ 14,13 SAY "          "
     @ 14,36 SAY "        "
     @ 14,53 SAY "            "
     @ 14,71 SAY "        "
     @ 16,20 SAY "          "
     @ 16,51 SAY "          "
     @ 20,3  SAY "          "
     @ 20,35 SAY "          "
     @ 20,48 SAY "            "
     @ 20,64 SAY "             "
     @ 22,0 SAY  "CETTE PIECE NE FIGURE PAS DANS LE FICHIER, Validez"
     SET CONSOLE OFF
     WAIT
     SET CONSOLE ON
     @ 22,0 SAY "                                                              "
     ELSE
* SI LA REFERENCE SAISIE EXISTE --> RECHERCHE DE LA DEMANDE D'ACHAT
* CORRESPONDANTE DANS LE FICHIER : DA ( BASE PRIMAIRE )
     SELECT PRIMARY
     STORE S.DA:NUM TO MDA:NUM
     FIND &MDA:NUM

* SI LA DEMANDE D'ACHAT N'EXISTE PAS --> AFFICHAGE DES DATES SOUS
* LA FORME : "  /  /  "
        IF # = 0
        @ 14,36 SAY "  /  /  "
        @ 14,71 SAY "  /  /  "
```

```
* SI LA DEMANDE D'ACHAT EXISTE --> AFFICHAGE DES RENSEIGNEMENTS
* COMPLEMENTAIRES CORRESPONDANT A CETTE DEMANDE D'ACHAT.
        ELSE
        @ 14,36 SAY DA:DATE
        @ 14,53 SAY CMD:NUM
        @ 14,71 SAY CMD:DATE
        @ 16,20 SAY QTE:CMD
        ENDIF

* AFFICHAGE DES DIFFERENTES CARACTERISTIQUES DE LA PIECE
     @ 6,13 SAY S.ATELIER
     @ 6,31 SAY S.SECTION
     @ 6,47 SAY S.MACHINE
     @ 8,17 SAY S.LIBELLE
     @ 14,13 SAY S.DA:NUM
     @ 16,51 SAY S.QTE:RESTE
     @ 20,3  SAY S.MINI:STOCK
     @ 20,35 SAY S.QTE:STOCK
     @ 20,51 SAY S.PRIX:UHT
     STORE S.PRIX:UHT * S.QTE:STOCK TO PRIX:THT
     @ 20,67 SAY PRIX:THT

* PRISE EN COMPTE DES ENTREES OU SORTIES DE PIECES DU STOCK TANT
* QUE LA FIN DE TRAITEMENT N'EST PAS DEMANDEE
STORE "0" TO CONTINU
DO WHILE CONTINU = "0"
STORE "     " TO QTE
STORE "F" TO REPONSE
@ 22,0 SAY "Est-ce une Reception, une Sortie, la Fin traitement (R/S/F) " ;
GET REPONSE
READ
@ 22,0 SAY "                                                              ;
    "
SELECT SECONDARY
IF !(REPONSE) = "S"

****************** SORTIE DE PIECE(S) DU STOCK ********************
* CETTE OPTION VOUS PERMETTRA DE DECREMENTER LA QUANTITE EN STOCK *
***********************************************************************
```

```
@ 22,0 SAY "Donnez la quantite a sortir du stock " GET QTE PICTURE ;
"9999"
READ
STORE VAL(QTE) TO QUANTITE
* SI LA QUANTITE A DEDUIRE EST SUPERIEURE A CELLE EN STOCK
* UN MESSAGE D'ERREUR SERA AFFICHE, ET AUCUN TRAITEMENT NE
* SERA EFFECTUE.
    IF QUANTITE > S.QTE:STOCK
    @ 23,0 SAY "Transaction impossible, Validez"
    SET CONSOLE OFF
    WAIT
    SET CONSOLE ON
    @ 23,0 SAY "                                        "

* SI LA QUANTITE A DEDUIRE EST INFERIEURE OU EGALE A CELLE
* EN STOCK --> MODIFICATION DE LA QUANTITE EN STOCK,  PUIS
* AFFICHAGE DE LA  NOUVELLE  QUANTITE  ET  DU  PRIX  TOTAL
*  CORRESPONDANT
    ELSE
    REPLACE S.QTE:STOCK WITH S.QTE:STOCK - QUANTITE
    STORE S.QTE:STOCK * S.PRIX:UHT TO PRIX:THT
    @ 20,67 SAY PRIX:THT
    @ 20,35 SAY "         "
    @ 20,35 SAY S.QTE:STOCK
    ENDIF
ELSE
    SELECT SECONDARY
    IF !(REPONSE) = "R"

********************** RECEPTION DE PIECE(S) ************************
* CETTE OPTION VOUS PERMETTRA DE DECREMENTER LA QUANTITE RESTANT A *
* LIVRER ET INCREMENTER SIMULTANEMENT LA QUANTITE EN STOCK.        *
*********************************************************************

    @ 22,0 SAY "Donnez la quantite a entrer en stock " GET QTE ;
PICTURE "9999"
    READ
STORE VAL(QTE) TO QUANTITE

* SI LA QUANTITE SAISIE EST SUPERIEURE A CELLE RESTANT A LIVRER
* UN MESSAGE D'ERREUR SERA AFFICHE, ET AUCUN TRAITEMENT NE SERA
```

```
* EFFECTUE.
        IF QUANTITE > S.QTE:RESTE
        @ 23,0 SAY "Transaction impossible, Validez"
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
        @ 23,0 SAY "                                        "

* SI LA  QUANTITE A  DEDUIRE EST INFERIEURE OU EGALE A CELLE
* RESTANT A LIVRER --> MODIFICATION DE LA QUANTITE EN STOCK,
* PUIS AFFICHAGE DE LA NOUVELLE QUANTITE EN STOCK ET DU PRIX
* TOTAL CORRESPONDANT
        ELSE
        REPLACE S.QTE:RESTE WITH S.QTE:RESTE - QUANTITE
        REPLACE S.QTE:STOCK WITH S.QTE:STOCK + QUANTITE
        STORE S.QTE:STOCK * S.PRIX:UHT TO PRIX:THT
        @ 20,67 SAY PRIX:THT
        @ 20,35 SAY S.QTE:STOCK
        @ 16,51 SAY "      "
        @ 16,51 SAY S.QTE:RESTE
        ENDIF
    ELSE

************************ FIN DE TRAITEMENT ************************
* CETTE OPTION TERMINE LA MODIFICATION ( RECEPTION OU SORTIE  DE *
* PIECE(S) DU STOCK ), ET VOUS PERMET DE PROCEDER A L'EVENTUELLE *
* MODIFICATION D'UNE AUTRE REFERENCE.                            *
*****************************************************************

        IF !(REPONSE) = "F"
        STORE "N" TO CONTINU

* SI LA REPONSE EST DIFFERENTE DE : "R","S" OU "F" --> AFFICHAGE
* D'UN MESSAGE D'ERREUR.
        ELSE
        @ 22,0 SAY "AUCUN TRAITEMENT NE CORRESPOND A VOTRE SELECTION    "
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
        ENDIF FIN TRAITEMENT D'UNE PIECE
    ENDIF RECEPTION
```

```
ENDIF SORTIE
ENDDO CONTINU
    ENDIF EXISTENCE D'UNE PIECE
ENDIF SAISIE NUMERO DE REFERENCE
ENDDO CONTENTSOR
ENDDO
```

FIICHIER DE COMMANDE VISTOCK.CMD

VISTOCK.CMD

```
************** VISUALISATION D'ENREGISTREMENT(S) ***************
* CE FICHIER DE COMMANDE PERMET DE CONSULTER UN ENREGISTREMENT *
* SELON SON NUMERO DE REFERENCE.                               *
****************************************************************

SET EXACT ON
@ 0,0 SAY  "FIN = VALIDEZ"
STORE " " TO VALIDEZ
SELECT PRIMARY
USE DA INDEX DA
SELECT SECONDARY
USE STOCK INDEX PIECE
STORE "            " TO MPIECE
STORE "0" TO CONTVISU
DO WHILE CONTVISU = "0"
@ 10,11 GET MPIECE
READ

* SI AUCUN NUMERO DE REFERENCE N'EST SAISI OU SI IL EST IDENTIQUE
* AU PRECEDENT -> RETOUR AU MENU TRAITEMENT PIECES DETACHEES.

IF MPIECE = " " .OR. VALIDEZ = MPIECE
RELEASE VALIDEZ,CONTVISU,MPIECE,MDA:NUM,COLONNE,QUANTITE,PRIX:THT
SET EXACT OFF
RETURN
ELSE
```

```
* EFFACEMENT DE LA 22eme LIGNE SUR LAQUELLE SUBSISTE PEUT ETRE
* UN MESSAGE D'AVERTISSEMENT.
@ 22,0 SAY "                                                    ;
                    "

STORE MPIECE TO VALIDEZ
SELECT SECONDARY
FIND &MPIECE

* SI LE NUMERO DE REFERENCE SAISI  N'EXISTE  PAS --> EFFACEMENT
* DES DIFFERENTE ZONE DE VISUALISATION ET AFFICHAGE D'UN MESSAGE
* D'ERREUR.
     IF # = 0
     @ 6,13 SAY "      "
     @ 6,31 SAY "      "
     @ 6,47 SAY "                          "
     @ 8,17 SAY "                                    "
     @ 14,13 SAY "          "
     @ 14,36 SAY "         "
     @ 14,53 SAY "            "
     @ 14,71 SAY "         "
     @ 16,20 SAY "          "
     @ 16,51 SAY "          "
     @ 20,3  SAY "          "
     @ 20,35 SAY "          "
     @ 20,48 SAY "            "
     @ 20,64 SAY "          "
     @ 22,0 SAY  "CETTE PIECE NE FIGURE PAS DANS LE FICHIER, Validez"
     SET CONSOLE OFF
     WAIT
     SET CONSOLE ON
     @ 22,0 SAY "                                                    "

* SI LE NUMERO DE REFERENCE SAISI EXISTE --> RECHERCHE DES RENSEI-
* GNEMENTS COMPLEMENTAIRES AU NUMERO DE LA DEMANDE D'ACHAT.

     ELSE
     SELECT PRIMARY
     STORE S.DA:NUM TO MDA:NUM
     FIND &MDA:NUM
```

```
* SI LE NUMERO DE LA DEMANDE D'ACHAT N'EXISTE PLUS --> AFFICHAGE
* DES DATES SOUS LA FORME "  /  /  "
        IF # = 0
        @ 14,36 SAY "  /  /  "
        @ 14,71 SAY "  /  /  "

* SI LA DEMANDE D'ACHAT EXISTE --> AFFICHAGE DES DIFFERENTS
* RENSEIGNEMENTS COMPLEMENTAIRES.
        ELSE
        @ 14,36 SAY DA:DATE
        @ 14,53 SAY CMD:NUM
        @ 14,71 SAY CMD:DATE
        @ 16,20 SAY QTE:CMD
        ENDIF

* AFFICHAGE DES CARACTERISTIQUES DE L'ARTICLE
     @ 6,13 SAY S.ATELIER
     @ 6,31 SAY S.SECTION
     @ 6,47 SAY S.MACHINE
     @ 8,17 SAY S.LIBELLE
     @ 14,13 SAY S.DA:NUM
     @ 16,51 SAY S.QTE:RESTE
     @ 20,3  SAY S.MINI:STOCK

* AFFICHAGE D'UN MESSAGE D'AVERTISSEMENT LORSQUE LA QUANTITE
* EN STOCK EST INFERIEURE A LA QUANTITE MINIMUM.

        IF S.QTE:STOCK < MINI:STOCK
        @ 22,0 SAY "ATTENTION, La quantite en stock est inferieure ;
a la quantite minimum"
        ELSE
        ENDIF
     @ 20,35 SAY "          "
     @ 20,35 SAY S.QTE:STOCK
     @ 20,51 SAY S.PRIX:UHT
     STORE S.PRIX:UHT * S.QTE:STOCK TO PRIX:THT
     @ 20,64 SAY PRIX:THT
     ENDIF
ENDIF
ENDDO VISTOCK
```

FICHIER DE COMMANDE RESTOCK.CMD

RESTOCK.CMD

```
********** GENERATION D'ETAT POUR LE FICHIER STOCK *************
* CE FICHIER DE COMMANDE VOUS PERMETTRA D'IMPRIMER SUR L'ECRAN *
* ( ET EVENTUELLEMENT SUR L'IMPRIMANTE ), LES  ENREGISTREMENTS *
* CORRESPONDANTS A VOTRE SELECTION.                            *
****************************************************************

USE STOCK INDEX PIECE
STORE '0' TO CONTREPORT
DO WHILE CONTREPORT = '0'
STORE "F" TO REPONSE4
STORE "N" TO IMPRIMANTE
ERASE
@  3,27 SAY "SORTIE D'ETAT"
@  5, 0 SAY "+-----------------------------------------------------;
-------------------------+"
@  6, 0 SAY "I                                                     ;
                          I"
@  7, 0 SAY "I  LISTE DES COMMANDES POUR REACTUALISER LE STOCK     ;
  --> 1                   I"
@  8, 0 SAY "I                                                     ;
                          I"
@  9, 0 SAY "I  LISTE DES PIECES SELON LE LIBELLE DE VOTRE CHOIX   ;
  --> 2                   I"
@ 10, 0 SAY "I                                                     ;
                          I"
@ 11, 0 SAY "I  LISTE DES PIECES SELON UNE PLAGE DE PRIX           ;
  --> 3                   I"
@ 12, 0 SAY "I                                                     ;
           REPONSE        I"
@ 13, 0 SAY "I  FIN DE TRAITEMENT                                  ;
  --> F                   I"
@ 14, 0 SAY "I                                                     ;
                          I"
@ 15, 0 SAY "I                                                     ;
                          I"
```

```
@ 16, 0 SAY "I                                                    ;
                          I"
@ 17, 0 SAY "I                                                    ;
                          I"
@ 18, 0 SAY "I                                                    ;
                          I"
@ 19, 0 SAY "I                                                    ;
                          I"
@ 20, 0 SAY "+-------------------------------------------------------;
-------------------------+"
@ 12,72 GET REPONSE4
READ
* SI LA SELECTION EST DIFFERENTE DE "1", "2" OU "3" -> AFFICHAGE D'UN
* MESSAGE D'ERREUR.
IF !(REPONSE4)#"F" .AND. REPONSE4#"1" .AND. REPONSE4#"2" .AND. REPONSE4#"3"
@ 22,0 SAY "Aucun traitement ne correspond a votre selection"
ELSE
 IF !(REPONSE4)="F"
 RELEASE REPONSE4,CONTREPORT,IMPRIMANTE,PARAMETRE
 RETURN
 ELSE
 @ 22,0 SAY "Desirez-vous une sortie sur imprimante (O/N) ? " GET IMPRIMANTE
 READ
 ERASE

* SI LA SELECTION EST : "1"  --> IMPRESSION  DE TOUS LES  ENREGISTREMENTS
* POUR LESQUELS LA SOMME DE LA QUANTITE EN STOCK ET CELLE RESTANT A LIVRER
* EST INFERIEURE A LA QUANTITE MINIMUM.

  IF REPONSE4 = "1"
     IF !(IMPRIMANTE) = "O"
     REPORT FORM ETATMINI FOR QTE:STOCK + QTE:RESTE < MINI:STOCK TO PRINT
     ELSE
     REPORT FORM ETATMINI FOR QTE:STOCK + QTE:RESTE < MINI:STOCK
     ENDIF
  ELSE

* SI LA SELECTION EST : "2" -> SAISIE DU LIBELLE RECHERCHE, ET   IMPRESSION
* DE TOUS LES ENREGISTREMENTS DONT LA ZONE LIBELLE CONTIENT TOUTE OU PARTIE
* DU PARAMETRE RECHERCHE.
```

```
      IF REPONSE4 = "2"
      @ 4,0 SAY " "
      ACCEPT "Donnez le parametre recherche " TO PARAMETRE
      ERASE
      STORE !(PARAMETRE) TO PARAMETRE
         IF !(IMPRIMANTE) = "O"
         REPORT FORM SELESTOC FOR "&PARAMETRE" $ LIBELLE TO PRINT
         ELSE
         REPORT FORM SELESTOC FOR "&PARAMETRE" $ LIBELLE
         ENDIF
         ELSE

* SI LA SELECTION EST : "3" -> SAISIE DE LA PLAGE DE PRIX PUIS
* IMPRESSION DE TOUS LES ENREGISTREMENTS DONT LE PRIX UNITAIRE
* EST COMPRIS ENTRE CES DEUX EXTREMES.

          IF REPONSE4 = "3"
          STORE 0.00 TO PRIX1
          STORE 0.00 TO PRIX2
          @ 1,0 SAY "Donnez le prix minimum " GET PRIX1
          @ 2,0 SAY "Donnez le prix maximum " GET PRIX2
          READ
          ERASE
          STORE "FOR PRIX:UHT>=PRIX1 .AND. PRIX:UHT<=PRIX2" TO CONDITION
             IF !(IMPRIMANTE) = "O"
             REPORT FORM SELESTOC &CONDITION TO PRINT
             RELEASE CONDITION,PRIX1,PRIX2
             ELSE
             REPORT FORM SELESTOC &CONDITION
             RELEASE CONDITION,PRIX1,PRIX2
             ENDIF
          ENDIF 3
     ENDIF 2
  ENDIF 1
 ENDIF F
ENDIF
? "Validez"
SET CONSOLE OFF
             WAIT
             SET CONSOLE ON
             @ 22,0 SAY "                                          "
             ENDDO
```

FICHIER DE COMMANDE TRADA.CMD

TRADA.CMD

```
******************** TRAITEMENT DES DEMANDES D'ACHAT ********************
* Ce fichier de commande vous propose differentes options de traitement *
* au sein du fichier des demandes d'achat. Selon votre selection il *
* lancera le traitement correspondant. La reponse de defaut : "F" vous *
* permettra de revenir au menu principal : MENU2                       *
* Comme pour le fichier de commande : TRASTOCK, la meme presentation *
* d'ecran d'une demande d'achat sera utilisee pour les differentes op- *
* tions ( 1, 2, 3 ou 4 ), en precisant en haut et a droite de l'ecran *
* le mode en cours.                                                    *
************************************************************************

STORE '0' TO CONTDA
DO WHILE CONTDA = '0'
ERASE
STORE 'F' TO REPONSE3
@  1,27 SAY "TRAITEMENT DES DEMANDES D'ACHAT"
@  5, 0 SAY "+------.-----------------------------------------------------;
-----------------+"
@  6, 0 SAY "I                                                            ;
                 I"
@  7, 0 SAY "I                              SELECTIONNEZ VOTRE OPTION     ;
                 I"
@  8, 0 SAY "I                                                            ;
                 I"
@  9, 0 SAY "I                                                            ;
                 I"
@ 10, 0 SAY "I  AJOUT DE NOUVEAUX ENREGISTREMENTS -> 1                    ;
                 I"
@ 11, 0 SAY "I                                                            ;
                 I"
@ 12, 0 SAY "I  MODIFICATION D'ENREGISTREMENTS    -> 2                    ;
                 I"
@ 13, 0 SAY "I                                                            ;
                 I"
@ 14, 0 SAY "I  SAISIE NUMERO DE COMMANDE         -> 3                    ;
                 I"
```

```
@ 15, 0 SAY "I                                                    REPONSE    ;
                I"
@ 16, 0 SAY "I  VISUALISATION D'ENREGISTREMENTS    -> 4                      ;
                I"
@ 17, 0 SAY "I                                                               ;
                I"
@ 18, 0 SAY "I  SORTIE D'ETAT SUIVANT SELECTION    -> 5                      ;
                I"
@ 19, 0 SAY "I                                                               ;
                I"
@ 20, 0 SAY "I  RETOUR AU MENU INITIAL             -> F                      ;
                I"
@ 21, 0 SAY "+-------------------------------------------------------------;
----------------+"
@ 15,60 GET REPONSE3
READ
IF !(REPONSE3) = 'F'
RELEASE CONTDA,REPONSE3,MODEDA
RETURN
ELSE
 IF REPONSE3 = '1'
 STORE 'AJOUT' TO MODEDA
 DO DASCR
 DO AJDA
 USE
 ELSE
   IF REPONSE3 = '2'
   STORE 'MODIFICATION' TO MODEDA
   DO DASCR
   DO MODIFDA
   USE
   ELSE
     IF REPONSE3 = "3"
     STORE "SAISIE CDE" TO MODEDA
     DO DASCR
     DO CDEDA
     USE
     ELSE
```

```
       IF REPONSE3 = "4"
       STORE "VISUALISATION" TO MODEDA
       DO DASCR
       DO VISDA
       USE
       ELSE
         IF REPONSE3 = "5"
         DO REDA
         USE
         ELSE
           @ 22,0 SAY 'AUCUN TRAITEMENT NE CORRESPOND A VOTRE SELECTION, ;
VALIDEZ'
           SET CONSOLE OFF
           WAIT
           SET CONSOLE ON
         ENDIF 5
       ENDIF 4
     ENDIF 3
   ENDIF 2
 ENDIF 1
ENDIF F
ENDDO CONTDA
```

FICHIER DE COMMANDE DASCR.CMD

DASCR.CMD

```
ERASE
@  1, 0 SAY "                                                              +-----;
---------------+"
@  2, 0 SAY "                                                              I Mode;
:                I"
@  2,66 SAY MODEDA
@  3, 0 SAY "+-----------------------------------+----------------+-----;

---------------+"
```

```
@  4, 0 SAY "I                                        I                        ;
                  I"
@  5, 0 SAY "I No D.A.              DATE              I No CDE              DATE;
                  I"
@  6, 0 SAY "I                                        I                        ;
                  I"
@  7, 0 SAY "I Emetteur                               I                        ;
                  I"
@  8, 0 SAY "I                                        I Type                   ;
                  I"
@  9, 0 SAY "I Service                                I                        ;
                  I"
@ 10, 0 SAY "I                                        I                        ;
                  I"
@ 11, 0 SAY "+-------------------------------------+-----+-----------------;
---------------+"
@ 12, 0 SAY "I              FOURNISSEUR                  I           MACHINE;
                  I"
@ 13, 0 SAY "+-----------------------------------------+-----------------;
---------------+"
@ 14, 0 SAY "I                                          I                  ;
                  I"
@ 15, 0 SAY "+------------+--------------+-------------+-----------------;
---+-----------+"
@ 16, 0 SAY "I  Qte Cde   I   Reference   I          Designation            ;
   I Prix UHT   I"
@ 17, 0 SAY "+------------+--------------+-------------------------------;
---+-----------+"
@ 18, 0 SAY "I            I              I                                  ;
   I            I"
@ 19, 0 SAY "+------------+--------------+-------------------------------;
---+-----------+"
@ 20, 0 SAY "I Imputation                                                Delai;
 semaine          I"
@ 21, 0 SAY "+-----------------------------------------------------------------;
---------------+"
RETURN
```

FICHIER DE COMMANDE AJDA.CMD

AJDA.CMD

```
************ CREATION DE NOUVEAUX ENREGISTREMENTS **************
* CE FICHIER DE COMMANDE VOUS PERMETTRA DE CREER DE NOUVELLE(S) *
* DEMANDE(S) D'ACHAT - DANS LA MESURE  OU  ELLE(S) N'EXISTE(NT) *
* PAS - ET DE METTRE A JOUR  SIMULTANEMENT  L'ENREGISTREMENT DE *
* LA PIECE CONCERNEE DANS LE FICHIER STOCK.                     *
*****************************************************************

SET EXACT ON
* DEFINITION DE LA BASE PRIMAIRE
SELECT PRIMARY
USE STOCK INDEX PIECE
* DEFINITION DE LA BASE SECONDAIRE
SELECT SECONDARY
USE DA INDEX DA
STORE '0' TO CONTAJ
DO WHILE CONTAJ = '0'
SELECT SECONDARY
STORE '           ' TO MDA:NUM
STORE '              ' TO MPIECE
* EFFACEMENT DES DIFFERENTES ZONES DE SAISIE A L'ECRAN
@ 5,27 SAY '          '
@ 7,11 SAY '                              '
@ 8,45 SAY '                    '
@ 9,11 SAY '         '
@ 14,2 SAY '                                                  '
@ 14,46 SAY '                             '
@ 18,1 SAY '              '
@ 18,14 SAY '                '
@ 18,31 SAY '                                '
@ 18,67 SAY '              '
@ 20,13 SAY '                                '
@ 20,71 SAY '      '
@ 22,0 SAY '                                                                '
```

```
* SAISIE DU NUMERO DE LA DEMANDE D'ACHAT
@ 5,10 GET MDA:NUM
READ
* SI AUCUN NUMERO N'EST SAISIE -> RETOUR AU MENU TRAITEMENT DES D.A.
IF MDA:NUM = ' '
SET EXACT OFF
RELEASE MDA:NUM,MDA:DATE,MTYPE,MEMETTEUR,MSECTION,MFOURNI,MPIECE
RELEASE MQTE:CMD,MIMPUTATIO,MDELAI,MPRIX:UHT
RETURN
ELSE
* RECHERCHE L'EXISTENCE DE LA DEMANDE D'ACHAT SAISIE DANS LA BASE SECONDAIRE
* BASE SECONDAIRE = BASE DA
SELECT SECONDARY
FIND &MDA:NUM
        IF # > 0
        @ 22,0 SAY 'La D.A. '
        @ 22,9 SAY MDA:NUM
        @ 22,21 SAY ' est deja enregistree, VALIDEZ'
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
        ELSE
* SI LA DEMANDE D'ACHAT N'EXISTE PAS DEJA -> SAISIE DE LA REFERENCE
* DE LA PIECE A COMMANDER
        STORE '              ' TO MPIECE
        @ 18,14 GET MPIECE
        READ
* RECHERCHE L'EXISTENCE DE LA REFERENCE SAISIE DANS LA BASE PRIMAIRE
* BASE PRIMAIRE = BASE STOCK
        SELECT PRIMARY
        FIND &MPIECE
                IF # = 0
                @ 22,0 SAY "La piece "
                @ 22, 9 SAY MPIECE
                @ 22,21 SAY " n'a pas ete enregistree, enregistrez la, ;
S.V.P."
                SET CONSOLE OFF
                WAIT
                SET CONSOLE ON
                ELSE
```

```
* SI CETTE REFERENCE EXISTE -> SAISIE DES AUTRES PARAMETRES
                STORE '         ' TO MDA:DATE
                STORE '                ' TO MTYPE
                STORE '                              ' TO MEMETTEUR
                STORE '      ' TO MSECTION
                STORE '                                                  ' TO MFOURNI
                STORE 0 TO MQTE:CMD
                STORE '                              ' TO MIMPUTATIO
                STORE '00' TO MDELAI
                STORE PRIX:UHT TO MPRIX:UHT
                @ 5,27 GET MDA:DATE PICTURE "99/99/99"
                @ 7,11 GET MEMETTEUR
                @ 9,11 GET MSECTION
                @ 8,45 GET MTYPE
                @ 14,2 GET MFOURNI
                @ 14,46 SAY MACHINE
                @ 18,1 GET MQTE:CMD
                @ 18,31 SAY LIBELLE
                @ 18,67 GET MPRIX:UHT
                @ 20,13 GET MIMPUTATIO
                @ 20,72 GET MDELAI PICTURE "99"
                READ
* MISE A JOUR DE LA REFERENCE CONCERNEE DANS LA BASE PRIMAIRE
* BASE PRIMAIRE = BASE DU STOCK DES PIECES DETACHEES
                REPLACE DA:NUM WITH MDA:NUM
                REPLACE QTE:CMD WITH MQTE:CMD
                REPLACE QTE:RESTE WITH P.QTE:RESTE + MQTE:CMD
                REPLACE PRIX:UHT WITH MPRIX:UHT
* ENREGISTREMENT DE LA DEMANDE D'ACHAT DANS LA BASE SECONDAIRE
* BASE SECONDAIRE = BASE DES DEMANDES D'ACHAT
                SELECT SECONDARY
                APPEND BLANK
                REPLACE DA:NUM WITH MDA:NUM
                REPLACE DA:DATE WITH MDA:DATE
                REPLACE EMETTEUR WITH MEMETTEUR
                REPLACE SECTION WITH MSECTION
                REPLACE TYPE WITH MTYPE
                REPLACE FOURNI WITH MFOURNI
                REPLACE QTE:CMD WITH MQTE:CMD
                REPLACE PIECE WITH MPIECE
                REPLACE PRIX:UHT WITH MPRIX:UHT
```

```
                REPLACE IMPUTATION WITH MIMPUTATIO
                REPLACE DELAI WITH VAL(MDELAI)
                REPLACE CMD:DATE WITH "  /  /  "
                ENDIF
        ENDIF
ENDIF
ENDDO AJDA
```

FICHIER DE COMMANDE MODIFDA.CMD

MODIFDA.CMD

```
*************** MODIFICATION DE DEMANDES D'ACHAT **************

SET EXACT ON
* DEFINITION DE LA BASE PRIMAIRE
SELECT PRIMARY
USE STOCK INDEX PIECE
* DEFINITION DE LA BASE SECONDAIRE
SELECT SECONDARY
USE DA INDEX DA
STORE "NON" TO EFFACEMENT
STORE '0' TO CONTAJ
DO WHILE CONTAJ = '0'
SELECT SECONDARY
STORE '          ' TO MDA:NUM
* EFFACEMENT DES DIFFERENTES ZONES DE SAISIE A L'ECRAN
@ 5,27 SAY  '          '
@ 5,47 SAY "              "
@ 5,66 SAY "          "
@ 7,11 SAY  '                          '
@ 8,45 SAY  '                    '
@ 9,11 SAY  '         '
@ 14,2 SAY '                                              '
```

```
@ 14,46 SAY '                              '
@ 18,1 SAY  '             '
@ 18,14 SAY '               '
@ 18,31 SAY '                                    '
@ 18,67 SAY '             '
@ 20,13 SAY '                                '
@ 20,71 SAY '      '
@ 22,0 SAY '                                                                  ;
           '
@ 23,0 SAY '                                      '
* SAISIE DU NUMERO DE LA DEMANDE D'ACHAT
@ 5,10 GET MDA:NUM
READ
* SI AUCUN NUMERO N'EST SAISI -> RETOUR AU MENU TRAITEMENT DES D.A.
IF MDA:NUM = ' '
SET EXACT OFF
IF EFFACEMENT="OUI"
@ 22,0 SAY "Prise en compte des modifications en cours."
@ 23,0 SAY "Veuillez patienter quelques instants ....."
PACK
ENDIF
RELEASE MDA:NUM,MDA:DATE,MTYPE,MEMETTEUR,MSECTION,MFOURNI,MPIECE
RELEASE MQTE:CMD,MIMPUTATIO,MDELAI,MPRIX:UHT,EFFACEMENT
RELEASE MCMD:NUM,MCMD:DATE,REPONSE
RETURN
ELSE
* RECHERCHE L'EXISTENCE DE LA DEMANDE D'ACHAT SAISIE DANS LA BASE SECONDAIRE
SELECT SECONDARY
FIND &MDA:NUM
        IF # = 0 .OR. *
        @ 22,0 SAY 'La D.A. '
        @ 22,9 SAY MDA:NUM
        @ 22,21 SAY 'ne figure pas dans le fichier, Validez'
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
        ELSE
```

```
* RECHERCHE L'EXISTENCE DE LA REFERENCE SAISIE DANS LA BASE PRIMAIRE
        SELECT PRIMARY
        STORE S.PIECE TO MPIECE
        FIND &MPIECE
                IF # = 0
                @ 18,14 SAY MPIECE
                @ 22,0 SAY "La piece "
                @ 22, 9 SAY MPIECE
                @ 22,21 SAY " ne figure plus dans le fichier"
                @ 23,0 SAY "Modification impossible"
                SET CONSOLE OFF
                WAIT
                SET CONSOLE ON
                ELSE
                @ 14,46 SAY P.MACHINE
                @ 18,31 SAY P.LIBELLE
                SELECT SECONDARY
                STORE S.DA:DATE TO MDA:DATE
                STORE S.CMD:NUM TO MCMD:NUM
                STORE S.CMD:DATE TO MCMD:DATE
                STORE S.TYPE TO MTYPE
                STORE S.EMETTEUR TO MEMETTEUR
                STORE S.SECTION TO MSECTION
                STORE S.FOURNI TO MFOURNI
                STORE S.QTE:CMD TO MQTE:CMD
                STORE S.IMPUTATION TO MIMPUTATIO
                STORE STR(S.DELAI,2) TO MDELAI
                STORE S.PRIX:UHT TO MPRIX:UHT
                @ 5,27 GET MDA:DATE PICTURE "99/99/99"
                @ 5,47 GET MCMD:NUM
                @ 5,66 GET MCMD:DATE PICTURE "99/99/99"
                @ 7,11 GET MEMETTEUR
                @ 9,11 GET MSECTION
                @ 8,45 GET MTYPE
                @ 14,2 GET MFOURNI
                @ 18,1 GET MQTE:CMD
                @ 18,14 SAY MPIECE
                @ 18,67 GET MPRIX:UHT
                @ 20,13 GET MIMPUTATIO
                @ 20,72 GET MDELAI PICTURE "99"
                READ
```

```
* SI AUCUNE MODIFICATION N'A ETE EFFECTUEE -> EFFACEMENT OUI/NON ?

                    IF MDA:DATE=S.DA:DATE .AND. MCMD:NUM=S.CMD:NUM .AND. ;
MCMD:DATE=S.CMD:DATE .AND. MEMETTEUR=S.EMETTEUR .AND. MSECTION=S.SECTION ;
 .AND. MTYPE=S.TYPE .AND. MFOURNI=S.FOURNI .AND. MQTE:CMD=S.QTE:CMD ;
 .AND. MPRIX:UHT=S.PRIX:UHT .AND. MIMPUTATIO=S.IMPUTATION .AND. ;
VAL(MDELAI)=DELAI
                    STORE "N" TO REPONSE
                    @ 22,0 SAY "Voulez-vous l'effacer ? (O/N) " GET REPONSE
                    READ
                        IF REPONSE = "O"
                                IF QTE:RESTE-QTE:CMD < 0
                                @ 22,0 SAY "Toute ou partie de cette ;
commande est livree"
                                @ 23,0 SAY "Effacement impossible, Validez ."
                                SET CONSOLE OFF
                                WAIT
                                SET CONSOLE ON
                                @ 22,0 SAY "                                                    "
                                @ 23,0 SAY "                                                    "
                                ELSE
                                SELECT SECONDARY
                                STORE "OUI" TO EFFACEMENT
                                DELETE
                                SELECT PRIMARY
                                REPLACE P.DA:NUM WITH "          "
                                REPLACE QTE:RESTE WITH QTE:RESTE - QTE:CMD
                                ENDIF
                        ENDIF
                    ELSE
                     IF QTE:RESTE-QTE:CMD < 0
                     @ 22,0 SAY "Tout ou partie de cette commande est livree"
                     @ 23,0 SAY "Modification impossible, Validez ."
                     SET CONSOLE OFF
                     WAIT
                     SET CONSOLE ON
                     @ 22,0 SAY "                                                    "
                     @ 23,0 SAY "                                    "
                     ELSE
```

```
* MISE A JOUR DE LA REFERENCE CONCERNEE DANS LA BASE PRIMAIRE
* BASE PRIMAIRE = BASE DU STOCK DES PIECES DETACHEES
                    SELECT PRIMARY
                    REPLACE QTE:RESTE WITH P.QTE:RESTE + (MQTE:CMD - S.QTE:CMD)
                    REPLACE QTE:CMD WITH MQTE:CMD
                    REPLACE PRIX:UHT WITH MPRIX:UHT
* ENREGISTREMENT DE LA DEMANDE D'ACHAT DANS LA BASE SECONDAIRE
* BASE SECONDAIRE = BASE DES DEMANDES D'ACHAT
                    SELECT SECONDARY
                    REPLACE DA:DATE WITH MDA:DATE
                    REPLACE CMD:NUM WITH MCMD:NUM
                    REPLACE CMD:DATE WITH MCMD:DATE
                    REPLACE EMETTEUR WITH MEMETTEUR
                    REPLACE SECTION WITH MSECTION
                    REPLACE TYPE WITH MTYPE
                    REPLACE FOURNI WITH MFOURNI
                    REPLACE QTE:CMD WITH MQTE:CMD
                    REPLACE PRIX:UHT WITH MPRIX:UHT
                    REPLACE IMPUTATION WITH MIMPUTATIO
                    REPLACE DELAI WITH VAL(MDELAI)
                     ENDIF
                    ENDIF
                ENDIF
        ENDIF
ENDIF
ENDDO MODIFDA
```

FICHIER DE COMMANDE CDEDA.CMD

CDEDA.CMD

```
****************** SAISIE DE NUMERO(S) DE COMMANDE ****************
* CE FICHIER VOUS PERMETTRA DE SAISIR LE NUMERO DE COMMANDE RELATIF *
* A UNE DEMANDE D'ACHAT                                             *
*********************************************************************

SET EXACT ON
SELECT PRIMARY
USE STOCK INDEX PIECE
```

```
SELECT SECONDARY
USE DA INDEX DA
STORE '0' TO CONTCDE
DO WHILE CONTCDE = '0'
SELECT SECONDARY
@ 5,27 SAY '           '
@ 7,11 SAY '                              '
@ 9,11 SAY '        '
@ 5,47 SAY '                '
@ 5,66 SAY '           '
@ 8,45 SAY '                    '
@ 14,2 SAY '                                          '
@ 14,46 SAY '                          '
@ 18,1 SAY '              '
@ 18,14 SAY '                 '
@ 18,31 SAY '                                '
@ 18,67 SAY '              '
@ 20,13 SAY '                                  '
@ 20,71 SAY '     '
STORE '           ' TO MDA:NUM
@ 5,10 GET MDA:NUM
READ
IF MDA:NUM = ' '
SET EXACT OFF
RELEASE MDA:NUM,MCMD:NUM,MCMD:DATE,CONTCDE
RETURN
ELSE
FIND &MDA:NUM
        IF # = 0
        @ 22,0 SAY 'La D.A. '
        @ 22,9 SAY MDA:NUM
        @ 22,21 SAY " n'a pas ete enregistree, VALIDEZ"
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
        ELSE
        STORE "             " TO MCMD:NUM
        STORE "         " TO MCMD:DATE
```

```
        @ 5,27 SAY DA:DATE
        @ 7,11 SAY EMETTEUR
        @ 9,11 SAY SECTION
        @ 8,45 SAY TYPE
        @ 14,2 SAY FOURNI
        @ 14,46 SAY MACHINE
        @ 18,1 SAY QTE:CMD
        @ 18,16 SAY PIECE
        @ 18,31 SAY LIBELLE
        @ 18,69 SAY PRIX:UHT
        @ 20,13 SAY IMPUTATION
        @ 20,71 SAY DELAI
                IF CMD:NUM # " "
                @ 5,47 SAY CMD:NUM
                @ 5,66 SAY CMD:DATE
                @ 22,0 SAY "Il existe deja un numero de commande pour ;
la demande d'achat "
                @ 22,62 SAY DA:NUM
                SET CONSOLE OFF
                WAIT
                SET CONSOLE ON
                @ 22,0 SAY '                                                ;

                ELSE
                SELECT PRIMARY
                STORE S.PIECE TO RECHERCHE
                FIND &RECHERCHE
                        IF # = 0
                        @ 22,0 SAY "La piece"
                        @ 22,9 SAY S.PIECE
                        @ 22,20 SAY "ne figure plus dans le fichier, ;
enregistrez la,"
                        @ 22,68 SAY "Validez"
                        SET CONSOLE OFF
                        WAIT
                        SET CONSOLE ON
                        @ 22,0 SAY "                                        ;
                                   "
```

```
                        ELSE
                        @ 5,47 GET MCMD:NUM
                        @ 5,66 GET MCMD:DATE PICTURE "99/99/99"
                        READ
                        SELECT SECONDARY
                        REPLACE CMD:NUM WITH MCMD:NUM
                        REPLACE CMD:DATE WITH MCMD:DATE
                        ENDIF
                ENDIF
        ENDIF
ENDIF
ENDDO SAISIE NUMERO DE COMMANDE
```

FICHIER DE COMMANDE VISDA.CMD

VISDA.CMD

```
************* VISUALISATION D'UNE DEMANDE D'ACHAT *************

SET EXACT ON
STORE " " TO VALIDEZ
@ 0,0 SAY "FIN = VALIDEZ"
SELECT PRIMARY
USE STOCK INDEX PIECE
SELECT SECONDARY
USE DA INDEX DA
STORE "          " TO MDA:NUM
STORE "0" TO CONTVISU
DO WHILE CONTVISU = "0"
@ 5,10 GET MDA:NUM
READ
        IF MDA:NUM = " " .OR. MDA:NUM = VALIDEZ
        RELEASE VALIDEZ,CONTVISU,MDA:NUM,MPIECE
        SET EXACT OFF
        RETURN
        ELSE
        STORE MDA:NUM TO VALIDEZ
        SELECT SECONDARY
```

```
        FIND &MDA:NUM
                IF # = 0
                @ 5,27 SAY "          "
                @ 7,11 SAY "                                  "
                @ 9,11 SAY "         "
                @ 5,47 SAY "               "
                @ 5,66 SAY "            "
                @ 8,45 SAY "                    "
                @ 14,2 SAY "                                                  "
                @ 14,46 SAY "                                 "
                @ 18,1 SAY "         "
                @ 18,14 SAY "                 "
                @ 18,31 SAY "                                         "
                @ 18,67 SAY "              "
                @ 20,13 SAY "                                 "
                @ 20,71 SAY "     "
                @ 22,0 SAY "CETTE DEMANDE D'ACHAT NE FIGURE PAS DANS LE ;
FICHIER, Validez"
                SET CONSOLE OFF
                WAIT
                SET CONSOLE ON
                @ 22,0 SAY "                                                  ;
             "
                STORE "           " TO MDA:NUM
                ELSE
                SELECT PRIMARY
                STORE S.PIECE TO MPIECE
                FIND &MPIECE
                    IF # = 0
                    @ 14,46 SAY "                                 "
                    @ 18,31 SAY "                                         "
                    @ 22,0  SAY "La piece"
                    @ 22,9 SAY MPIECE
                    @ 22,22 SAY "ne figure plus dans le fichier stock, ;
Validez"
                    SET CONSOLE OFF
                    WAIT
                    SET CONSOLE ON
                    @ 22,0 SAY "                                                  ;
                         "
```

```
                    ELSE
                    @ 14,46 SAY MACHINE
                    @ 18,31 SAY LIBELLE
                    ENDIF
                @ 5,27 SAY S.DA:DATE
                @ 7,11 SAY S.EMETTEUR
                @ 9,11 SAY S.SECTION
                @ 5,47 SAY S.CMD:NUM
                @ 5,66 SAY S.CMD:DATE
                @ 8,45 SAY S.TYPE
                @ 14,2 SAY S.FOURNI
                @ 18,1 SAY S.QTE:CMD
                @ 18,15 SAY S.PIECE
                @ 18,69 SAY S.PRIX:UHT
                @ 20,13 SAY S.IMPUTATION
                @ 20,71 SAY S.DELAI
                ENDIF
        ENDIF
ENDDO VISUALISATION DEMANDE D'ACHAT
```

FICHIER DE COMMANDE REDA.CMD

REDA.CMD

```
******************** GENERATION D'ETAT(S) **********************

USE DA INDEX DA
STORE "O" TO CONTREPORT
DO WHILE CONTREPORT = "O"
STORE "N" TO IMPRIMANTE
STORE "F" TO REPONSE5
ERASE
@  3, 0 SAY "+-------------------------------------------------;
-------------------------+"
@  4, 0 SAY "I                                                  ;
                         I"
@  5, 0 SAY "I                    GENERATION D'ETAT SUIVANT VOTRE ;
CHOIX                      I"
```

```
@  6, 0 SAY "I                                                          ;
                              I"
@  7, 0 SAY "I                                                          ;
                              I"
@  8, 0 SAY "I   LISTE DES COMMANDES AVEC UN DELAI DE LIVRAISON NON;
 RESPECTE --> 1              I"
@  9, 0 SAY "I                                                          ;
                              I"
@ 10, 0 SAY "I   LISTE DES DEMANDES D'ACHAT SELON LE NOM DE L'EMET;
TEUR       --> 2              I"
@ 11, 0 SAY "I                                                          ;
                              I"
@ 12, 0 SAY "I   LISTES DES DEMANDES D'ACHAT DANS UNE PERIODE DONN;
EE         --> 3              I"
@ 13, 0 SAY "I                                                          ;
                              I"
@ 14, 0 SAY "I   FIN DE TRAITEMENT                                      ;
           --> F              I"
@ 15, 0 SAY "I                                                          ;
                              I"
@ 16, 0 SAY "I                                                          ;
                              I"
@ 17, 0 SAY "I                                                          ;
                              I"
@ 18, 0 SAY "I                                                          ;
                              I"
@ 19, 0 SAY "+-------------------------------------------------------;
-------------------------+"
@ 20,59 SAY "I REPONSE"
@ 20,69 GET REPONSE5
@ 20,79 SAY "I"
@ 21,59 SAY "+--------------------+"
READ
IF !(REPONSE5)#"F" .AND. REPONSE5#"1" .AND. REPONSE5#"2" .AND. REPONSE5#"3"
@ 22,0 SAY "AUCUN TRAITEMENT NE CORRESPOND A VOTRE SELECTION"
ELSE
 IF !(REPONSE5) = "F"
 RELEASE REPONSE5,CONTREPORT,IMPRIMANTE
 RETURN
 ELSE
```

```
@ 22,0 SAY "Desirez-vous une sortie sur imprimante (O/N) ? " GET IMPRIMANTE
READ
  IF REPONSE5 = "1"
  ERASE
       IF !(IMPRIMANTE) = "O"
       REPORT FORM SELECTDA FOR DELAI < SEMAINE .AND. $(DATE,7,2)=;
       $(DA:DATE,7,2) TO PRINT
       ELSE
       REPORT FORM SELECTDA FOR DELAI < SEMAINE .AND. $(DATE,7,2)=;
       $(DA:DATE,7,2)
       ENDIF
  ELSE
       IF REPONSE5 = "2"
       ERASE
       @ 4,0 SAY " "
       ACCEPT "Donnez le nom de l'emetteur " TO PARAMETRE
       ERASE
              IF !(IMPRIMANTE) = "O"
              REPORT FORM EMETDA FOR PARAMETRE $ EMETTEUR TO PRINT
              RELEASE PARAMETRE
              ELSE
              REPORT FORM EMETDA FOR PARAMETRE $ EMETTEUR
              RELEASE PARAMETRE
              ENDIF
ELSE
                 IF REPONSE5 = "3"
                 ERASE
                 STORE "      " TO PERIODE
                 @ 1,0 SAY "Donnez la periode 'MM/AA' " GET PERIODE PICTURE "99/99"
                 READ
                 STORE "FOR $(DA:DATE,4,5) = PERIODE" TO CONDITION
                      ERASE
                      IF !(IMPRIMANTE) = "O"
                      REPORT FORM DADATE &CONDITION TO PRINT
                      RELEASE CONDITION,DATE1,DATE2
                      ELSE
                      REPORT FORM DADATE &CONDITION
                      RELEASE CONDITION,PERIODE
                      ENDIF
                 ELSE
                 ENDIF 3
```

```
        ENDIF 2
   ENDIF 1
 ENDIF F
ENDIF
? "Validez"
SET CONSOLE OFF
WAIT
SET CONSOLE ON
@ 22,0 SAY "                                                    "
ENDDO GENERATION D'ETAT(S)
```

Troisième partie

GUIDE DE PROGRAMMATION

TABLE DES MATIERES

GUIDE DE PROGRAMMATION

CHAPITRE I

CHAPITRE II

ANNEXES

ZIP

CHAPITRE I

COMMENT UTILISER dBASE

Afin d'exécuter le programme dBASE, placez la disquette de distribution dBASE (ou de préférence une copie de cette disquette) dans n'importe quelle unité de disque utilisable. Si B est l'unité de disque, tapez B : suivi d'un retour de chariot puis tapez la ligne suivante :

```
A>DBASE
```

Le programme sera mémorisé et commencera à s'exécuter en demandant la date :

```
** ENTREZ LA DATE DU JOUR FORME JJ/MM/AA (SINON RETURN)
```

Cette date sera prise en compte dans le traitement de vos bases de données et sera aussi imprimée dans les titres de rapports générés ce jour-là. La date est vérifiée avec le calendrier.

ATTENTION : le contrôle avec le calendrier n'est pas correct pour le 29 février des années 1900 et 2100. Une barre ou tout autre caractère spécial (à l'exception du point) peut être utilisé pour délimiter les nombres.
Exemples de dates correctes :

```
1,1,84
02 02 85
3/17/86
```

Il vous sera ensuite visualisé un message :

```
***  dBASE II   VER 2.4***
```

Le point figurant sur la deuxième ligne est un message de dBASE II; il vous signale que dBASE est prêt à accepter vos commandes. Les commandes de dBASE II sont généralement des phrases impératives : un verbe suivi probablement de phrases qui donnent des instructions concernant l'exécution du programme. dBASE examine la totalité de chaque ligne avant d'en exécuter la commande. Si dBASE trouve une erreur dans la commande, il le signalera à l'utilisateur avec des messages d'erreur à la console. En général, l'utilisateur peut corriger uniquement la partie erronée plutôt que de réentrer la totalité de la commande. Lorsque dBASE trouve une erreur qu'il ne peut décrire clairement, il suppose que c'est une erreur de syntaxe et visualise la ligne erronée avec un point d'interrogation au début de la phrase, cause de la confusion.

Quelques exemples d'erreurs :

```
. DISPRAY MEMORY
*** CETTE COMMANDE NE PEUT ETRE UTILISEE ***
DISPRAY MEMORY
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ? Y
REMPLACER : PR
PAR       : PL
DISPLAY MEMORY
*** DESIREZ-VOUS D'AUTRES CORRECTIONS (Y/N) ? (cr)

. STORE (2+2 TO X
*** ERREUR DE SYNTAXE ***
      ?
STORE (2+2 TO X
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ? Y
REMPLACER : +2
PAR       : +2)
STORE (2+2) TO N
*** DESIREZ-VOUS D'AUTRES CORRECTIONS (Y/N) ? N
  4

. SUM TO X
*** INEXISTENCE DE L'EXPRESSION A TOTALISER :
SUM TO X
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ? N
```

Le programme peut aussi être exécuté de la façon suivante :

DBASE <nom-de-fichier>

Cette commande introduit dBASE en mémoire, lit le fichier de commande <nom-de-fichier> et commence immédiatement l'exécution de ce dernier fichier. Cette forme est particulièrement utile lorsque vous utilisez dBASE dans un fichier SUBMIT ou lorsque vous utilisez l'option enchaînement de la commande dBASE, QUIT.

Les caractères de contrôle

ctrl-P	active ou désactive l'impression (voir commande SET PRINT).
ctrl-U	efface la ligne courante.
ctrl-X	efface la ligne courante (sauf si vous êtes en mode édition).
Del	efface le dernier caractère entré.
ESC	Permet d'arrêter l'exécution de longues commandes. Exemples : DISPLAY, COUNT, DELETE, INPUT, LIST, LOCATE, RECALL, REPLACE, SKIP et SUM. ESC permet aussi de vous dégager des commandes ACCEPT, INPUT, REPORT (dialogue) et WAIT. Dans tous les cas, ESC redonne le contrôle au moniteur et visualise un point comme message. Avant d'exécuter un fichier de commande, dBASE vérifie l'existence d'un caractère ESC. NOTE : Cette possibilité de sortie peut être mise hors fonction par la commande SET ESCAPE OFF.

LES FICHIERS dBASE

Un fichier est une collection d'informations stockée dans un organe périphérique contenant les données de l'utilisateur. Vous pouvez corriger et stocker des informations dans un fichier. Les fichiers sont regroupés en six types, chacun d'eux concerne soit une opération particulière de dBASE II soit une création de ce dernier.

Tous les fichiers dBASE sont des fichiers standard CP/M avec un nom de fichier de huit caractères et une extension <type de fichier> de trois caractères. Vous trouverez listés ci-dessous les différents types de fichiers utilisés par dBASE. Lorsqu'une commande fait appel à un fichier, l'extension n'est pas spécifiée car dBASE connaît le type de fichier relatif à cette commande. Par exemple, si un fichier a déjà une extension .DBF, il n'est pas utile de spécifier cette extension dans vos commandes lors de la manipulation de ce fichier.

FICHIERS DE DONNEES	.DBF
FICHIERS MEMOIRE	.MEM
FICHIERS DE COMMANDE	.CMD
FICHIERS DE RAPPORT	.FRM
FICHIERS DE SORTIE DE TEXTE	.TXT
FICHIERS D'INDEX	.NDX
FICHIERS DE FORMAT	.FMT

Fichiers de données (.DBF)

dBASE est un logiciel de base de données. Ses fichiers de base de données peuvent contenir la structure d'un enregistrement et de 0 à 65 535 enregistrements. La structure de l'enregistrement est essentiellement un plan de format d'enregistrement de données. La structure peut contenir au maximum 32 entrées différentes. Chaque entrée se réfère à un champ de données de l'enregistrement. La structure contient les données suivantes :

- Le nom du champ de données.
- Le type de données à l'intérieur des champs.
- La taille des champs de données.
- La position de la donnée à l'intérieur des enregistrements.

Nom des champs de données
Le nom peut avoir au maximum 10 caractères. Dans toute opération de dBASE, les champs de données seront référencés par ce nom. De par leur nature, les champs de noms sont de type alphanumérique (plus les deux-points). Cependant, les champs doivent commencer avec une lettre et les deux-points doivent être insérés dans le nom. Voici quelques exemples.

Exemples de quelques noms de champs de données :

A	
A123456789	
ABC :DEF	
A :B :C :D :E :	
ABCD :	nom de champ incorrect, les deux-points ne sont pas insérés dans le nom
ABC,DEF	nom de champ incorrect, la virgule n'est pas autorisée

Type de données
Pour spécifier le contenu des champs de données, dBASE permet trois types : des chaînes de caractères ('ABCDE'), des quantités numériques (2 ou 5*18), ou des valeurs logiques (vraie/faux).

Dimension de champ
C'est le nombre de caractères (octets) nécessaire pour contenir les données qui seront placées à l'intérieur de ce champ. Les champs de chaînes de caractères et les champs numériques peuvent aller de 1 à 254 caractères. Le compte du champ numérique doit inclure la virgule décimale. Les champs logiques ont toujours une longueur de un caractère. Pour les champs numériques, le nombre de caractères à droite de la virgule décimale doit également être contenu dans la structure.

Après avoir défini la structure de son fichier, l'utilisateur peut y entrer des données dans autant d'enregistrements qu'il le désire. Normalement, l'utilisateur ne peut accéder qu'à un seul fichier de données structuré (USE fichier ou fichier en USE). Il existe cependant un moyen d'utiliser deux bases de données en même temps. Voir les commandes SELECT et JOIN.

Fichiers memoires — .MEM

Ce sont des fichiers de variables mémoire (semblables aux variables d'enregistrement) contenant un maximum de 64 variables.

Les valeurs des variables mémoire sont indépendantes de la base de données en USE. La position de l'enregistrement dans le fichier en USE n'a aucun rapport avec les variables mémoire. Les variables mémoire sont utilisées pour garder les constantes, les résultats des calculs, et les chaînes de substitution symboliques. Les règles pour nommer, taper et donner une dimension aux variables mémoire sont identiques aux règles de champs de variables décrites plus haut.

La commande SAVE écrira toutes les variables mémoire courantes dans un fichier mémoire. La commande RESTORE lira le fichier mémoire et mettra des valeurs dans les variables mémoire.

Fichiers de commandes — .CMD

Un fichier de commandes contient une suite de commandes dBASE. L'utilisateur pourra, par une fréquente sauvegarde de ces suites de commandes, manipuler facilement ses fichiers de base de données.

Les fichiers de commandes peuvent être créés et modifiés par les éditeurs et/ou les traitements de texte bien que nous pouvons, sous dBASE, par la commande MODIFY COMMAND, créer et modifier les fichiers de commandes. Les fichiers de commandes s'exécutent par la commande DO. Ces fichiers contiennent toutes les commandes dBASE, cependant nous devons être prudents puisque certaines commandes, telles que CREATE, INSERT, APPEND (entrées du clavier), exigent que l'utilisateur entre ses données au-delà du contenu du fichier de commandes.

Les fichiers de commandes peuvent être imbriqués : ils peuvent contenir des commandes DO qui seront alors exécutées. Pour la pratique de cette imbrication, la prudence est à nouveau recommandée car dBASE permet au maximum l'ouverture de 16 fichiers simultanément. Par conséquent, si vous avez déjà un fichier en USE, nous pouvons imbriquer seulement 15 fichiers de commandes. Certaines commandes utilisent aussi des fichiers de travail (SORT utilise deux fichiers supplémentaires; REPORT, INSERT, COPY, SAVE, RESTORE et PACK utilisent un fichier supplémentaire). Exemple, si une commande SORT figure dans un fichier de commandes imbriqué, il ne nous reste plus que 13 niveaux de fichiers de commandes disponibles (un fichier en USE, 2 fichiers de travail pour la commande SORT et 13 fichiers de commande = 16). Chaque fois qu'un fichier de commandes émet une commande RETURN, ou lorsque la fin d'un fichier est atteinte dans un fichier de commandes, ce dernier est fermé et ses ressources sont disponibles pour d'autres programmes.

Fichiers de rapport (.FRM)

La commande REPORT génère soit un fichier de format, soit utilise un fichier de format existant. Le fichier de format contient des instructions pour le générateur de rapport : titres, en-têtes, total et contenu des colonnes. dBASE construit ces fichiers de format en dialoguant avec l'utilisateur. Ces fichiers peuvent être modifiés par des éditeurs ou par des traitements de texte. Toutefois, il est plus facile de définir un nouveau rapport de façon conversationnelle.

Fichiers de sortie de texte (.TXT)

Les fichiers de sortie de texte sont créés lorsque les commandes « SET ALTERNATE TO <nom-de-fichier> » et « SET ALTERNATE ON » ont été spécifiées. Pour plus de détails, veuillez vous référer à la commande SET. Lorsque les options SDF (System Data Format) ou DELIMITED sont utilisées, les commandes COPY et APPEND supposeront qu'il existe un fichier de texte .TXT.

Fichiers d'index (.NDX)

Les fichiers index sont générés par la commande INDEX de dBASE. Ils contiennent les clés et les pointeurs des enregistrements d'un fichier de base de données. Indexer un fichier est une technique dBASE : il permet une localisation rapide des données à l'intérieur d'une grande base de données. Pour plus d'informations, veuillez voir la commande INDEX.

Fichiers de format (.FMT)

Un fichier de format contient seulement des phrases avec « @ » et des commentaires précédés d'« * ». Son identification se fait par la commande SET FORMAT TO <nom-de-fichier> et est activée par les commandes READ. Tout comme les fichiers de commandes, les fichiers de format peuvent être créés et modifiés par n'importe quel traitement de texte ou par la commande MODIFY COMMAND. Les fichiers de format ne sont cependant pas nécessaires. Les phrases avec « @ » et « * », si besoin est, sont généralement incorporées dans le fichier de commandes.

EXPRESSIONS

Sous dBASE, une expression est un ensemble d'opérandes simples et d'opérateurs pouvant être évalués pour former une nouvelle valeur simple. Exemple, « 2 + 2 » est une expression qui peut être évaluée pour la valeur 4. Les expressions ne sont pas nécessairement de nature numérique. L'expression ABC + DEF peut être évaluée pour la valeur ABCDEF (concaténation de chaînes de caractères), ou l'expression 1 > 2 peut être évaluée pour la valeur logique (booléenne) « .F. » (fausse).

Les expressions dBASE peuvent contenir les éléments suivants :

- Variables de champ de base de données
- Variables mémoire
- Constantes à l'intérieur des commandes (littérales)
- Fonctions
- Opérations

Variables
Une variable sous dBASE signifie un champ de données dont la valeur peut changer. Dans un fichier dBASE, les champs d'un enregistrement courant sont des variables. Leurs contenus peuvent être changés par le déplacement du pointeur de fichier ou par l'édition de l'enregistrement courant. Les variables sont créées et changées par les

commandes STORE, RESTORE, COUNT, SUM, WAIT, ACCEPT ou INPUT. Ces variables sont appelées variables mémoires.

Voici trois types de variables :

- Chaînes de caractères
- Quantités numériques
- Logiques

Constantes
Une constante (ou littéral) est une donnée dont la valeur est fixe ou définie par elle-même. Exemple : '1', 'abc', et .T. sont des constantes qui ont une valeur fixe, peu importe la position de la base de données ou des commandes de variable mémoire. Ce sont des littéraux qui représentent ce qu'ils sont (contrairement aux variables qui sont des noms représentant une valeur). Les valeurs représentées par les constantes sont respectivement : numériques, chaînes de caractères (contenant les lettres « a », « b » et « c ») ou logique (booléenne) valeur VRAIE (« .T. »).

Les constantes chaînes de caractères doivent être entourées d'apostrophes ('), de guillemets («) ou de crochets ([ ]). Si une chaîne de caractères contient l'un de ces délimiteurs, cette chaîne doit être entourée par d'autres délimiteurs. Par exemple, les chaînes 'abc[def] ghi' et [abc'def ' ghi] sont des chaînes de caractères correctes alors que 'abc'def 'ghi est une chaîne incorrecte.

Les constantes logiques (vraie/fausse) sont représentées par « T », « t », « Y », ou « y » pour les valeurs vraies (vraies ou affirmatives), et « F », « f », « N » ou « n » pour les valeurs fausses (fausses ou négatives).

Fonctions

Les fonctions sont des opérations à but spécial pouvant être utilisées dans les expressions pour exécuter des tâches qui sont difficiles ou impossibles avec l'utilisation des expressions simples. Les types de fonctions sous dBASE sont : numériques, caractères et logiques. Le type de la fonction est basé sur le type de valeur que cette fonction va générer.

Fonction nombre entier

INT(<expression numérique>)

Cette fonction évalue une expression numérique et, si nécessaire, ignore la partie fractionnaire pour donner une valeur entière. La valeur de la fonction INT est une valeur tronquée de l'expression numérique. Exemples :

```
. ? INT(123.456)
 123
. STORE 123.456 TO X
 123.456
. ? INT(X)
 123
```

Fonction numéro d'enregistrement

#

La valeur de la fonction numéro d'enregistrement est un nombre entier correspondant au numéro de l'enregistrement courant. Exemples :

```
. ? #
  4   (en supposant que la base de données est en USE et positionnée à
      l'enregistrement numéro 4)
. SKIP
. ? #
  5
```

Fonction de chaîne

STR(< expression numérique > , < dimension > ,[< décimale >])

Cette fonction évalue une expression numérique pour fournir une chaîne de caractères. La valeur de la fonction STR est une chaîne de caractères. Si les décimales sont spécifiées, ce sera le nombre de chiffres à la droite de la virgule décimale. Tous les spécifiants peuvent être des littéraux, des variables, ou des expressions.

ATTENTION : Lorsque cette fonction est utilisée pour générer une clé d'index, les spécifieurs DOIVENT être des littéraux. Exemple :

```
. ? STR(123.456,9,3)
  123.456
```

Fonction de sous-chaîne

$(<expression>,<début>,<dimension>)

Cette fonction forme une chaîne de caractères à partir d'une zone spécifique d'une autre chaîne. La valeur de la fonction de sous-chaîne est la chaîne de caractères de dimension <dimension> contenant les caractères de <expression> à partir du caractère <début>. <début> et <dimension> peuvent être littéraux, variables, ou expressions.

Si <début> plus <dimension> est supérieur à la longueur de <expression> + 1, l'<expression> est trop courte et la chaîne formée ne comprendra que les caractères disponibles. Voir exemple suivant.

ATTENTION : Lorsque la fonction est utilisée pour générer une clé d'index, les spécifieurs doivent être littéraux. Exemples :

```
. ? $('abcdefghi',3,3)
  cde
. store 3 to m
  3
. store 3 to n
  3
. ? $('abcdefghi',m,n)
  cde
. ? $('abcdefghi',6,7)
  fghi
```

Fonction de chaîne numérique

VAL(<chaîne de caractères>)

Cette fonction permet d'effectuer la conversion d'une chaîne de caractères numériques (chiffre 0 à 9, signe + ou —, séparateur décimal) en la valeur numérique entière correspondante. Si le premier caractère de la chaîne est différent de +, — ou chiffre de 0 à 9, la valeur numérique correspondante est nulle.

Il existe un autre moyen de convertir les nombres en chiffres par l'utilisation de « & » devant le nom d'une variable de chaîne. Lors de la substitution, le signe « & » convertira la chaîne en chiffre (y compris les décimales). Exemples :

```
. ? VAL('123')
  123
. ? VAL('123xxx')
  123
. ? VAL('123.456')
  123
. STORE '123.456' TO NUM
123.456
. ? 14 + &NUM
  137.456
. ? VAL("A123")
. 0
```

Fonction de conversion

RANK (<chaîne de caractères>)

Cette fonction renvoie la valeur numérique correspondant au premier caractère de la <chaîne de caractères>. Elle est équivalente à la fonction ASC de plusieurs versions du langage BASIC. Exemple :

```
. ? RANK ('A')
65
```

Fonction de longueur

LEN(<chaîne de caractères>)

Cette fonction génère un entier dont la valeur représente le nombre de caractères compris dans la chaîne citée. Exemple :

```
. STORE 'abc' TO CHAINE
. ? LEN(CHAINE)
  3
```

Fonction d'effacement d'enregistrement

*

C'est une fonction logique dont la valeur est vraie si l'enregistrement courant a été marqué pour effacement futur, autrement la valeur sera fausse. Exemple :

```
. ? *
.T.       (en supposant que la base de données est en USE et que
          l'enregistrement courant a été marqué pour effacement par la
          commande DELETE)
```

Fonction de fin-de-fichier

EOF

C'est une fonction logique dont la valeur est vraie si la fin du fichier en USE a été atteinte. (L'enregistrement sera le dernier enregistrement de la base de données). Exemples :

```
. ? EOF
.F.       (en supposant que la base de données est en USE et n'est pas
          positionnée sur le dernier enregistrement)
. GOTO BOTTOM
. ?  EOF
.F.
. SKIP
. ? EOF
.T.
```

Fonction de recherche de sous-chaîne

@(<chaîne 1>,<chaîne 2>)

Cette fonction génère un nombre entier dont la valeur est le numéro du caractère de la < chaîne 2 > où commence une sous-chaîne identique à la < chaîne 1 >. Si la chaîne 1 n'existe pas dans la chaîne 2, la valeur de la fonction « @ » sera nulle. La fonction « @ » est semblable à l'opérateur de sous-chaîne « $ » sauf que la fonction « @ » nous dira l'endroit où la première chaîne est trouvée dans la seconde chaîne. Exemple :

```
. ? @('def','abcdefghi')
 4
```

Fonction de majuscules

!(< chaîne de caractères >)

Cette fonction génère une même chaîne de caractères sauf que tous les caractères minuscules seront convertis en majuscules. Exemple :

```
. ? !('abc')
ABC
```

Fonction de conversion de nombre en caractère

CHR(< expression numérique >)

Cette fonction génère un caractère ASCII dont le code est l'expression numérique. Exemple, si expression a la valeur 13, CHR(13) génère un retour chariot en caractère ASCII. Cette fonction est utile lorsque l'utilisateur veut envoyer des contrôles directs à son matériel, le plus souvent des imprimantes. Exemple :

```
. ? 'abcd'+CHR(13)+'___'
abcd
```

Fonction de date

DATE()

Cette fonction générera une chaîne de caractères contenant la date. La date peut être entrée à chaque démarrage dBASE ou à n'importe quel moment par l'utilisation de la commande SET DATE TO. Exemple :

```
. ? DATE()
06/15/85
. STORE DATE() TO MEMVAR
06/15/85
. SET DATE TO 4 1 86
. ? DATE()
04/01/86
```

Fonction de fichier

FILE(< chaîne d'expression >)

C'est une fonction logique dont la valeur est vraie si le fichier dont le nom est la valeur de l'expression, sinon la valeur sera fausse. Exemple :

```
.? FILE('TRACE')
.T.
.USE TRACE
```

Fonction de type

TYPE(< expression >)

Cette fonction génère une chaîne d'un caractère contenant un 'C','N', ou 'L' d'après le type de l'expression : caractère, numérique ou logique. Exemple :

```
. STORE 1 TO X
. ? TYPE(X)
N
```

Fonction TRIM

TRIM(< chaîne de caractères >)

Cette fonction enlève les blancs inutiles à la droite d'un champ.

NOTE : Cette fonction ne doit pas être utilisée dans la commande INDEX, puisque la dimension de la clé doit être calculable pour l'usage interne de dBASE. Exemples :

```
. STORE 'ABC    ' TO S
. ? LEN(S)
  6
. STORE TRIM(S) TO S
. ? LEN(S)
  3
```

Opérations

Il existe quatre types d'opérations : arithmétique, comparaison, logique ou chaîne. Il est important de savoir que les deux opérandes doivent être du même type.

Il nous est possible d'ajouter des nombres entiers aux nombres entiers, d'accoler des caractères avec des caractères, mais l'addition d'un nombre entier avec un caractère donnera une erreur de syntaxe.

```
. STORE 3 TO A
  3
. STORE '3' TO B
3
. ? A+B

*** ERREUR DE SYNTAXE ***
  ?
? A+B
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ?
```

L'erreur découle de la différence d'interprétation par la machine des nombres et des caractères : un nombre 3 veut dire — 3 hex, alors que le caractère 3 a la valeur ASCII de 33 en hex. Le programme ne sait pas s'il doit faire l'addition ou la concaténation. En utilisant les mêmes variables que l'exemple précédent :

```
. ? A+VAL(B)
  6
```

La chaîne de caractères '3' a été convertie en nombre entier et l'addition a été exécutée.

Les opérateurs arithmétiques (génèrent des résultats arithmétiques)

+	: addition
—	: soustraction
*	: multiplication
/	: division
()	: ensemble

Exemples :

```
. ? (4+2)*3
 18
. ? 4+(2*3)
 10
```

(Exemple d'utilisation des parenthèses pour grouper les calculs).

Les opérateurs de comparaison (génèrent des résultats logiques)

<	: inférieur à
>	: supérieur à
=	: égal à
#	: différent de
<=	: inférieur ou égal à
>=	: supérieur ou égal à
$	: opérateur de sous-chaîne

(Si A et B sont des chaînes de caractères, A$B sera vrai si et seulement si la chaîne A est égale à B ou est contenue dans B). Exemples :

```
. ? 'abc'$'abcdefghi'
 .T.
. ? 'abcd'$'ghijkl'
 .F.
. DISPLAY FOR '8080'$TITRE
```

(Tous les enregistrements contenant '8080' dans le titre seront visualisés.)

Les opérateurs logiques (génèrent des résultats logiques)

.OR. : ou booléen
.AND. : et booléen
.NOT. : non booléen

Exemples :

```
. store t to a
 .T.
. store f to b
 .F.
. ? a .or. b
 .T.
. store .not. b to c
 .T.
? a .and. c
 .T.
```

Les opérateurs de chaîne (génèrent des résultats de chaîne)

\+ : concaténation de chaîne
− : concaténation de chaîne avec élimination des blancs inutiles.

Exemples :

```
. STORE 'ABCD  'TO A
ABCD
. STORE 'EFGH' TO B
EFGH
. ? A+B
ABCD  EFGH
. STORE 'ABCDE   ' TO A
ABCDE
. STORE '1234  67' TO B
1234  67
. ? A-B
ABCDE1234  67
```

Dans une concaténation de chaînes, les deux chaînes sont simplement ajoutées l'une à l'autre.

Dans une concaténation de chaînes avec effacement des blancs, les blancs inutiles sont déplacés à la fin de la chaîne. Ne sont pas altérés les blancs à gauche de la chaîne ou insérés dans la chaîne.

Priorité d'exécution
Les ensembles d'opérateurs arithmétiques, chaîne et logiques ont une priorité d'exécution (c'est-à-dire, que telle opération est à exécuter avant telle autre opération). Le tableau suivant vous donnera l'ordre de priorité pour chacun des trois principaux opérateurs. Exemple :

```
. ? 4+2*3
10
```

Priorité des opérateurs arithmétiques	Priorité des opérateurs de chaîne de caractères	Logique
1) parenthèses, fonctions 2) signes +,— 3) *,/ 4) +,— 5) relations	parenthèses, fonctions relations, $(option sous-chaîne) +,— (concaténation)	.NOT. .AND. .OR.

MACRO SUBSTITUTION

Lorsque dBASE rencontre dans une commande le signe & suivi du nom de la chaîne de caractères de la variable mémoire, il remplace le & et la variable mémoire par la chaîne de caractères. Ce qui permet à l'utilisateur de définir une seule fois les parties des commandes et de les appeler à n'importe quel moment dans ses différentes opérations.

Les macros sont utiles lorsque des expressions complexes doivent être utilisées fréquemment. Elles permettent aussi l'échange de paramètres à l'intérieur des fichiers de commandes. Tout caractère compris entre le signe & et le caractère spécial suivant (y compris l'espace), est pris comme nom de variable mémoire.

Si l'utilisateur désire ajouter des caractères à la substitution symbolique, le nom de la variable mémoire devra se terminer par un point. Tout comme le signe &, le point sera enlevé lors de la substitution.

Si le signe & n'est pas suivi d'un nom de variable mémoire correct, il n'y aura pas d'expansion et le signe & restera dans la ligne de commande.
Exemples :

```
. ACCEPT "Entrez la lettre de l'unité de disque" to DR
USE &DR:nom-du-fichier
```

(lors de l'exécution, si B a été entré en réponse à la demande ACCEPT, nous aurons USE B :nom-du-fichier)

```
. STORE 'DELETE RECORD ' TO T
&T 5
(à l'exécution, deviendra DELETE RECORD 5)
```

COMMENT FAIRE L'INTERFACE AVEC DES LOGICIELS AUTRES QUE dBASE

dBASE peut lire des données à partir des fichiers créés par des logiciels autres que dBASE (BASIC, FORTRAN, PASCAL), et peut générer des fichiers acceptés par ces logiciels.

La commande APPEND peut lire les fichiers de texte standard ASCII (utilisation des conventions CP/M pour une ligne de texte suivie par un retour chariot et alimentation d'une ligne) par la spécification de l'option SDF (System Data Format). De la même façon, la commande COPY générera des fichiers de format standard ASCII lorsque l'option SDF est utilisée. A moins que vous ne le spécifiez explicitement, les fichiers créés avec les options SDF et DELIMITED auront une extension .TXT.

Il existe des logiciels et langages qui lisent et écrivent des fichiers dans un format délimité. Sous cette forme, tous les champs sont séparés par des virgules et les chaînes de caractères sont entourées par des apostrophes. dBASE peut ajouter (APPEND) et copier (COPY) ces fichiers lorsque le mot-clé DELIMITED est compris dans la commande. SDF est sous-entendu lorsque DELIMITED est utilisé.

Puisque certains logiciels utilisent des apostrophes ou des guillemets pour délimiter les chaînes de caractères, APPEND accepte ces deux types de délimiteurs. La commande COPY génère normalement les apostrophes, mais accepte tout caractère défini par la phrase WITH de la clause DELIMITED. Il est hautement recommandé de n'utiliser que les apostrophes et les guillemets.

Il existe un cas spécial lorsqu'une virgule est utilisée dans la commande COPY avec la phrase WITH : tous les blancs à droite d'une chaîne de caractères et précédant les champs numériques sont ignorés, les chaînes de caractères ne seront pas entourées d'apostrophes ni de n'importe quel autre caractère.
Exemples :

```
.USE <nom-de-fichier>.DBF
.COPY TO <nom-de-fichier>.TXT DELIMITED WITH "

.USE <nom-de-fichier>.DBF
.APPEND FROM <nom-de-fichier>.DAT SDF
```

LES DIFFERENTES CATEGORIES DE COMMANDES

Pendant l'utilisation de dBASE, différentes commandes peuvent être combinées pour accomplir une tâche particulière. Certaines commandes dBASE sont élaborées d'après la structure des langages des machines les plus modernes. Ces commandes sont classées dans la catégorie des commandes FICHIER DE COMMANDE. Il existe quelques règles de contrôle spéciales pour l'utilisation de ces commandes.

Création de fichiers
Les commandes suivantes créent un fichier de base de données et des fichiers associés :

CREATE	crée une nouvelle structure de fichier de base de données.
COPY	crée des copies à partir de bases de données existantes.
MODIFY	modifie les structures des bases de données.
REPORT	crée un fichier de rapport.
SAVE	sauvegarde les variables mémoire dans un fichier.
INDEX	crée un fichier d'index.
REINDEX	ré-index un ancien fichier d'index.
JOIN	rassemble les sorties de deux bases de données.
TOTAL	totalise les données des enregistrements.

Addition de données
Les commandes suivantes ajoutent de nouveaux enregistrements dans les bases de données :

APPEND	ajoute des données à la fin d'un fichier.
CREATE	permet l'ajout de données lors de la création.
INSERT	insère des données dans un fichier.

Edition de données
Les commandes suivantes éditent les données à l'intérieur des bases de données :

CHANGE	édite les colonnes des champs.
BROWSE	édite et visualise un plein écran d'édition.
DELETE	marque un enregistrement pour effacement logique.
EDIT	change des champs de données spécifiques dans une base de données.
PACK	efface physiquement les enregistrements marqués.
RECALL	restitue les enregistrements logiquement effacés.
REPLACE	remplace des champs de données par des valeurs.
READ	remplace des données définies dans le plein écran par l'utilisateur.
UPDATE	permet la mise à jour batch d'une base de données.

Commandes de visualisation de données
Les commandes suivantes visualisent les données choisies d'une base de données :

@	visualise des données formatées par l'utilisateur à la console ou à l'imprimante.
BROWSE	visualise au maximum 19 enregistrements avec autant de champs que peut en contenir un écran.
COUNT	compte le nombre d'enregistrements remplissant certaines conditions.
DISPLAY	visualise les enregistrements, les champs et les expressions.
READ	visualise les données et les informations en mode plein écran.
REPORT	formate et visualise un rapport de données.
SUM	calcule et visualise le résultat des expressions d'un groupe d'enregistrements.
?	visualise une liste d'expressions.

Commandes de positionnement
Les commandes suivantes positionnent le pointeur sur l'enregistrement désiré :

CONTINUE	par les conditions spécifiées de la commande LOCATE, positionne le pointeur à l'enregistrement suivant.
FIND	positionne le pointeur à l'enregistrement correspondant à la clé du fichier d'index.
GOTO	positionne le pointeur à un enregistrement spécifique.
LOCATE	localise un enregistrement remplissant une certaine condition.
SKIP	positionne le pointeur à l'enregistrement plus ou moins n.

Commandes de manipulations de fichier
Les commandes suivantes affectent la totalité des fichiers de base de données :

APPEND	ajoute des fichiers dBASE ou des fichiers SDF (System Data Format).
COPY	copie les bases de données dans d'autres bases ou dans des fichiers SDF.
DELETE	efface les fichiers logiques.
DO	spécifie un fichier de commandes avec des commandes à exécuter ultérieurement.
RENAME	renomme un fichier.
SELECT	permet de passer d'un fichier en USE à un autre.
SORT	crée une copie de la base de données triée sur un des champs.
USE	spécifie le fichier de base de données utilisé pour toutes les opérations jusqu'à l'issue d'un autre USE.

Commandes de variable memoire
Les commandes suivantes manipulent les variables mémoire :

ACCEPT	stocke une chaîne de caractères dans une variable mémoire.
COUNT	stocke les calculs dans les variables mémoire.
DISPLAY	peut visualiser les variables mémoire.
INPUT	stocke les expressions dans les variables mémoire.
RESTORE	rétablit les variables mémoire sauvegardées.
SAVE	sauvegarde les variables mémoire du fichier.
STORE	stocke les expressions dans les variables mémoire.
SUM	stocke les sommes dans les variables mémoire.
WAIT	accepte l'entrée d'une clé simple dans les variables mémoire.

Commandes des fichiers de commande
Les commandes suivantes permettent le contrôle et l'usage des fichiers de commandes :

ACCEPT	permet l'entrée des chaînes de caractères dans des variables mémoire.
CANCEL	annule l'exécution du fichier de commandes.
DO	exécute les fichiers de commande et permet des boucles structurées dans ces fichiers.
IF	permet l'exécution conditionnelle des commandes.
ELSE	alterne le cours d'exécution des commandes à l'intérieur d'un IF.
ENDDO	termine une commande DO WHILE.
ENDIF	termine une commande IF.
INPUT	permet l'entrée des expressions dans les variables mémoire.
LOOP	revient au point de départ d'une commande DO WHILE.
MODIFY COMMAND	permet l'édition du fichier de commandes.
RETURN	termine un fichier de commandes.
SET	installe les paramètres de contrôle dBASE.
WAIT	suspend le traitement d'un fichier de commandes.

Commandes contrôlant le matériel
Les commandes suivantes contrôlent les périphériques tels que imprimantes et consoles :

EJECT	effectue un saut de page à l'imprimante.
ERASE	efface la console.

OPERATION PLEIN ECRAN

Clés de contrôle pour les opérations plein écran :

ctrl-E,A	Revient au champ de données précédent
ctrl-X,F	Avance au champ de données suivant
ctrl-S	Recule d'un caractère dans le champ de données
ctrl-D	Avance d'un caractère dans le champ de données
ctrl-Y	Met à blanc le champ de données courant
ctrl-V	Active/Désactive le mode Insertion
ctrl-G	Efface le caractère qui se trouve sous le curseur
Del	Efface les caractères à gauche du curseur
ctrl-Q	Abandonne le plein écran et redonne le contrôle à dBASE. Tout changement effectué dans les variables de la base de données n'est pas pris en compte.

En mode EDIT :

ctrl-U	Marque ou démarque les enregistrements pour effacement logique
ctrl-R	Ecrit l'enregistrement courant sur disque et visualise l'enregistrement précédent
ctrl-C	Ecrit l'enregistrement courant sur disque et visualise l'enregistrement suivant
ctrl-W ou ctrl-O	Ecrit l'enregistrement courant sur disque et sort du mode Edition d'écran

En mode MODIFY :

ctrl-N	Déplace tous les articles d'un champ pour permettre l'insertion d'un nouveau champ
ctrl-T	Efface le champ où se trouve le curseur et remonte tous les autres champs
ctrl-C	Permet le défilement des champs vers le bas
ctrl-R	Permet le défilement des champs vers le haut
ctrl-W	Ecrit les données sur disque et revient aux commandes dBASE
ctrl-Q	Sort du mode Edition sans sauvegarder les corrections

En mode APPEND, CREATE, ou INSERT :

ctrl-C ou ctrl-R	Ecrit l'enregistrement courant sur disque et traite l'enregistrement suivant

Pour terminer l'opération et revenir aux commandes dBASE, faites un retour chariot lorsqu'aucune correction n'a été effectuée et lorsque le curseur est à sa position initiale.

En mode BROWSE :

ctrl-U	Marque ou démarque un enregistrement pour effacement logique
ctrl-R	Ecrit l'enregistrement courant sur disque et visualise l'enregistrement précédent
ctrl-C	Ecrit l'enregistrement courant sur disque et visualise l'enregistrement suivant
ctrl-W	Ecrit l'enregistrement en cours sur disque et sort du mode Edition
ctrl-Z	Recule les champs de l'écran d'une position vers la gauche
ctrl-B	Avance les champs de l'écran d'une position vers la droite

LES COMMANDES

Les explications et les définitions des commandes de dBASE se trouvent dans ce chapitre. L'utilisateur devrait se familiariser avec ces explications fondamentales avant de continuer à lire les commandes.

Définition des symboles

La compréhension des symboles spéciaux dans le format des commandes de dBASE est d'une importance vitale. Veuillez lire attentivement et entièrement le tableau suivant.

Symbole	Définition
<commandes> ou <phrase>	signifie toutes les phrases valides de dBASE. Signifie également phrases complètes. Un IF sans un ENDIF (ou un DO WHILE sans ENDDO) est une phrase incomplète, alors que REPORT est d'elle-même une phrase entière.
<chaîne de caractères> ou <chaîne>	signifie n'importe quelle chaîne de caractères. Les chaînes de caractères sont les caractères entourés par des apostrophes, des guillemets ou des crochets.
<délimiteur>	signifie tout caractère spécial; les caractères spéciaux sont les caractères de ponctuation du clavier tels que les guillemets « », la parenthèse gauche (, la parenthèse droite), l'astérisque *, le signe égal =, et le signe @.
<exp>	signifie une expression; une expression peut être créée par l'assemblage des nombres, des fonctions, des noms de champs ou des chaînes de caractères d'une manière significative. Exemple : « 4 + 8 », « doc = '3' .or. doc = '4' », « $('abc' + &somestr,n,3) = 'abcdefg' ».
<exp list>	signifie une liste d'expressions séparées par des virgules. En général, des expressions simples sont utilisées. Deux des exemples du paragraphe précédent sont plutôt compliqués mais le premier est simple.
<champ>	signifie un nom de champ dans un enregistrement Exemples : article, coût, date, etc.
<champ list> ou <list>	signifie la liste des noms de champs d'un enregistrement séparés par des virgules.

NEXT n	signifie les n enregistrements suivants, y compris l'enregistrement en cours; NEXT commence par l'enregistrement en cours. n doit avoir une valeur littérale (c'est-à-dire ne doit être ni une variable mémoire ni une expression).
RECORD n	signifie seulement l'enregistrement n; rappel, n ne doit être ni une variable mémoire ni une expression.

Il existe d'autres symboles spéciaux utilisés dans le format des commandes. Ces symboles sont spécifiques aux commandes et seront expliqués en même temps que les commandes.

Règles d'opérations

Comme pour tout langage de commande, il existe des règles à suivre pour pouvoir exécuter un programme. Les règles suivantes traduisent le format général des commandes sous une forme plus spécifique.

1. Le verbe de n'importe quelle commande doit être le premier caractère non blanc d'une ligne de commande; les phrases peuvent suivre dans n'importe quel ordre. Un verbe est un mot d'action; CREATE, APPEND, REPORT, SET, DISPLAY et ERASE sont des exemples de verbes exécutant une action spécifique. Les phrases sont équivalentes aux adverbes; elles décrivent la portée de l'action. FOR, NEXT et WITH sont des mots qui commencent une phrase. Ces mots seront référencés comme mots-clés.

2. Peu importe leur nombre, les blancs sont utilisés pour séparer les mots et les phrases. Rappelez-vous cependant que ces blancs seront comptés dans la limite des 254 caractères, décrite dans la règle 3.

3. Toutes les commandes doivent avoir moins de 254 caractères de longueur (même après extension d'une macro).

4. Les commandes et les mots-clés peuvent être abrégés aux quatre premiers caractères (ou plus). Exemple : DISPLAY STRUCTURE est égal à DISP STRU, ou DISPL STRUCT, etc.
   ATTENTION : L'orthographe de l'abréviation doit être correcte.

5. Les lettres majuscules ou minuscules peuvent être utilisées pour entrer les commandes, les mots-clés, les champs de noms, les noms de variables mémoire, ou les noms de fichiers.

6. Il existe dans les commandes des parties optionnelles (ces parties peuvent être ignorées lors de l'utilisation de la commande). Les crochets sont utilisés dans le format des commandes pour spécifier la partie des phrases optionnelle. Ces

<fichier> ou <nom de fichier>	signifie n'importe quel nom de fichier. Ces noms doivent obéir aux règles définies dans le chapitre III.
<fichier de rapport>	signifie le nom d'un fichier de rapport. Voir fichiers de rapport et commande REPORT pour le pourquoi et le comment de ce type de fichier.
<fichier d'index>	signifie le nom du fichier où sont placées les informations indexées. Voir fichier d'index et commande INDEX pour le comment et le pourquoi de ce type de fichier.
<clé>	signifie le nom du champ indexé. Les clés sont importantes. Il peut y avoir plusieurs index sur une base de données. Chaque index est différent (ou est une combinaison de) clés. Les clés peuvent être des expressions ou des noms de champs. Pour plus d'informations, voir la commande INDEX.
<memvar>	signifie variable mémoire; les variables mémoire sont les variables créées par la commande STORE ou par l'utilisation d'une commande qui sauvegarde certaines valeurs pour une utilisation future (ACCEPT, INPUT, etc.). dBASE permet l'utilisation de 64 variables mémoire maximum.
<memvar list>	signifie une liste de variables mémoire séparées par des virgules.
<n>	signifie un littéral. Les littéraux sont des nombres qui ne découlent pas des variables mémoire ou des calculs. Exemple : « 4 + 8 » n'est pas un littéral, alors que « 4 » et « 9876 » sont des littéraux.
<étendue>	spécifie la portée d'une commande; étendue signifie jusqu'où la commande doit s'exécuter. Voir ci-dessous les trois valeurs possibles d'une étendue :
ALL	signifie tous les enregistrements du fichier. La commande ALL recherche tous les enregistrements du fichier pour le traitement. ALL est la valeur par défaut de certaines commandes. Pour d'autres commandes, la valeur par défaut sera l'enregistrement en cours. (Exemple : commande DELETE). La description de chaque commande nous dira quelle est la valeur par défaut de l'étendue. L'utilisation de la phrase FOR dans n'importe quelle commande aura ALL comme valeur par défaut.

phrases sont utilisées pour modifier l'action des commandes. Les mots en majuscules sont les mots-clés et ceux-ci doivent être entrés lorsque la phrase qui les contient est utilisée.

7. Un mot réservé est un mot-clé qui génèrera une erreur s'il est utilisé à tort. Il n'existe pas de mots réservés sous dBASE. Cependant, il vaut mieux éviter l'utilisation des mots-clés de dBASE dans vos noms de champs ou vos noms de fichiers.

8. Dans un fichier de commandes, les phrases dBASE doivent être imbriquées correctement. « Imbriquer une phrase » signifie qu'une phrase doit s'ajuster à l'intérieur d'une autre phrase. Ceci est particulièrement important pour l'exécution des groupes IF-ELSE-ENDIF et DO WHILE-ENDDO. Le fait de décaler les commandes dans un fichier permet de vérifier si les phrases sont correctement imbriquées.

```
DO WHILE .NOT. EOF
  phrase
  .
  IF A .AND. B
    .
    d'autres phrases
    .
  ELSE
    .
    DO WHILE A <= 57
       .
       encore d'autres phrases
       .
    ENDDO
    .
    et encore d'autres phrases
    .
  ENDIF
  .
  d'autres phrases qui n'en finissent pas
  .
ENDDO
```

```
DO WHILE .NOT. EOF
  .
  phrase
  .
  IF changements de valeur
  .
  ENDDO
  .
  encore des phrases
  .
  ENDIF
```

CHAPITRE II

LISTE DES COMMANDES dBASE

Vous trouverez dans les pages suivantes la liste alphabétique des différentes commandes ou instructions utilisables par dBASE II. Un descriptif, ainsi que des exemples d'utilisation, expliqueront chacune des commandes ou instructions.

?

? ? [<exp list>]
? ? [<exp list>]

La commande « ? » est une forme spéciale de la commande DISPLAY; c'est l'équivalent de DISPLAY OFF <exp>. Elle peut être utilisée pour montrer la valeur d'une expression ou d'une liste d'expressions. La commande « ? » (interprétée comme « qu'est-ce ») peut utiliser des variables mémoire, des champs de données, des constantes ou des fonctions. Le « ? » utilisé sans expression génère une ligne blanche.

La seconde forme de cette commande , « ? ? » fonctionne comme un simple « ? », sauf qu'il n'y aura pas de retour à la ligne ou de retour chariot avant l'impression d'une expression. Cette commande peut être utilisée dans les fichiers de commandes afin d'afficher ou d'imprimer plusieurs expressions sur la même ligne.

Exemples :

```
.USE EXEMPLE
. 4
. ? #
 4
. ? NOM
CHANG, LEE
. ? 5+9
 14
```

Le fichier de commandes suivant utilise le ? pour augmenter la clarté de la visualisation. Ce fichier a été exécuté en tapant : « DBASE H :FICHIER ».

```
set default to g
use trace index trace
disp stru
?
accept "Entrez la date du jour." to dte
set date to &dte
release dte
return
```

```
STRUCTURE DU FICHIER            : TRACE.DBF
NOMBRE D'ENREGISTREMENTS        : 02359
DATE DE LA DERNIERE MISE A JOUR : 10/08/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP   DIM    DECIMALE(S)
001     HAUT         C    024
002     TRCHP        C    005
003     DOC          C    024
004     DESCR        C    080
005     NATURE       C    010
006     STATUT       C    006
007     TESTEE       C    004
** TOTAL **            00154

Entrez la date du jour. : 14 10 85
```

@

@ <coordonnées> [SAY <exp> [USING <format>]]
[GET <variable> [PICTURE <format>]]

Cette instruction fonctionne avec les commandes SET FORMAT TO, ERASE, EJECT, CLEAR GETS et READ. Elle permet une visualisation rapide, à l'écran ou sur l'imprimante, d'informations spécifiques, à une coordonnée précise.
Suivant l'utilisation de SET FORMAT TO, la commande « @ » est interprétée de différentes façons. Les différentes combinaisons sont expliquées ci-dessous.

Les <coordonnées> « x,y » peuvent avoir une ou deux significations, elles seront soit des coordonnées d'écran soit des coordonnées d'imprimante. « x » signifie ligne et « y » colonne. Sur la plupart des écrans, « x » varie de 0 à 23 et « y » de 0 à 79, ce qui nous donne 24 lignes de 80 colonnes. dBASE utilise la ligne 60 pour envoyer ses messages à l'utilisateur : ce dernier devrait donc éviter l'utilisation de cette ligne. Les coordonnées « x » et « y » de l'imprimante varient de 0 à 254. Qu'elles soient utilisées pour l'écran ou l'imprimante, les coordonnées peuvent être des variables mémoire numériques, ou des expressions numériques. La commande SET FORMAT permet de choisir l'utilisation de l'écran ou de l'imprimante.

Lorsqu'une commande SET FORMAT TO SCREEN (option par défaut) est utilisée, la commande « @ » visualise les données à la console. Les coordonnées 0,0 signifient « angle supérieur gauche de l'écran de visualisation ». Les coordonnées 10,15 signifient « 11^{e} ligne et 16^{e} colonne ».
Nous vous rappelons que la ligne 0 de l'écran ne devrait pas être utilisée. Les commandes « @ » peuvent être émises dans n'importe quel ordre à l'écran. Par exemple, nous pouvons SAY quelque chose à la ligne 15 avant de SAY quelque chose à la ligne 10. Les colonnes peuvent également être affectées dans un ordre quelconque.

Lorsqu'une commande SET FORMAT TO PRINT est émise, la commande « @ » transmettra les données à l'imprimante. Le côté supérieur gauche du papier est repéré par les coordonnées 0,0. Afin de ne pas gaspiller de papier, les commandes « @ » destinées à l'imprimante doivent être émises dans l'ordre. Toutes les commandes de la ligne 5 doivent précéder les commandes de la ligne 6; de même, les commandes de la colonne 10 doivent précéder les commandes de la colonne 20, etc. Si l'ordre n'est pas respecté, un changement de page s'effectuera avant l'impression de la nouvelle ligne.

A l'émission de la commande SET FORMAT TO SCREEN, un ERASE effacera de l'écran toutes les informations qui y figuraient, réinitialisera les GET et les coordonnées à 0,0.

Un SET FORMAT TO PRINT émettra une commande EJECT pour saut de page et réinitialisera les coordonnées à 0,0.

La commande SAY permet de visualiser une expression qui ne sera pas altérée à l'édition par la commande READ. La phrase USING est utilisée pour formater une expression émise par SAY.
Les commandes SAY peuvent être utilisées soit à l'écran, soit sur l'imprimante. Les commandes GET, par contre, ne peuvent être reconnues sans la commande SET FORMAT TO SCREEN.

La commande GET visualise la valeur courante d'un champ de variable ou d'une variable mémoire. Avant l'émission d'une phrase GET, la variable doit exister pour être éditée, plus tard, par la commande READ. La phrase PICTURE peut être utilisée avec la phrase GET pour permettre un formatage spécial et une validation d'entrée des données (voir commande READ). Si la clause PICTURE n'est pas utilisée, le type de données (caractère, numérique ou logique) formera une PICTURE implicite.

Lors de l'utilisation de la commande GET, si le type de données du champ de variable ou des variables mémoire est logique, seuls les caractères 'T', 'F', 'Y' et 'N' (ou leurs équivalences en minuscules) pourront être entrés.

Nous pouvons utiliser au maximum 64 commandes GET. Pour réinitialiser les GET existants, utilisons soit la commande ERASE, soit la commande CLEAR GETS.

Lorsque l'option SET FORMAT TO SCREEN est active et que « @ » n'utilise ni SAY ni GET, des espaces seront affichés à partir de la position définie par les coordonnées (ligne, colonne) jusqu'à la fin de la ligne. Ainsi, « @10,0 » imprimera des espaces sur la totalité de la 11^{e} ligne.

Lorsqu'un SET FORMAT TO SCREEN est utilisé, un READ doit être émis afin d'exploiter les instructions GET (voir la commande READ). Toutefois, lorsqu'un SET FORMAT TO PRINT est utilisé, les commandes «@» n'ont pas besoin de commande READ pour compléter leur action.

Le fait de pouvoir imprimer sans passer par une commande READ permet à l'utilisateur de formater directement ses sorties sur du matériel pré-imprimé (exemple : chèques, ordres d'achat, etc.). L'utilisateur se souviendra seulement que les commandes « @ » doivent être émises aux positions où l'imprimé devra être complété.

L'utilisation de SET FORMAT TO PRINT est souvent nécessaire pour imprimer plus d'un article. La possibilité de substituer les variables mémoire aux valeurs des coordonnées est importante. Le fichier de commandes suivant est un exemple de génération de rapport spécial.

```
SET FORMAT TO PRINT
GOTO TOP
STORE 7 TO CNTR
DO WHILE .NOT. EOF
   IF CNTR >= 50
      EJECT
      STORE 7 TO CNTR
   ENDIF
   @ CNTR,12 SAY P USING 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   @ CNTR,48 SAY D USING 'XXXXXXXXXX'
   @ CNTR,64 SAY P1 USING 'XXXXXXXXXXXXXXXXX'
   @ CNTR,88 SAY U USING 'XXXXXXXXXX'
   @ CNTR,104 SAY P2 USING 'XXXXXXXXXXXXXXXXX'
   IF RCD <> 0
      @ CNTR,130 SAY RCD USING '9999'
   ENDIF
   STORE CNTR + 1 TO CNTR
   SKIP
ENDDO
RETURN
```

Dans cet exemple, un changement de page s'effectuera après impression de 57 lignes. Son but est d'imprimer la plupart des champs d'une base de données (et plus tard, sélectivement, un de ces champs). Prenez la précaution de réserver suffisamment de place pour la variable de la phrase SAY. Si le masque défini par USING est plus court que la variable ou le champ, la variable ou le champ sera tronqué. Le <format> du USING (les chaînes 'XXX...X') est expliqué plus loin.

Lors de l'utilisation de la commande SET FORMAT TO PRINT, l'impression à des positions précises (à ligne, colonne) de différentes informations sera correcte dans la mesure où tout ou partie de deux informations distinctes n'utilise pas une position commune sur la même ligne. Dans ce cas, l'impression de la fin de la ligne se fera en une seule et unique position, à l'extrême droite (superposition de caractères). Pour le SET FORMAT TO SCREEN, la dernière information émise se superposera à l'information précédente.

La dernière forme de la commande SET FORMAT est : SET FORMAT TO <fichier de format>. L'utilisation de cette commande avec émission de READ, permet aux commandes « @ » d'être lues dans le <fichier de format>. L'utilisateur peut ainsi définir un écran formaté pour des présentations spécifiques. Il est important de noter que l'utilisation des fichiers de format n'est pas nécessaire pour l'utilisation des « @ », ces derniers pouvant figurer dans les fichiers de commandes. Voir la commande READ pour plus d'informations.

Formats :

Les fonctions PICTURE et USING ont pour but de définir un format d'édition ou de saisie spécifique sur l'écran ou sur papier. Le tableau suivant définit les différentes possibilités de masque d'édition (SAY), ou de saisie (GET) :

Caractère de format	Fonction SAY	Fonction GET
#	imprime ou affiche uniquement les nombres	permet seulement la saisie d'un nombre (1,2,...,8,9,0) et des caractères « . », « + », « — » et « » (espace)
9	identique à #	identique à #
X	imprime le caractère qui suit	permet la saisie de tous les caractères
A	imprime le caractère qui suit	permet seulement la saisie des caractères alphabétiques
$ ou *	imprime un nombre, un $ ou un * à l'emplacement nul d'un champ numérique	imprime les données telles qu'elles sont
!	sans effet	convertit les caractères alphabétiques minuscules en majuscules

Exemple :

```
. @ 5,1 SAY 'ENTREZ LE NUMERO DE TELEPHONE' GET TNO PICTURE '(99)999-99-99'
```

Le message 'ENTREZ LE NUMERO DE TELEPHONE' sera visualisé suivi par : '(bbb)bbbb-bbbb' (b = blanc). La valeur de TNO avait été initialisée à blanc. Lors de l'utilisation de la commande READ, seuls les chiffres (0 à 9), ainsi que les caractères « . »,« + », « — » et les espaces pourront être saisis. Après la saisie, le contenu de TNO pourrait être '(32)123-45-67'. Tous les caractères non fonctionnels du format PICTURE (« (»,« — ») seront stockés dans la variable.

```
. @ 10,50 SAY HEURES*TAUX USING '*******.99'
```

Puisqu'il n'y a pas de phrase GET, la commande « @ » peut être utilisée soit à l'écran, soit sur l'imprimante. « @ » peut vous permettre de libeller des chèques. Le signe « * » sera imprimé sur toutes les positions non remplies à gauche du montant à imprimer. Par exemple : si heures = 40 et taux = 12.50, la visualisation sera '****500.00'. Cette caractéristique est utilisée pour éviter tout risque de falsification du montant à payer.

Lorsque des virgules sont utilisées dans la partie entière d'un format, elles seront remplacées par le caractère les précédant, si aucune valeur significative ne se trouve à gauche de cette virgule.

```
. @ 10,50 SAY HEURES * TAUX USING '$$$,$$$.99'
```

La sortie sera : $$$$500.00 et non $$$,500.00.

En principe, nous devrons émettre des commandes « @ » puis, si des phrases GET ont été incluses, une commande READ permettra l'édition ou l'entrée des données dans les variables demandées par GET. Dans l'exemple suivant, l'écran est formaté par plusieurs GET et, une base de données est remplie d'informations d'après ces GET. Le dernier enregistrement de la base de données aura un « O » dans le champ « nom ». Cet enregistrement sera effacé puisqu'il est inutile.

```
SET FORMAT TO SCREEN
USE F:EXEMPLE
ERASE
DO WHILE NOM ! '0'
  APPEND BLANC
  @ 5,0 SAY "ENTREZ LE NOM SUIVANT" ;
        GET NOM PICTURE 'XXXXXXXXXXXXXXXXXXX'
  @ 6,0 SAY "ENTREZ LE NUMERO DE TELEPHONE" ;
        GET TELE:EXTSN PICTURE 'XXXXX'
  @ 6,40 SAY "ENTREZ ARRET COURRIER" ;
        GET COU:STOP PICTURE 'XXXXXXXXXX'
  READ
ENDDO
GOTO BOTTOM
DELETE
PACK
LIST
RETURN
```

Les commandes suivantes affectent l'opération de la commande « @ » :

* SET INTENSITY ON/OFF (la valeur par défaut est ON) affecte l'intensité de l'écran pour les GET et SAY.

* SET BELL ON/OFF (la valeur par défaut est ON) affecte le signal d'alarme lorsqu'un caractère incorrect est entré ou que la limite d'entrée des données est atteinte.

* SET COLON ON/OFF (la valeur par défaut est ON) affecte les variables de GET si celles-ci sont délimitées par « : ».

* SET DEBUG ON/OFF (la valeur par défaut est OFF) permet de trouver facilement les erreurs des commandes « @ » en changeant les messages ECHO et STEP de l'imprimante.

* SET SCREEN ON/OFF (la valeur par défaut est ON) permet l'utilisation des opérations plein écran.

* SET FORMAT TO SCREEN/PRINT/<fichier de format> détermine la destination des sorties (écran ou imprimante). SET FORMAT TO <fichier de format> établit un fichier de format avec des commandes « @ » pour la commande READ. SCREEN est la valeur par défaut.

* READ permet la saisie des différentes variables définies précédemment par l'instruction GET.

ACCEPT

ACCEPT [« <chaîne> »] TO <memvar>

Tout comme la commande INPUT, cette commande permet l'entrée des chaînes de caractères dans les variables mémoire. Les apostrophes entourant les chaînes de caractères ne sont pas obligatoires avec la commande ACCEPT. Peu importe l'entrée, ACCEPT crée une variable mémoire de type caractère en sortie. INPUT, par contre, détermine le type de données entré et crée une variable mémoire du même type.

Si nécessaire, <memvar> sera créé et la chaîne de caractères saisie y sera stockée. Si <chaîne> est présent, il sera visualisé à l'écran, suivi par « : », en tant que message avant acceptation de l'entrée. En réponse à la demande ACCEPT, si un retour chariot est entré, <memvar> recevra un simple caractère espace. Pour délimiter votre message de chaîne, vous pouvez utiliser soit des apostrophes, soit des guillemets, soit des crochets, cependant les signes de début et de fin doivent être semblables.

Exemples :

```
 .ACCEPT "ENTREZ LES PRENOMS DE LA PERSONNE" TO NOM
ENTREZ LES PRENOMS DE LA PERSONNE : Jean Jacques

.ACCEPT "ENTREZ LE NOM DE LA PERSONNE" TO NOM2
ENTREZ LE NOM DE LA PERSONNE : Denis Saurin

. DISP MEMO
NOM         (C)  Jean Jacques
NOM2        (C)  Denis Saurin
** TOTAL **      02 VARIABLES USED  00024 BYTES USED

. ACCEPT TO TOUT
 : TOUT CARACTERE

.DISP MEMO
NOM             Jean Jacques
NOM2            Denis Saurin
TOUT            TOUT CARACTERE
** TOTAL **      03 VARIABLES USED  00038 BYTES USED
```

APPEND

a. APPEND FROM <fichier> [FOR <exp>] [SDF] [DELIMITED WITH <delimiteur>]
b. APPEND BLANK
c. APPEND

Ces trois formes vous permettent d'ajouter des enregistrements dans la base de données en utilisation (USE). APPEND, CREATE et INSERT sont les seules commandes qui permettent d'ajouter des enregistrements à une base de données. APPEND et CREATE permettent d'ajouter plusieurs enregistrements en même temps, INSERT ne permet qu'un seul ajout.

Dans le cas « a. », les enregistrements à ajouter sont pris dans un autre fichier <fichier>. Si SDF est spécifié, les enregistrements sont supposés être en System Data Format (voir Chapitre I). Si les nouveaux enregistrements sont d'une dimension inférieure à ceux du fichier en cours d'utilisation, des espaces complèteront la partie vierge du nouvel enregistrement. Dans le cas contraire, tous les caractères excédant la dimension maximum des enregistrements du fichier en cours d'utilisation ne seront pas stockés. Les enregistrements sont ajoutés dans le fichier en USE jusqu'à détection de la fin du fichier d'origine.

Si le mot DELIMITED figure dans la commande APPEND, les enregistrements pris dans le fichier d'origine sont supposés utiliser un séparateur (Ex : « , ») entre chaque champ, ainsi qu'un séparateur (Ex : 0AH 0DH) entre chaque enregistrement. Certains langages de programmation génèrent des fichiers où les chaînes de caractères sont encadrées par des apostrophes ou des guillemets et les champs sont séparés par des virgules. En mode DELIMITED, dBASE enlève les apostrophes et les virgules des fichiers délimités et stocke les données dans la base structurée de destination.

Si les clauses SDF et DELIMITED ne sont pas utilisées, le fichier d'origine est supposé être structuré dBASE. Les structures des fichiers de destination (USE) et d'origine (FROM) sont comparées. Les champs de même nom dans les deux structures seront identifiés de telle façon que seules les données des champs de dénomination commune seront lues dans le fichier origine (FROM) afin d'être ajoutées dans le fichier en cours d'utilisation. Les compléments (espaces) ainsi que les troncatures de champs seront exécutés convenablement s'il y a lieu.

Si la phrase FOR est utilisée, dBASE ajoutera un par un les enregistrements du fichier origine (FROM) après vérification de la condition stipulée par FOR. Si l'expression est vraie, l'enregistrement sera ajoutée et dBASE passera à l'enregistrement suivant. Si l'expression est fausse, l'enregistrement sera ignoré et dBASE passera également à l'enregistrement suivant. Cette procédure se poursuit tant que la fin du fichier d'origine n'aura pas été rencontrée. En conclusion, le(s) champ(s) utilisé(s) dans l'expression conditionnelle doi(ven)t impérativement exister dans le fichier *de destination.*

Si la clause BLANK (forme b) est spécifiée, un enregistrement vierge est ajouté dans le fichier en cours d'utilisation (USE). Cet enregistrement pourra ensuite être modifié par la commande EDIT ou REPLACE.

Si la commande APPEND est utilisée sans clause spécifique (forme c), Les noms des différents champs de la structure du fichier en cours d'utilisation sont affichés, et la saisie d'un enregistrement est autorisée. La fin de saisie est obtenue par la frappe d'un retour chariot (RETURN, ENTER) en réponse à la première zone de saisie.

Si la base de données en USE est une base de données indexée, le fichier index spécifié dans la commande USE est automatiquement mis à jour lorsque de nouveaux enregistrements sont ajoutés (sauf pour APPEND BLANK). Tout autre fichier index associé à cette base de données doit être réindexé.

Lorsque vous faites APPEND en mode direct, la commande SET CARRY ON transfère les données de l'enregistrement précédent dans l'enregistrement en cours de saisie. Les corrections peuvent ensuite être effectuées. Cette opération est intéressante lorsque vous avez plusieurs enregistrements avec des données communes.

La commande APPEND est particulièrement utile lorsque vous devez compléter/modifier ou ajouter/effacer le contenu des champs dans une nouvelle base de données. Après la création (CREATE) d'une nouvelle base de données, la commande APPEND vous permettra de transférer tout ou partie d'une base existante vers cette nouvelle base. Des espaces seront stockés dans les champs dont la dénomination est spécifique à la nouvelle base.

Exemples :

```
.USE EXEMPLE

. DISPLAY STRUCTURE
STRUCTURE DU FICHIER             : EXEMPLE
NOMBRE D'ENREGISTREMENTS         : 00005
DATE DE LA DERNIERE MISE A JOUR : 31/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYPE  DIM    DECIMALE(S)
001     NOM           C    020
002     TELE:EXTSN    C    005
003     COU:STOP      C    010
** TOTAL **              00036

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357  123/456
00002  RODGERS, RENE         2468  180/103
00003  CASSIDY, BERNARD      3344  264/401
00004  CHANG, LEE            6743  190/901
00005  POST, WILEY           1011  84/13B

. APPEND

RECORD 00006
NOM  :       LANCASTER, RENE
TELE:EXTSN:  6623
COU:STOP :   170/430

RECORD 00007
NOM  :       NORRIS, R. "BOB"
TELE:EXTSN:  8093
COU:STOP :   427/396

RECORD 00008
NOM  :       (cr)
```

```
.DISPLAY ALL OFF NOM,TELE:EXTSN
NEUMAN, ALFRED E.     1357
RODGERS, RENE         2468
CASSIDY, BERNARD      3344
CHANG, LEE            6743
POST, WILEY           1011
LANCASTER, RENE       6623
NORRIS, R. "BOB"      8093

.APPEND FROM DUPE3
00007 ENREGISTREMENT(S) AJOUTE(S)

.DISPLAY ALL
00001 NEUMAN, ALFRED E.          1357  123/456
00002 RODGERS, RENE              2468  180/103
00003 CASSIDY, BERNARD           3344  264/401
00004 CHANG, LEE                 6743  190/901
00005 POST, WILEY                1011  84/13B
00006 LANCASTER, RENE            6623  170/430
00007 NORRIS, R. "BOB"           8093  427/396
00008 NEUMAN, ALFRED E.          1357
00009 RODGERS, RENE              2468
00010 CASSIDY, BERNARD           3344
00011 CHANG, LEE                 6743
00012 POST, WILEY                1011
00013 LANCASTER, RENE            6623
00014 NORRIS, R. "BOB"           8093

. APPEND BLANK

. DISPLAY
00015

. REPLACE NOM WITH 'RINEHART, JEAN'
00001 REMPLACEMENT(S)

. DISPLAY
00015 RINEHART JEAN
```

```
. DISPLAY ALL NOM,' ex =',TELE:EXTSN
00001  NEUMAN, ALFRED E.          ex = 1357
00002  RODGERS, RENE              ex = 2468
00003  CASSIDY, BERNARD           ex = 3344
00004  CHANG, LEE                 ex = 6743
00005  POST, WILEY                ex = 1011
00006  LANCASTER, RENE            ex = 6623
00007  NORRIS, R. "BOB"           ex = 8093
00008  NEUMAN, ALFRED E.          ex = 1357
00009  RODGERS, RENE              ex = 2468
00010  CASSIDY, BERNARD           ex = 3344
00011  CHANG, LEE                 ex = 6743
00012  POST, WILEY                ex = 1011
00013  LANCASTER, RENE            ex = 6623
00014  NORRIS, R. "BOB"           ex = 8093
00015  RINEHART, JEAN             ex =

. USE B:COURSE

. DISP STRU
STRUCTURE DU FICHIER            : B:COURSE.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 22/06/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM   DECIMALE(S)
001     ARTICLE      C    020
002     NO           N    005
003     PRIX         N    010    002
** TOTAL **            00036

. CREATE
*** DONNEZ LE NOM DU FICHIER : ACHAT
DONNEZ LA STRUCTURE DU FICHIER SELON LE FORMAT :
CHAMP     NOM   TYPE DIM DECIMALE(S)
 001     ARTICLE,C,25
 002     NO,N,5
 003     PRIX,N,10,2
 004     DATE,C,8
 005     (cr)
*** VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ? N

. USE ACHAT
```

```
. APPEND FROM B:COURSE
00008 ENREGISTREMENT(S) AJOUTE(S)

. LIST
00001  CONSERVES               5       0.75
00002  PAIN                    2       0.97
00003  BOEUF                   4       3.94
00004  ASSIETTES PAPIER        1       0.86
00005  FOURCHETTES PLASTIQUES  5       0.42
00006  SALADE                  2       0.53
00007  FROMAGE                 1       1.96
00008  LAIT                    2       1.30

. REPLACE ALL DATE WITH ' 7/ 4/86'
00008  REMPLACEMENT(S)

. LIST
00001  CONSERVES               5       0.75  7/ 4/86
00002  PAIN                    2       0.97  7/ 4/86
00003  BOEUF                   4       3.94  7/ 4/86
00004  ASSIETTES PAPIER        1       0.86  7/ 4/86
00005  FOURCHETTES PLASTIQUES  5       0.42  7/ 4/86
00006  SALADE                  2       0.53  7/ 4/86
00007  FROMAGE                 1       1.96  7/ 4/86
00008  LAIT                    2       1.30  7/ 4/86
```

L'exemple suivant vous montre comment ajouter des données provenant d'un fichier structuré de la façon suivante. Ce fichier aurait pu être créé par exemple avec l'une des différentes versions de BASIC.

structure du fichier : DELIM.DAT

```
'BARNETT, PIERRE',31415,6
'NICHOLS, BILL',76767,17
'MURRAY, CAROLINE',89793,4
'WARD, CHARLES A.',92653,15
'ANDERSON, JEAN REGINALD III','11528,16
```

Ajoutons le fichier dans un fichier structuré dBASE

```
. USE ORDRES

. DISP STRU
STRUCTURE DU FICHIER            : ORDRES.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYPE  DIM    DECIMALE(S)
001     CLIENT        C    020
002     PART:NO       C    005
003     MONTANT       N    005
** TOTAL **             00031

. LIST
00001  SWARTZ, JOE          31415    13
00002  SWARTZ, JOE          76767    13
00003  HARRIS, ARNOLD       11528    44
00004  ADAMS, JEAN          89793    12
00005  MACK, JAY            31415     3
00006  TERRY, HANS          76767     5
00007  JUAN, DON            21828     5
00008  SALT, CLARA          70296     9

. APPEND FROM DELIM.DAT DELIMITED
00005 ENREGISTREMENT(S) AJOUTE(S)

. LIST
00001  SWARTZ, JOE          31415    13
00002  SWARTZ, JOE          76767    13
00003  HARRIS, ARNOLD       11528    44
00004  ADAMS, JEAN          89793    12
00005  MACK, JAY            31415     3
00006  TERRY, HANS          76767     5
00007  JUAN, DON            21828     5
00008  SALT, CLARA          70296     9
00009  BARNETT, PIERRE      31415     6
00010  NICHOLS, BILL        76767    17
00011  MURRAY, CAROLINE     89793     4
00012  WARD, CHARLES A.     92653    15
00013  ANDERSON, JEAN  REGI 11528    16
```

Les exemples suivants vous montreront l'utilisation d'une expression conditionnelle (FOR), avec la commande : APPEND FROM <fichier>. Nous vous rappelons que le ou les champs sur le(s)quel(s) porte(nt) la condition, doi(ven)t figurer dans le fichier destination.

```
. USE CHECKS
. DISP STRU
STRUCTURE DU FICHIER            : CHECKS.DBF
NOMBRE D'ENREGISTREMENTS        : 00013
DATE DE LA DERNIERE MISE A JOUR : 18/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP   DIM    DECIMALE(S)
001     NUMERO       N    005
002     BENEFIT      C    020
003     MONTANT      N    010     002
004     CREDIT       L    001
005     DEBIT        L    001
** TOTAL **            00038

. LIST
00001  1 P.T.T.                   104.89 .F. .T.
00002  2 Gas Cie                    4.14 .F. .T.
00003  3 Electricite              250.31 .F. .T.
00004  4 Epicerie                1034.45 .F. .T.
00005 34 Personnel                561.77 .T. .F.
00006  6 Banque (frais)             4.00 .T. .T.
00007  7 Docteur PATOU            100.00 .T. .T.
00008  8 Pirates                  101.01 .F. .T.
00009  9 Garage                   500.01 .F. .T.
00010 10 Personnel                561.01 .T. .F.
00011 11 B.H.V.                    50.02 .F. .T.
00012 12 Personnel                561.77 .T. .F.
00013 13 Personnel                750.03 .T. .F.
```

```
. USE MOIS
. DISP STRU
STRUCTURE DU FICHIER            : MOIS.DBF
NOMBRE D'ENREGISTREMENTS        : 00003
DATE DE LA DERNIERE MISE A JOUR : 18/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYPE  DIM    DECIMALE(S)
001     NUMERO        N    005
002     MONTANT       N    010     002
003     CREDIT        L    001
** TOTAL **             00017

. LIST
00001     29       14.89 .T.
00002     16      764.09 .T.
00003     78       97.96 .T.
. APPEND FROM CHECKS FOR CREDIT
00006 ENREGISTREMENT(S) AJOUTE(S)
. APPEND FROM CHECKS FOR DEBIT
*** ERREUR DE SYNTAXE ***
                        ?
APPEND FROM CHECKS FOR DEBIT
DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ? N
```

La variable « DEBIT » ne figure pas dans la base de destination.

BROWSE

C'est une des plus puissantes commandes de dBASE pour l'édition et la visualisation des données. Les données d'un maximum de 19 enregistrements seront visualisées à l'écran (80 caractères par ligne). L'écran devrait être considéré comme une fenêtre sur une base de données. Vous pouvez faire défiler vos enregistrements vers le haut, vers le bas, vers la gauche ou vers la droite. Toutes les données peuvent être éditées en mode plein écran (voir Chapitre I).

Vous trouverez ci-dessous les touches de contrôle plein écran fonctionnant avec la commande BROWSE :

ctrl-E,A	revient au champ de données précédent
ctrl-X,F	avance au champ de données suivant
ctrl-D	avance d'un caractère
ctrl-S	recule d'un caractère
ctrl-Y	efface la ligne
ctrl-N	insère une ligne
ctrl-G	efface le caractère sous le curseur
Del	efface le caractère précédant le curseur
ctrl-Q	sort du mode BROWSE sans sauvegarde des corrections
ctrl-W	sort du mode BROWSE avec sauvegarde des corrections
ctrl-B	déplace la « fenêtre » vers la droite
ctrl-Z	déplace la « fenêtre » vers la gauche
ctrl-C	écrit l'enregistrement en cours et avance d'un enregistrement
ctrl-R	écrit l'enregistrement en cours et recule d'un enregistrement
ctrl-U	positionne ou enlève l'indicateur logique d'effacement

Exemple :

```
.BROWSE
```

CANCEL

Cette commande annule l'exécution d'un fichier de commandes pour revenir au mode conversationnel du clavier.

Exemple :

```
INPUT 'LE TRAITEMENT EST-IL TERMINE (Y/N)' TO X
IF X
  CANCEL
ENDIF
```

Cet exemple fait partie d'un fichier de commandes. La commande INPUT demande une réponse « oui » ou « non ». Si la réponse est affirmative (Y, y, T, ou t), la ligne de commande IF X sera satisfaite (car X sera logiquement vrai) et la commande CANCEL s'exécutera.

CHANGE

CHANGE [<étendue>] FIELD <liste> [FOR <exp>]

Cette commande permet à l'utilisateur d'effectuer des modifications dans sa base de données avec un minimum d'efforts. Tous les champs de données mentionnés dans la liste sont présentés à l'utilisateur dans l'ordre spécifié par <liste>. L'utilisateur a la possibilité d'entrer de nouvelles données, de les modifier ou de passer au champ suivant. Lorsque la <liste> est épuisée, CHANGE traite l'enregistrement suivant ainsi qu'il a été spécifié dans <étendue>. La valeur par défaut de <étendue> est l'enregistrement en cours.

Un champ peut être effacé dans sa totalité par la pression de CTRL-Y (suivi d'un retour chariot) en réponse au message CHANGE ?. La commande CHANGE peut être abandonnée par la pression de la touche ESCAPE.

Exemple :

```
. USE CARTES
. CHANGE FIELD DATE

ENREGISTREMENT : 00001

DATE : 19/08/85
CHANGE ? 85
TO         86

DATE : 19/08/86
CHANGE ? <enter>
```

CLEAR

CLEAR [GETS]

Si le mot GET est utilisé, tous les GETs en suspens (précédés de la commande « @ ») seront mis à blanc et l'écran est laissé intact. La commande ERASE permet également la remise à blanc des GETs en suspens, lors de l'effacement de l'écran.
Si GET n'est pas utilisé, la commande CLEAR réinitialise dBASE II. Toutes les bases de données en cours d'utilisation sont fermées, inutilisées; toutes les variables mémoire sont détruites et le secteur de travail primaire est à nouveau sélectionné.

La commande CLEAR devrait être utilisée au début d'un fichier de commandes, afin de se trouver dans un état connu.

Exemple :

```
. CLEAR
```

CONTINUE

Cette commande est utilisée avec la commande LOCATE. LOCATE et CONTINUE peuvent être séparées par d'autres commandes, cependant il existe des limites. Voir commande LOCATE pour plus d'informations.

COPY

COPY TO <fichier> [<étendue>] [FIELD <liste>] [FOR <exp>]
[SDF] [STRUCTURE] [DELIMITED [WITH <délimiteur>]]

Cette commande copie la base de données en cours d'utilisation dans un autre fichier. Le fichier peut être de format dBASE ou de format « System Data Format » (si l'option SDF est spécifiée).

Si la clause STRUCTURE est utilisée, seule la structure du fichier dBASE en cours d'utilisation est copiée dans le fichier de destination (TO).

Si une liste de champs est fournie après la clause FIELD, seul le contenu de ces champs sera transféré dans le fichier de destination (TO). COPY STRUCTURE FIELD <liste> copie uniquement la structure des champs cités dans la liste. Dans tous les cas, la nouvelle structure comprendra uniquement les champs spécifiés par la clause FIELD. Si la clause FIELD n'est pas utilisée, tous les champs seront copiés.

Si la clause SDF est spécifiée, seules les données (pas la structure) du fichier en cours d'utilisation (USE) sont copiées dans un autre fichier. Ce nouveau fichier sera de format standard ASCII, ce qui nous permet de générer des fichiers qui pourront ensuite être manipulés par d'autres logiciels que dBASE. Les clauses STRUCTURE et SDF ne peuvent être employées simultanément.

Si le mot DELIMITED est également utilisé dans la commande, les différents champs de chaînes de caractères du fichier de destination seront encadrés par des apostrophes et séparés les uns des autres par des virgules. L'encadrement des chaînes de caractères par des apostrophes est l'option par défaut de la spécification DELIMITED. La précision WITH peut être utilisée avec DELIMITED. Elle permet l'utilisation de n'importe quel caractère comme délimiteur. Si une virgule est utilisée comme délimiteur, les espaces complétant la partie droite d'une chaîne de caractères seront ignorés, aucune apostrophe n'encadrera les chaînes de caractères. La commande APPEND réagit uniquement aux apostrophes et aux guillemets.

Lors de l'utilisation de l'option DELIMITED ou SDF, l'extension du fichier de destination sera par défaut .TXT, si aucune autre n'est spécifiée. Dans les autres cas, le fichier de destination aura l'extension : .DBF.

Si le fichier de destination n'existe pas, il sera automatiquement créé.

Exemples :

```
. DISPLAY ALL OFF NOM,TELE:EXTSN
NEUMAN, ALFRED E.     1357
RODGERS, RENE         2468
CASSIDY, BERNARD      3344
CHANG, LEE            6743
POST, WILEY           1011
LANCASTER, RENE       6623
NORRIS, R. "BOB"      8093
```

```
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : EXEMPLE
NOMBRE D'ENREGISTREMENTS        : 00007
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP  NOM          TYPE  DIM    DECIMALE(S)
001    NOM           C    020
002    TELE:EXTSN    C    005
003    COU:STOP      C    010
** TOTAL **            00036

. COPY TO DUPE
00007 ENREGISTREMENT(S) TRANSFERE(S)

. COPY TO DUPE2 FOR TELE:EXTSN'8000'
00006 ENREGISTREMENT(S) TRANSFERE(S)

. USE DUPE2

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357   123/456
00002  RODGERS, RENE         2468   180/103
00003  CASSIDY, BERNARD      3344   264/401
00004  CHANG, LEE            6743   190/901
00005  POST, WILEY           1011   84/13B
00006  LANCASTER, RENE       6623   170/430

. USE EXEMPLE

. COPY FIELD NOM,TELE:EXTSN TO DUPE3
00007 ENREGISTREMENT(S) TRANSFERE(S)

. USE DUPE3

STRUCTURE DU FICHIER            : DUPE3
NOMBRE D'ENREGISTREMENTS        : 00007
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP  NOM          TYPE  DIM    DECIMALE(S)
001    NOM           C    020
002    TELE:EXTSN    C    005
** TOTAL **            00026
```

```
. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, RENE         2468
00003  CASSIDY, BERNARD      3344
00004  CHANG, LEE            6743
00005  POST, WILEY           1011
00006  LANCASTER, RENE       6623
00007  NORRIS, R. "BOB"      8093

. USE EXEMPLE

. COPY NEXT 4 TO DUPE5
00004 ENREGISTREMENT(S) TRANSFERE(S)

. USE DUPE5

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357   123/456
00002  RODGERS, RENE         2468   180/103
00003  CASSIDY, BERNARD      3344   264/401
00004  CHANG, LEE            6743   190/901

. USE ORDRES

. DISP STRUCTURE
STRUCTURE DU FICHIER              : ORDRES.DBF
NOMBRE D'ENREGISTREMENTS          : 00012
DATE DE LA DERNIERE MISE A JOUR : 01/07/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP  NOM          TYPE  DIM    DECIMALE(S)
001    CLIENT        C    020
002    PART:NO       C    005
003    MONTANT       N    005
** TOTAL **             00031
```

```
. LIST
00001  SWARTZ, JOE          31415    13
00002  SWARTZ, JOE          76767    13
00003  HARRIS, ARNOLD       11528    44
00004  ADAMS, JEAN          89793    12
00005  MACK, JAY            31415     3
00006  TERRY, HANS          76767     5
00007  JUAN, DON            21828     5
00008  SALT, CLARA          70296     9
00009  BARNETT, PIERRE      31415     6
00010  NICHOLS, BILL        76767    17
00011  MURRAY, CAROLINE     89793     4
00012  WARD, CHARLES A.     92653    15

. COPY TO DELIM.DAT DELIMITED
00012 ENREGISTREMENT(S) AJOUTE(S)

'SWARTZ, JOE          ','31415',    13
'SWARTZ, JOE          ','76767',    13
'HARRIS, ARNOLD       ','11528',    44
'ADAMS, JEAN          ','89793',    12
'MACK, JAY            ','31415',     3
'TERRY, HANS          ','76767',     5
'JUAN, DON            ','21828',     5
'SALT, CLARA          ','70296',     9
'BARNETT, PIERRE      ','31415',     6
'NICHOLS, BILL        ','76767',    17
'MURRAY, CAROLINE     ','89793',     4
'WARD, CHARLES A.     ','92653',    15
```

COUNT

COUNT [<étendue>] [FOR <exp>] [TO <memvar>]

Cette commande compte le nombre d'enregistrements d'un fichier en cours d'utilisation (USE). Si la clause FOR est spécifiée, seuls seront comptés les enregistrements correspondant à l'expression. Si la clause TO est utilisée, le résultat sera placé dans une variable mémoire. Cette variable est créée si elle n'existait pas déjà.

dBASE répond par le message :

```
COMPTEUR = xxxxx
```

Exemples :

```
. USE STOCK
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER              : STOCK
NOMBRE D'ENREGISTREMENTS          : 00009
DATE DE LA DERNIERE MISE A JOUR : 23/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP  NOM          TYPE  DIM    DECIMALE(S)
001    NUM:ARTI      N    006
002    NUM:CLASS     N    003
003    N:VENDR       N    005
004    DESCR         C    013
005    PRIX:UNIT     N    007      002
006    LOCATION      C    005
007    DISPO         N    004
008    VENDU         N    004
009    PRIX          N    007      002
** TOTAL **             00055

. DISPLAY ALL
00001  136928  13  1673 ADJ. INTERIM     7.13 189     9    0    9.98
00002  221679   9  1673 SM. COUTURE      5.17 173     4    1    7.98
00003  234561   0    96 PLASTIQUE        2.18 27    112   53    4.75
00004  556178   2   873 ADJ. INTERIM    22.19 117     3    0   28.50
00005  723756  73    27 COMPTEUR        19.56 354     6    1   29.66
00006  745336  13    27 PERCEUSE        12.65 63      7    2   15.95
00007  812763   2  1673 GLOBE           5.88 112     5    2    7.49
00008  876512   2   873 PRISE           3.18 45      7    3    4.25
00009  915332   2  1673 SUPPORT         1.32 97      7    3    1.98
00010  973328   0    27 CAN COVER       0.73 21     17    5    0.99

. COUNT
COMPTEUR = 00010

. COUNT FOR NUM:ARTI>500000
COMPTEUR = 00007
```

```
. COUNT FOR 'ADJ'$DESCR
COMPTEUR = 00002

. GOTO TOP

. COUNT FOR PRIX<10 NEXT 6
COMPTEUR = 00003

. GOTO TOP

. COUNT NEXT 6 FOR PRIX<10
COMPTEUR = 00003

. USE B:COURSE

. LIST
00001  CONSERVES               5       0.75
00002  PAIN                    2       0.97
00003  BOEUF                   4       3.94
00004  ASSIETTES PAPIER        1       0.86
00005  FOURCHETTES PLASTIQUES  5       0.42
00006  SALADE                  2       0.53
00007  FROMAGE                 1       1.96
00008  LAIT                    2       1.30

. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : B:COURSE.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 10/12/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM    DECIMALE(S)
001     ARTICLE      C    020
002     NO           N    005
003     PRIX         N    010      002
** TOTAL **            00036

. COUNT TO XX FOR PRIX>1
COMPTEUR = 00003

. ? XX
  3
```

CREATE

CREATE [< nom-de-fichier >]

Un nouveau fichier structuré dBASE est créé. L'utilisateur fournira la structure, les noms de champs et le nom du fichier.

Si le nom du fichier n'est pas spécifié dans la commande, un message affiché à l'écran demande à l'utilisateur le nom de son fichier :

```
*** DONNEZ LE NOM DU FICHIER :
```

L'utilisateur donne un nom de fichier d'après les restrictions suivantes : le nom peut contenir les caractères spéciaux autorisés par CP/M pour des buts spéciaux (par exemple, B : signifie « unité de disque B »).

Si le fichier existe déjà lors de l'émission de la commande CREATE, dBASE demande à l'utilisateur :

```
DESIREZ-VOUS DETRUIRE LE FICHIER EXISTANT ?
```

Question à laquelle l'utilisateur peut répondre affirmativement ou négativement, selon le cas.

Si c'est un nouveau fichier, ou si l'utilisateur a répondu affirmativement à la question de destruction, l'utilisateur peut maintenant définir la structure de son fichier. Le message suivant s'affiche :

```
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP    NOM,TYPE,DIMENSION,DECIMALE(S)
```

L'utilisateur peut maintenant entrer des noms de champs et la structure des informations associées. Un nom de champ est une chaîne de caractères d'une longueur maximale de 10 caractères et pouvant contenir des lettres alphabétiques, des chiffres et des « : ». Les noms de champs doivent commencer par un caractère alphabétique. Les champs peuvent être de trois types : chaîne de caractères, numériques ou logiques. Le type du champ est spécifié par l'un des trois caractères suivants :

C chaîne de caractères
N numérique
L logique

La dimension se rapporte à la longueur de champ (par exemple, une chaîne de caractères peut avoir 20 caractères de long, sa dimension est 20). Les données numériques peuvent être soit des nombres entiers soit des nombres décimaux. La dimension des nombres entiers est le nombre maximum de chiffres que ce champ peut contenir. Pour les nombres décimaux, deux dimensions sont demandées : la première correspond au nombre de chiffres pour la partie entière (séparateur décimal compris), la seconde au nombre de chiffres pour la partie décimale. Les données logiques ont une longueur d'un caractère.

Exemples :

```
. CREATE
*** DONNEZ LE NOM DU FICHIER : EXEMPLE
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP    NOM,TYPE,DIMENSION,DECIMALE(S)
 001     NOM,C,20
 002     TELE:EXTSN,C,5
 003     COU:STOP,C,10
 004     <enter>
*** VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ? Y

RECORD 00001
NOM  :       NEUMAN, ALFRED E.
TELE:EXTSN : 1357
COU:STOP :   123/456

RECORD 00002
NOM  :       RODGERS, RENE
TELE:EXTSN : 2468
COU:STOP :   180/103

RECORD 00003
NOM  :       CASSIDY, BERNARD
TELE:EXTSN : 3344
COU:STOP :   264/401
```

```
RECORD 00004
NOM  :        CHANG, LEE
TELE:EXTSN :  6743
COU:STOP :    190/901

RECORD 00005
NOM  :        POST, WILEY
TELE:EXTSN :  1011
COU:STOP :    84/13

RECORD 00006
NOM  :        <enter>


.DISPLAY STRUCTURE
*** AUCUNE BASE N'EST UTILISEE, DONNEZ SON NOM : EXEMPLE
STRUCTURE DU FICHIER              : EXEMPLE
NOMBRE D'ENREGISTREMENTS          : 00005
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYP    DIM   DECIMALE(S)
001     NOM           C     020
002     TELE:EXTSN    C     005
003     COU:STOP      C     010
** TOTAL **                00036

. DISPLAY ALL
00001  NEUMAN, ALFRED E.      1357   123/456
00002  RODGERS, RENE          2468   180/103
00003  CASSIDY, BERNARD       3344   264/401
00004  CHANG, LEE             6743   190/901
00005  POST, WILEY            1011   84/13B
```

DELETE

DELETE [<étendue>] [FOR <exp>]
DELETE FILE <nom de fichier>

Tous les enregistrements compris dans <étendue> (satisfaisant une éventuelle expression FOR) sont marqués pour effacement logique. La valeur par défaut de <étendue> est l'enregistrement en cours. Tant que la commande PACK n'est pas exécutée, les enregistrements ne sont pas physiquement effacés, cependant ces enregistrements (logiquement effacés) ne sont pas transférés, ajoutés ou triés. La commande RECALL permet de restituer au fichier les enregistrements logiquement effacés. Les enregistrements repérés pour un éventuel effacement peuvent être visualisés. Le repérage d'un enregistrement logiquement effacé est défini par la présence d'une étoile entre le numéro de l'enregistrement et le premier champ.

La seconde forme de la commande DELETE permet d'effacer du disque un fichier; l'espace qu'il occupait auparavant sera de nouveau disponible. Cependant, si le fichier est en cours d'utilisation, il ne peut pas être effacé.

Exemples :

```
. LIST
00001  136928  13  1673 ADJ. INTERIM     7.13 189     9    0    9.98
00002  221679   9  1673 SM. COUTURE      5.17 173     4    1    7.98
00003  234561   0    96 PLASTIQUE        2.18 27    112   53    4.75
00004  556178   2   873 ADJ. INTERIM    22.19 117     3    0   28.50
00005  723756  73    27 COMPTEUR        19.56 354     6    1   29.66
00006  745336  13    27 PERCEUSE        12.65 63      7    2   15.95
00007  812763   2  1673 GLOBE           5.88 112     5    2    7.49
00008  876512   2   873 PRISE           3.18 45      7    3    4.25
00009  915332   2  1673 SUPPORT         1.32 97      7    3    1.98

. DELETE RECORD 2
00001 ENREGISTREMENT(S) EFFACE(S)

. 5

. DELETE NEXT 3
00003 ENREGISTREMENT(S) EFFACE(S)
```

```
. LIST
00001  136928  13  1673 ADJ. INTERIM     7.13 189     9    0    9.98
00002 *221679   9  1673 SM. COUTURE      5.17 173     4    1    7.98
00003  234561   0    96 PLASTIQUE        2.18 27    112   53    4.75
00004  556178   2   873 ADJ. INTERIM    22.19 117     3    0   28.50
00005 *723756  73    27 COMPTEUR        19.56 354     6    1   29.66
00006 *745336  13    27 PERCEUSE        12.65 63      7    2   15.95
00007 *812763   2  1673 GLOBE           5.88 112     5    2    7.49
00008  876512   2   873 PRISE           3.18 45      7    3    4.25
00009  915332   2  1673 SUPPORT         1.32 97      7    3    1.98

. RECALL ALL

00004 ENREGISTREMENT(S) RESTITUE(S)

. LIST
00001  136928  13  1673 ADJ. INTERIM     7.13 189     9    0    9.98
00002  221679   9  1673 SM. COUTURE      5.17 173     4    1    7.98
00003  234561   0    96 PLASTIQUE        2.18 27    112   53    4.75
00004  556178   2   873 ADJ. INTERIM    22.19 117     3    0   28.50
00005  723756  73    27 COMPTEUR        19.56 354     6    1   29.66
00006  745336  13    27 PERCEUSE        12.65 63      7    2   15.95
00007  812763   2  1673 GLOBE           5.88 112     5    2    7.49
00008  876512   2   873 PRISE           3.18 45      7    3    4.25
00009  915332   2  1673 SUPPORT         1.32 97      7    3    1.98

. DISP FILES ON B
DATABASE FILES   #RCDS     LAST UPDATE
COURSE   DBF  00007     06/06/85
ACHAT    DBF  00007     06/05/85

. DELETE FILE B:ACHAT
*** LE FICHIER A ETE EFFACE

. DISPLAY FILES ON B
DATABASE FILES   #RCDS     LAST UPDATE
COURSE   DBF     00007     06/06/85
```

DISPLAY

a. DISPLAY [<étendue>] [FOR <exp>] [<exp list>] [OFF]
b. DISPLAY STRUCTURE
c. DISPLAY MEMORY
d. DISPLAY FILES [ON <unité de disque>] [LIKE <squelette>]
e. DISPLAY STATUS

DISPLAY est l'une des principales caractéristiques de dBASE. La finalité de toute gestion de données est la possiblité de visualiser ces données de différentes façons. La commande DISPLAY permet d'obtenir ce résultat.

Dans le cas « a. », tout ou partie de la base de données en cours d'utilisation (USE) est visualisée. Si les clauses <étendue> et FOR ne sont pas spécifiées, seul l'enregistrement en cours est visualisé. L'utilisation de la clause FOR permet la visualisation de tous les enregistrements du fichier pour lesquels la condition est satisfaite. Dans ce cas, tous les champs sont visualisés sauf si la spécification <exp list> est utilisée. Les expressions peuvent être des champs de données, des variables mémoire ou tout nombre littéral, caractère ou logique. Le numéro de chaque enregistrement précède chaque ligne. L'option OFF supprime la visualisation du numéro de chaque enregistrement.

Après visualisation d'un groupe de 15 enregistrements, DISPLAY attend l'appui sur une touche pour afficher les enregistrements suivants. Ceci permet à l'utilisateur de paginer ses écrans. La commande LIST est identique à DISPLAY bien que LIST fait défiler tous les enregistrements en même temps et non des groupes de 15 enregistrements. La touche ESCape met fin aux commandes DISPLAY et LIST.

Dans le cas « b. », seule la structure de la base de données en cours d'utilisation est visualisée.

Dans le cas « c. », toutes les variables mémoire sont visualisées, avec leur type et leur contenu.

Le cas « d. » est un moyen de visualiser tous les fichiers .DBF qui se trouvent sur l'unité de disque ainsi que différents renseignements liés à chaque fichier (nombre d'enregistrement(s), date de la dernière mise à jour). La phrase LIKE permet la visualisation d'autres types de fichiers. Le <squelette> est généralement de la forme *.type où type est TXT, FRM, MEM, ou toute autre chaîne de trois lettres. Ces fichiers sont visualisés de la même façon qu'avec la commande DIR de CP/M.

Le cas « e. » entraîne l'affichage des noms des fichiers de données en cours d'utilisation, ceux des fichiers index correspondants, ainsi que la définition de la clé de chaque index. La commande DISPLAY STATUS vous indique également l'état de toutes les options modifiables par la commande SET.

Exemples :

```
. USE B:STOCK

.DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : B:STOCK.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP    DIM   DECIMALE(S)
001     ARTICLE      C     020
002     PRIX         N     010    002
003     PART:NO      C     005
004     DISPO        N     005
** TOTAL **              00041 BYTES

. DISPLAY ALL ARTICLE,PART:NO, PRIX*DISPO,$(PART:NO,1,2) FOR ;
 PRIX > 100 .AND. DISPO > 2 OFF
PERCEUSE              89793      404997.00   89
TROMBONES             76767       15076.12   76
ANNEAUX               70296        1000.00   70

. DISPLAY MEMORY
NOM:CLIENT (C)  DANGLEMEYER, PATRICK
BUDGET     (N)   123456.70
STATUT     (L)  .T.
** TOTAL **      03 VARIABLES USED  00026 BYTES USED

. DISPLAY FILES ON B: LIKE *.FRM
TEST   FRM     ADMIN    FRM     ORDRES    FRM
```

```
. DISPLAY FILES
DATABASE FILES  #RCDS   LAST UPDATE
TEST     DBF    00077   00/00/00
ADRECS   DBF    00073   09/23/85
HISSTSTR  DBF    00000   06/29/85
TMPADMIN DBF
*** CE N' EST PAS UNE BASE DE DONNEES DE DBASE II ***
```

DO

a. DO <fichier>
b. DO WHILE <exp>
   <ligne(s) de commande>
   ENDDO
c. DO CASE
   CASE <exp>
   <ligne(s) de commande>
   CASE <exp>
   <ligne(s) de commande>
   .
   .
   .
   [OTHERWISE]
   <ligne(s) de commande>
   ENDCASE

Dans le cas « a. », le <fichier> est ouvert et lu. Ce fichier est connu sous le nom de fichier de commandes. Il comprend des commandes de dBASE. Les commandes DO peuvent être imbriquées jusqu'à 16 niveaux (par exemple, les fichiers de commandes contenant des commandes DO peuvent invoquer d'autres fichiers de commandes). Dans un fichier de commandes, le contrôle est repris avec une fin de fichier ou par une commande RETURN. Si un fichier de commandes a été appelé par un autre fichier, le contrôle est redonné au fichier appelant. Si, pendant l'exécution d'un fichier de commande, survient une commande CANCEL, tous les fichiers de commandes sont fermés, et la « main » est rendue à l'utilisateur.

Dans le cas « b. », si <exp> est évaluée comme une valeur logiquement vraie, les phrases suivant la commande DO sont exécutées jusqu'à la rencontre d'une commande ENDDO. Si <exp> est évalué comme une valeur logiquement fausse, le traitement se poursuit à partir de la ligne suivant le premier ENDDO rencontré.

Note : <ligne(s) de commande> signifie « ligne(s) de commande complète ». Le DO WHILE doit se terminer avec un ENDDO. Les phrases doivent s'imbriquer correctement. S'il existe un IF à l'intérieur d'un DO WHILE, le ENDDO ne doit pas figurer avant le ENDIF. Voir le Chapitre 9.2, Règle 8.

Exemples :

```
DO CALPAYE

DO WHILE .NOT.EOF
  DISPLAY NOM
  .
  .
  .
  SKIP
ENDDO
```

Le cas « c. » est une extension de la commande DO. Le nombre d'instructions CASE suivant la commande DO CASE n'est pas limité, l'utilisation de l'instruction OTHERWISE est facultative.

DO CASE est une procédure structurée. Chaque cas CASE ainsi que le cas OTHERWISE est traité individuellement. Chaque condition est définie par l'instruction CASE, toute autre condition non définie est traitée par l'instruction OTHERWISE. Reportez-vous au premier exemple suivant.

L'instruction DO CASE peut être assimilée à une imbrication de IF et de ENDIF. Le résultat de l'exécution de DO CASE est le même que celui d'une suite de IF et ENDIF.

```
DO CASE                                IF ARTICLE='ORANGES'
    CASE ARTICLE='ORANGES'                 traitements
      traitements                      ELSE
    CASE ARTICLE='POMMES'      =           IF ARTICLE='POMMES'
      traitements                            traitements
    OTHERWISE                              ELSE
      traitements                            traitements
    ENDCASE                                ENDIF
                                       ENDIF
```

L'<exp>ression de chaque instruction CASE est examinée. Si la première expression est vraie, le traitement suivant le premier CASE est exécuté. La fin d'un traitement DO CASE est défini par l'instruction ENDCASE. Si plus d'une instruction CASE est satisfaite seule, la première rencontrée est exécutée.

Lorsqu'aucune instruction CASE n'est satisfaite et que l'instruction OTHERWISE existe, le traitement suivant OTHERWISE est exécuté. Si aucune instruction CASE n'est satisfaite et qu'OTHERWISE n'existe pas, aucun traitement n'est effectué et le DO CASE se termine.

Toute commande placée entre le DO CASE et le premier CASE ne sera pas traitée.

Exemples :

```
DO CASE
    CASE ARTICLE="BANANE"
         traitement BANANE
    CASE ARTICLE="GINGEMBRE"
         traitement GINGEMBRE
    CASE ARTICLE="SOUPE"
         traitement SOUPE
    OTHERWISE
         traitement des autres produits
ENDCASE
```

Dans l'exemple ci-dessus, toutes les expressions utilisent un nom de champ identique (ceci n'est pas obligatoire), et chaque condition CASE peut être effectuée sur des noms de champ différents, comme le montre l'exemple de la page suivante.

```
DO CASE
    CASE JOUR="LUNDI"
         traitement pour LUNDI
    CASE TEMPS="PLUIE"
         traitement pour PLUIE
    CASE VILLE="PARIS"
         traitement pour PARIS
ENDCASE
```

S'il existe un Lundi pluvieux à Paris, seul le CASE pour LUNDI est traité.

Les expressions des CASE doivent être des expressions logiques pouvant faire intervenir des chaînes de caractères, n'importe quel type de données, des valeurs logiques ou numériques, ainsi que des opérateurs.

```
DO CASE
    CASE VARIABLE1 = 2 + 1
       traitement pour addition
    CASE .NOT. A
       traitement pour valeur logique
    CASE VARIABLE2 = "A"$"ABCDEF"
       traitement pour une chaîne logique
    OTHERWISE
       autres traitements
ENDCASE
```

La phrase ENDCASE est utilisée pour terminer une séquence DO CASE. Lorsque le traitement d'un CASE ou d'un OTHERWISE est terminé, le programme passe à la ligne suivant le ENDCASE.

EDIT

EDIT [n]

Cette commande permet à l'utilisateur de modifier le contenu des différents champs d'un enregistrement du fichier en cours d'utilisation. L'utilisation et le fonctionnement de la commande EDIT varient suivant le mode d'utilisation de l'écran de dBASE (voir la commande SET SCREEN).

Lorsque dBASE est en mode plein écran, l'édition se fait soit par EDIT soit par EDIT n (n représentant l'enregistrement à éditer). En l'absence de n, dBASE demande les coordonnées de l'enregistrement à éditer. Lorsque la commande EDIT est utilisée sans le mode plein écran, dBASE demande :

COORDONNEES :

L'utilisateur entre les coordonnées du champ de données à modifier et (en option) les nouvelles données. Les coordonnées du champ de données sont : le numéro d'enregistrement et le numéro du champ (ou le nom du champ). Si de nouvelles données sont spécifiées, dBASE remplace le contenu du champ par celles-ci. Dans le cas contraire, dBASE visualise le contenu du champ et propose à l'utilisateur une éventuelle modification. Si aucun changement n'est désiré, un retour chariot laisse le contenu du champ intact. Que les données du champ soient changées ou non, dBASE demande à l'utilisateur les coordonnées suivantes avec un autre message « COORDONNEES : ».

Après avoir entré les premières coordonnées, l'utilisateur peut omettre de donner à nouveau des valeurs aux coordonnées : dBASE utilise alors les valeurs précédentes. Nous pouvons sortir du mode EDIT par un retour chariot en réponse à la demande COORDONNEES.

La totalité du contenu d'un champ peut être effacée par un Ctrl-Y suivi d'un retour chariot lorsque le message CHANGE ? est visualisé. La modification d'un champ de données peut être abandonnée par la pression de Ctrl-Q. Cette commande ne sauvegarde pas les corrections effectuées et restitue au champ de données son contenu original.

Si le fichier en cours de modification est indexé et que le fichier index est utilisé simultanément, ce dernier est automatiquement mis à jour dans le cas où les modifications sont effectuées sur la ou les zones de clé. Si plus d'un fichier index est associé à la base de données, les fichiers inutilisés ne sont pas modifiés par EDIT. Ils doivent être réindexés.

Exemples :

```
. USE COURSE

.DISPLAY STRUCTURE
STRUCTURE DU FICHIER              : COURSE.DBF
NOMBRE D'ENREGISTREMENTS          : 00006
DATE DE LA DERNIERE MISE A JOUR : 07/03/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYP    DIM   DECIMALE(S)
001     ARTICLE       C     020
002     NO            N     005
003     PRIX          N     010    002

** TOTAL **                00046

. LIST
00001  CONSERVES                5        0.69
00002  PAIN                     2        0.89
00003  BOEUF                    4        3.59
00004  SALADE                   1        0.49
00005  LAIT (1 L)               2        1.19
00006  CHARBON                  1        0.69

. EDIT
COORDONNEES : 5,ARTICLE,LAIT (1/2 L)

COORDONNEES : 2,1

ARTICLE : PAIN

CHANGE ? N
TO      N DE MIE

ARTICLE : PAIN DE MIE
CHANGE ? <enter>
COORDONNEES : 6,1

ARTICLE : CHARBON
CHANGE ? ON
TO      ON, 5 SACS
```

```
ARTICLE : CHARBON, 5 SACS
CHANGE ? <enter>
COORDONNEES : ,2

NO :     1
TO : 2
COORDONNEES : 4

NO :     1
TO : 2
COORDONNEES : <enter>

. LIST
00001  CONSERVES              5        0.69
00002  PAIN DE MIE            2        0.89
00003  BOEUF                  4        3.59
00004  SALADE                 2        0.49
00005  LAIT (1/2 L)           2        1.19
00006  CHARBON 5 SACS         2        0.69
```

(Le fichier de commandes suivant permet à l'utilisateur de modifier un enregistrement à partir de la saisie de son numéro. Le « & » est essentiel pour l'exécution de ce programme, il effectue la conversion de la chaîne reçue par ACCEPT en un nombre que peut exploiter la commande EDIT.)

```
STORE '1' TO X
DO WHILE X <> '0'
  ACCEPT "Entrer un numéro d'enregistrement" TO X
  EDIT &X
ENDDO
```

EJECT

Cette commande permet d'effectuer un saut de page sur l'imprimante si celle-ci est sélectionnée (SET PRINT ON ou SET FORMAT TO PRINT). Elle remet à zéro les compteurs de lignes et de colonnes si le formatage de l'impression est utilisé (commande : @,ligne,colonne..).

Exemple :

```
. EJECT
```

ENDDO

Cette commande est utilisée pour terminer une boucle DO WHILE. Lors de son exécution, l'état de la variable conditionnant la boucle est testée. Selon l'état de celle-ci, le traitement reprend à partir de la ligne suivant le DO (état logique vrai), ou bien se poursuit à la ligne suivant le ENDDO (état logique faux).
Voir commande DO.

ERASE

Cette commande efface l'écran et positionne le curseur à l'angle supérieur gauche de l'écran. En utilisant la commande « (a » avec l'option SET SCREEN ON, ERASE efface de la mémoire les précédentes zones de saisie (GET).

Exemple :

```
. ERASE
```

FIND

FIND <chaîne de caractères> ou '<chaîne de caractères>'

Cette commande permet à dBASE de localiser, dans une base de données indexée, le premier enregistrement dont la clé est égale à <chaîne de caractères>. FIND permet une localisation rapide des enregistrements à l'intérieur d'une base de données indexée. Sur un système à disques souples, la commande FIND met en moyenne deux secondes pour trouver l'enregistrement recherché.

FIND fonctionne uniquement sur les bases de données indexées (voir description de la commande INDEX). Si la commande INDEX utilise une expression de chaîne de caractères comme clé, la commande FIND permet de retouver un enregistrement avec seulement les premiers caractères de la clé. L'enregistrement trouvé est le premier dont la <chaîne de caractères> correspond exactement au début ou à l'ensemble de la clé du fichier index. Par exemple : un enregistrement dont la clé est 'DURAND, JEAN' peut être trouvé par la phrasse 'FIND DUR' FIND localise toujours uniquement le premier caractère dont la clé est semblable à la <chaîne de caractères>. Même si le pointeur est déplacé plus bas dans le fichier, la recherche sur une même clé s'arrête toujours sur le premier enregistrement correspondant.

Si l'index est créé avec une clé numérique, l'enregistrement trouvé est le premier dont la clé est arithmétiquement égale à celle de la recherche.

Note : Si les clés index sont composées de caractères et de nombres, l'argument recherché est une chaîne de caractères avec ou sans apostrophes. Les apostrophes ne sont nécessaires que pour les chaînes de caractères dont la clé originale est précédée d'espaces. Dans ce cas, le nombre exact d'espaces doit figurer entre les apostrophes.

Si une variable mémoire est utilisée comme argument de recherche, elle doit être précédée de la macro-instruction : « & ». Exemple : FIND &NOM où NOM est une variable mémoire de chaîne de caractères. Les variables mémoire numériques doivent d'abord être converties en chaîne par la fonction STR avant de pouvoir être utilisées avec les macro-instructions. Voir le Chapitre 5 sur les macro-instructions.

Lorsqu'un enregistrement est localisé avec la commande FIND, il peut être traité comme n'importe quel autre enregistrement de la base de données : il pourra être interrogé, modifié, utilisé dans des calculs, etc. Les commandes de dBASE telles que LIST, REPORT, COPY, etc., traitent d'abord l'enregistrement trouvé, puis, en séquence, les autres enregistrements, suivant l'ordre croissant de la clé.

Si aucun enregistrement ne possède une clé identique à la <chaîne de caractères>, le message : « Enregistrement non trouvé » est affiché. Dans ce cas, la variable interne « # », contenant le numéro de l'enregistrement en cours a une valeur nulle.

Si vous désirez localiser un deuxième enregistrement sur la même clé, SKIP ou LOCATE FOR <exp> doivent être utilisées.

SET EXACT ON permet à la commande FIND de trouver un enregistrement dont la clé recherchée est strictement identique à celle du fichier index (exception faite pour les espaces à droite des données d'un champ.

Exemples :

```
. USE COURSE INDEX COURSNDX

. LIST
00003   BOEUF                   1       1.96
00006   CHARBON                 2       1.06
00001   CONSERVES               5       0.75
00005   LAIT                    2       0.75
00002   PAIN                    2       0.53
00004   SALADE                  2       1.30

. FIND LAIT

. DISPLAY
00005   LAIT                    2       0.75

. DISPLAY NEXT 3
00005   LAIT                    2       0.75
00002   PAIN                    2       0.53
00004   SALADE                  2       1.30

. FIND C

. DISPLAY
00006   CHARBON                 2       1.06

.FIND CONSE

00001   CONSERVES               5       0.75

. FIND C

. DISPLAY
00006   CHARBON                 2       1.06
```

FIND permet également la recherche avec un fichier indexé sur clés multiples, dans le cas où les différentes clés sont encadrées par des apostrophes.

```
. list
0000   Commen  vole              Bird  I« M«      IBM00   29/02/04
00005  Procédures                Bird, I. M.      IBM002  25/09/06
00002  Pêches et Chasses          Fish, U. R.     URF001  30/12/23
00008  Santé                     Knight and Gale  KG001   04/08/44
00006  Vacances en Europe        Knight and Gale  KG002   24/06/42
00004  Comment cuisiner          Lynch, I.        IL001   01/04/00
00003  Comment survivre          Lynch, M.        ML001   01/01/30
00007  Nana                      Zola,  E.        LS001   01/12/73
00009  L'argent                  Zola,  E.        LS002   24/04/73

. find "Bird, I.M.      IBM002"

. disp
00005  Procédures                Bird, I. M.      IBM002  25/09/06

. find "Lynch, M."

. disp
00003  Comment Survivre          Lynch, M.        ML001   01/01/30

. find "Zola,  E.      LS002"

. disp
00009  L'argent                  Zola,  E.        LS002   24/04/73
```

GO ou GOTO

```
a. GOTO RECORD <n>
b. GOTO TOP
c. GOTO BOTTOM
d. <n>
e. GOTO <memvar>
```

Cette commande est utilisée pour repositionner le pointeur d'une base de données.

Dans les cas « a. » ou « d. », le pointeur se positionne sur l'enregistrement numéro <n>. Le cas d est la commande abrégée du cas « a. ».

Pour les cas « b. » et « c. », le pointeur se positionne au premier/dernier (TOP/BOTTOM) enregistrement du fichier en cours d'utilisation. Lorsque le fichier en cours d'utilisation est indexé, le premier/dernier enregistrement n'est pas nécessairement le premier/dernier enregistrement physique de la base de données mais plutôt le premier/dernier enregistrement correspondant à l'index du fichier.

Le cas « e. » peut être utilisé pour se positionner sur le numéro d'enregistrement que contient une variable mémoire.

Exemples :

```
. USE COURSE

. GOTO RECORD 7
  7

. DISPLAY
00007  CHARBON                  2       0.75

. GOTO TOP

. DISPLAY
00001  CONSERVES                5       0.75

. LIST
00001  CONSERVES                5       0.75
00002  PAIN                     2       0.97
00003  BOEUF                    4       3.94
00004  SALADE                   2       0.53
00005  FROMAGE                  1       1.96
00006  LAIT                     2       1.30
00007  CHARBON                  2       0.75

. STORE 4 TO NUM:ENR
  4

. GOTO NUM:ENR

. DISP
00004  SALADE                   2       0.53
```

HELP

HELP <commande>

La commande HELP affiche un résumé des caractéristiques de la <commande>, ainsi que la syntaxe d'utilisation de celle-ci. Les différents messages d'aide sont inclus dans le fichier : DBASEMSG.TXT. Ce fichier peut être modifié par un éditeur de texte afin d'y ajouter de nouvelles informations. La présence de ce fichier sur le disque est facultative, elle est nécessaire uniquement si vous désirez utiliser la commande HELP.

Exemple :

```
. HELP CREATE
```

IF

```
IF <exp>
     <commandes>
[ELSE
     <commandes>]
ENDIF
```

Cette commande permet l'exécution conditionnelle de différents traitements. Elle est utilisée dans les fichiers de commandes. Lorsque <exp>ression est vraie, les commandes suivant le IF sont exécutées. Lorsque <exp>ression est fausse, les commandes suivant le ELSE sont exécutées. Si aucun ELSE n'est spécifié et que <exp>ression est fausse, toutes les commandes sont ignorées jusqu'à la rencontre d'un ENDIF. Le nombre d'imbrications de IF, n'est pas limité.

Note : <commandes> correspond à une ou plusieurs lignes du fichier de commandes. L'utilisation d'un IF nécessite la présence d'un ENDIF à la fin du traitement conditionnel. Si IF est utilisé dans une boucle DO WHILE, la commande ENDIF doit être rencontrée avant ENDDO. Voir le Chapitre I (Règles d'opérations), Règle 8, pour un complément d'information.

Exemples :

```
IF STATUT='MARIE'
   DO MCOUT
ELSE
   DO SCOUT
ENDIF


IF X=1
  STORE VILLE+ETAT TO LOCATION
ENDIF
```

INDEX

INDEX ON <expression> TO <nom fichier index>

Cette commande permet d'indexer le fichier en cours d'utilisation sur l'<expression>. L'<expression> représente la (les) zone(s) de clé. Cette commande permet la création d'un nouveau fichier (fichier index). Ce fichier contient les pointeurs correspondant aux enregistrements du fichier en cours d'utilisation. Le fichier index est créé de telle sorte que les enregistrements sont affichés dans l'ordre ascendant de la clé. Aucune modification n'est effectuée sur le fichier de données. Une organisation dans l'ordre décroissant de la clé peut être obtenue sur une expression numérique.

L'utilisation d'un fichier indexé permet, grâce à la commande FIND, de localiser rapidement les enregistrements d'une base de données. Il n'est pas nécessaire d'indexer une base de données à moins de vouloir optimiser l'application. Une base de données indexée peut être utilisée ultérieurement avec ou sans index.

Dans la plupart des cas, nous n'utilisons qu'une seule fois la commande INDEX pour un fichier donné. Par exemple, la commande APPEND met automatiquement à jour le fichier index lorsque de nouveaux enregistrements sont créés.

Si l'on désire utiliser de nouveau une base de données indexée, l'ouverture de la base doit être libellé de la façon suivante : USE <nom du fichier de la base de données> INDEX <nom du fichier indexé>.

Vous pouvez créer autant de fichiers index, pour une seule et même base, que vous le désirez. Cependant, seul le fichier index en cours sera automatiquement mis à jour par les commandes APPEND , EDIT, REPLACE, BROWSE ou READ.

De la même façon, lors du compactage (PACK) d'une base indexée, seul le fichier index en cours d'utilisation est mis à jour. Si d'autres fichiers index sont associés à cette base, il est nécessaire de les réindexer.

ATTENTION : La fonction TRIM ne doit pas être utilisée avec tout ou partie des zones de clé du fichier index. Si les fonctions $ ou STR sont utilisées dans les clés, la longueur de chaque zone doit être exprimée uniquement sous forme d'une valeur numérique (ni variables, ni expressions).
exemple : INDEX ON $(NOM,N,5) +STR(MONTANT,5) TO NDXFILE
mais pas : INDEX ON $(NOM,N,N+5)+STR(MONTANT,DIMVAR) TO NDXFILE

Exemples :

```
. USE ACHAT

. LIST
00001   CONSERVES              5       0.75
00002   PAIN                   2       1.06
00003   BOEUF                  4       4.33
00004   ASSIETTES PAPIER       1       0.94
00005   FOURCHETTES            5       0.42
00006   SALADE                 2       0.53
00007   FROMAGE                1       1.96
00008   LAIT                   2       1.30
00009   CHARBON                2       0.75

. DISPLAY STRUCTURE
STRUCTURE DU FICHIER             : ACHAT.DBF
NOMBRE D'ENREGISTREMENTS         : 00009
DATE DE LA DERNIERE MISE A JOUR  : 07/03/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYP   DIM    DECIMALE(S)
001     ARTICLE       C    020
002     NO            N    005
003     PRIX          N    010     002
** TOTAL **              00036
```

```
. NOTE CREATION D'UN FICHIER INDEXE ACHNDX

 . INDEX ON ARTICLE TO ACHNDX

 . NOTE MAINTENANT LISTE DU FICHIER INDEXE

. LIST
00004  ASSIETTES PAPIER          1      0.94
00003  BOEUF                     4      4.33
00009  CHARBON                   2      0.75
00001  CONSERVES                 5      0.75
00005  FOURCHETTES               5      0.42
00007  FROMAGE                   1      1.96
00008  LAIT                      2      1.30
00002  PAIN                      2      1.06
00006  SALADE                    2      0.53

. NOTE UTILISATION DE LA COMMANDE FIND SUR FICHIER INDEXE

. FIND LAIT

. DISPLAY
00008  LAIT                      2      1.30

. FIND CONSE

. DISPLAY
00001  CONSERVES                 5      0.75

. SKIP
ENREGISTREMENT : 00005

 DISPLAY
00005  FOURCHETTES               5      0.42

. SKIP -1
ENREGISTREMENT : 00001

. DISPLAY
00001  CONSERVES                 5      0.75
```

```
. USE ACHAT

. LIST
00001  CONSERVES                5      0.75
00002  PAIN                     2      1.06
00003  BOEUF                    4      4.33
00004  ASSIETTES PAPIER         1      0.94
00005  FOURCHETTES              5      0.42
00006  SALADE                   2      0.53
00007  FROMAGE                  1      1.96
00008  LAIT                     2      1.30
00009  CHARBON                  2      0.75

. USE ACHAT INDEX ACHNDX

. LIST
00004  ASSIETTES PAPIER         1      0.94
00003  BOEUF                    4      4.33
00009  CHARBON                  2      0.75
00001  CONSERVE                 5      0.75
00005  FOURCHETTES              5      0.42
00007  FROMAGE                  1      1.96
00008  LAIT                     2      1.30
00002  PAIN                     2      1.06
00006  SALADE                   2      0.53

. USE LIVRES
. DISP STRU
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : LIVRES.DBF
NOMBRE D'ENREGISTREMENTS        : 00009
DATE DE LA DERNIERE MISE A JOUR : 18/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYP   DIM    DECIMALE(S)
001     TITRE         C    025
002     AUTEUR        C    015
003     CAT:NUM       C    006
004     ARR:DTE       C    008
** TOTAL **              00055
```

```
. INDEX ON AUTEUR + CAT:NUM TO LIVRES
00009 ENREGISTREMENT(S) INDEXE(S)

. list
00001  Comment voler          Bird, I. M.      IBM001 29/02/04
00005  Procédures            Bird, I. M.      IBM002 25/09/06
00002  Pêches et Chasses     Fish, U. R.      URF001 30/12/23
00008  Santé                 Knight and Gale  KG001  04/08/44
00006  Vacances en Europe    Knight and Gale  KG002  24/06/42
00004  Comment cuisiner      Lynch, I.        IL001  01/04/00
00003  Comment survivre      Lynch, M.        ML001  01/01/30
00007  Nana                  Zola, E.         LS001  01/12/73
00009  L'argent              Zola, E.         LS002  24/04/73
```

INPUT

INPUT [« <message> »] TO <memvar>

Cette commande permet la saisie d'expressions dans des variables mémoire. Elle peut être utilisée à l'intérieur des fichiers de commandes. Si nécessaire, <memvar> est créée et l'expression est stockée dans <memvar>. Si un <message> est utilisé, il est affiché devant la zone de saisie.

Le type de <memvar> est le même que celui des données saisies. Si la saisie est une chaîne de caractères, <memvar> sera de type caractère; si la saisie est une expression numérique, <memvar> sera de type numérique. Dans le cas de la saisie de « T » ou de « Y » (pour Vrai ou Oui), <memvar> sera une variable logique de valeur vraie. Par contre, si un « F » ou « N » (pour Faux ou Non) est entré, >memvar< sera une variable logique de valeur fausse. La fonction TYPE peut être utilisée pour déterminer clairement le type d'entrée.

Nous pouvons utiliser soit des apostrophes, soit des guillemets, pour encadrer la chaîne de caractères saisie. Cependant, les délimiteurs de début et de fin doivent être semblables.

INPUT doit être utilisé pour entrer uniquement des données numériques et logiques. La commande ACCEPT est utilisée pour entrer des chaînes de caractères.

Exemples :

```
. INPUT TO X
:3
  3

. INPUT TO X
:23/17.000+X
 4.352

. INPUT 'DONNEZ VOS INFORMATIONS' TO Q
DONNEZ VOS INFORMATIONS : 12345
  12345

. INPUT 'ENTREZ T SI TOUT EST CORRECT' TO LOG
ENTREZ T SI TOUT EST CORRECT : T
.T.

. INPUT "ENTREZ UNE CHAINE DE CARACTERES" TO CHAR
ENTREZ UNE CHAINE DE CARACTERES : 'LA CHAINE DOIT ETRE DELIMITEE'
LA CHAINE DOIT ETRE DELIMITEE

. DISP MEMO
X           (N)   3
Z           (N)   4.352
Q           (N)   12345
LOG         (L)   .T.
CHAR        (C)   LA CHAINE DOIT ETRE DELIMITEE
** TOTAL **        05 VARIABLES USED  00047 BYTES USED
```

INSERT

INSERT [BEFORE] [BLANK]

Cette commande permet d'insérer un enregistrement au sein du fichier de données. La commande INSERT ne permet l'insertion que d'un seul enregistrement à la fois.

La phrase BEFORE est utilisée pour effectuer l'insertion avant l'enregistrement en cours. Autrement, le nouvel enregistrement est placé juste après celui en cours. A moins que vous n'utilisiez la précision BLANK, les messages apparaissant à l'écran pour l'entrée des données sont identiques aux commandes CREATE et APPEND. Si la précision BLANK est spécifiée, un enregistrement vierge est inséré.

Si la commande SET CARRY ON est utilisée, les informations de l'enregistrement précédent sont recopiées dans le nouvel enregistrement.

Nous devons éviter si possible d'exécuter la commande INSERT dans une grande base de données non indexée (cette opération étant très longue à s'exécuter). Lors de l'insertion d'un enregistrement dans un fichier indexé, le temps d'exécution est identique à celui d'une création (APPEND), quelle que soit la dimension du fichier de données.

Exemples :

```
. USE ACHAT

. LIST
00001  CONSERVES          5     0.69
00002  PAIN               2     0.89
00003  BOEUF              4     3.59
00004  SALADE             2     0.49
00005  LAIT (1/2 L)       2     1.19
00006  CHARBON 5 SACS     2     0.69

. GOTO RECORD 4

. INSERT

RECORD 00005

ARTICLE : FROMAGE
NO      : 1
PRIX    : 1.79
```

```
. LIST
00001  CONSERVES              5      0.69
00002  PAIN                   2      0.89
00003  BOEUF                  4      3.59
00004  SALADE                 2      0.49
00005  FROMAGE                1      1.79
00006  LAIT (1/2 L)           2      1.19
00007  CHARBON  5 SACSGS      2      0.69

. GOTO RECORD 4

. INSERT BEFORE

RECORD 00004

ARTICLE :  ASSIETTES PAPIER
NO      :  1
PRIX    :  .79

. LIST
00001  CONSERVES              5      0.69
00002  PAIN                   2      0.89
00003  BOEUF                  4      3.59
00004  ASSIETTES PAPIER       1      0.79
00005  SALADE                 2      0.49
00006  FROMAGE                1      1.79
00007  LAIT  (1/2 L)          2      1.19
00008  CHARBON                2      0.69

. 4

. DISPLAY
00004  ASSIETTES PAPIER       1      0.79

. INSERT BLANK
```

```
. LIST
00001  CONSERVES              5      0.69
00002  PAIN                   2      0.89
00003  BOEUF                  4      3.59
00004  ASSIETTES PAPIER       1      0.79
00005
00006  SALADE                 2      0.49
00007  FROMAGE                1      1.79
00008  LAIT                   2      1.19
00009  CHARBON                2      0.69

. 5

. REPLACE ARTICLE WITH 'FOURCHETTES' .AND. NO WITH 5 .AND. PRIX WITH .39

00001 REMPLACEMENT(S)

. LIST
00001  CONSERVES              5      0.69
00002  PAIN                   2      0.89
00003  BOEUF                  4      3.59
00004  ASSIETTES PAPIER       1      0.79
00005  FOURCHETTES            5      0.39
00006  SALADE                 2      0.49
00007  FROMAGE                1      1.79
00008  LAIT                   2      1.19
00009  CHARBON                2      0.69
```

JOIN

JOIN TO <fichier> FOR <expression> [FIELDS <liste champs>]

C'est l'une des plus puissantes commandes de dBASE. Elle permet d'assembler deux bases de données pour créer une troisième base selon certains critères.

Les deux bases de données utilisées sont les fichiers primaire et secondaire. La commande SELECT PRIMARY est émise en premier, ensuite s'exécute la commande JOIN. Cette dernière positionne un pointeur au premier enregistrement du fichier primaire et teste l'<expression> pour chaque enregistrement du fichier secondaire. Lorsque le test est positif, l'enregistrement est ajouté à la nouvelle base de données de

destination. Lorsque la fin du fichier secondaire est atteinte, le pointeur se positionne sur l'enregistrement suivant de la base primaire fichier avant de commencer une nouvelle scrutation du fichier secondaire. La même opération se répétera jusqu'au dernier enregistrement du fichier primaire.

Si la phrase FIELDS n'est pas utilisée, la nouvelle base de données comprend l'ensemble des champs des fichiers primaire et secondaire (ces champs sont limités à 32 par dBASE).

Si la phrase FIELDS est spécifiée, seuls les champs, et uniquement ceux spécifiés par liste, sont placés dans le nouveau fichier.

Si la dimension des bases de données primaire et secondaire est importante, l'exécution de cette commande sera très longue. La prudence est recommandée lors de l'utilisation de cette commande car une base de données ne peut pas contenir plus de 65.535 enregistrements.

Exemples :

```
. USE STOCK
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : STOCK.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP   DIM    DECIMALE(S)
001     ARTICLE      C    020
002     PRIX         N    010
003     PART:NO      C    005
004     DISPO        N    005
** TOTAL **            00041

. LIST
00001  MONTRE                9.99 24776      1
00002  PANIER                1.67 31415     18
00003  DIVERS               16.33 92653      7
00004  HORLOGE          134999.00 89793      3
00005  CUISINIERE           34.72 21828     77
00006  TROMBONES           198.37 76767     76
00007  ANNEAUX             200.00 70296      5
00008  CENDRIER             22.00 11528     16
```

```
. SELECT SECONDARY

. USE ORDRES
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : ORDRES.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP   DIM    DECIMALE(S)
001     CLIENT       C    020
002     PART:NO      C    005
003     MONTANT      N    005
** TOTAL **            00031

. LIST
00001  SWARTZ, JOE              31415    13
00002  SWARTZ, JOE              76767    13
00003  HARRIS, ARNOLD           11528    44
00004  ADAMS, JEAN              89793    12
00005  MACK, JAY                31415     3
00006  TERRY, HANS              76767     5
00007  JUAN, DON                21828     5
00008  SALT, CLARA              70296     9

. SELECT PRIMARY

. JOIN TO ANNOTATE FOR PART:NO=S.PART:NO;
 FIELD CLIENT,ARTICLE,MONTANT,PRIX

. USE ANNOTATE

. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : ANNOTATE.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYP   DIM    DECIMALE(S)
001     CLIENT       C    020
002     ARTICLE      C    020
003     MONTANT      N    005
004     PRIX         N    010     002
** TOTAL **            00056
```

```
. LIST
00001   SWARTZ, JOE          PANIER          13       1.67
00002   MACK, JAY            PANIER           3       1.67
00003   ADAMS, JEAN          HORLOGE         12  134999.00
00004   JUAN, DON            CUISINIERE       5      34.72
00005   SWATRZ, JOE          TROMBONES       13     198.37
00006   TERRY, HANS          TROMBONES        5     198.37
00007   SALT, CLARA          ANNEAUX          9     200.00
00008   HARRIS, ARNOLD       CENDRIER        44      22.00

. USE STOCK

. JOIN TO RETOUR FOR PART:NO=S.PART:NO.AND.DISPO<MONTANT;
 FIELD CLIENT;ARTICLE

. USE RETOUR
. LIST
00001   ADAMS, JEAN          HORLOGE
00002   SALT, CLARA          ANNEAUX
00003   HARRIS, ARNOLD       CENDRIER
```

LIST

a. LIST [<étendue>] [FOR <expression>] [<liste d'expressions>] [OFF]
[WHILE <expression>] [FIELD <liste de champs>]
b. LIST STRUCTURE
c. LIST MEMORY
d. LIST FILES [ON <unité de disque>] [LIKE <masque>]
e. LIST STATUS

Cette commande est semblable à la commande DISPLAY, à l'exception d'une pause après l'affichage d'un groupe de 15 enregistrements. Notez cependant que les commandes LIST STRUCTURE, LIST FILES, LIST MEMORY et LIST STATUS fonctionnent exactement comme la commande DISPLAY.

Si l'option SET DELETE ON est utilisée, seuls les enregistrements non repérés pour effacement seront affichés.

LOCATE

LOCATE [<étendue>] [FOR <exp>]
[CONTINUE]

Cette commande recherche, dans le fichier en cours d'utilisation, l'enregistrement dont les champs de données satisfont la condition stipulée dans l'<exp>ression. Lorsqu'un enregistrement est trouvé, le message suivant est visualisé :

```
ENREGISTREMENT n
```

La commande CONTINUE permet de poursuivre la recherche. Les différentes commandes de dBASE peuvent être utilisées entre LOCATE et CONTINUE.
Note : le nombre de caractères de l'<exp>ression est limité à 128 et non à 254. Voir CONTINUE.

Si aucun enregistrement ne correspond à la sélection, le message FIN DE FICHIER est visualisé et le pointeur positionné sur le dernier enregistrement. Si la clause NEXT (voir Etendue, Chapitre 9.1) est utilisée dans cette commande, et si aucun enregistrement n'est trouvé, le message FIN DE LOCALISATION est affiché et le pointeur est également positionné sur le dernier enregistrement.

Note : l'exécution de LOCATE est plus rapide sur un fichier non indexé.

Exemples :

```
. USE ACHAT
. LIST
00001  CONSERVES           5       0.69
00002  PAIN                2       0.89
00003  BOEUF               4       3.59
00004  ASSIETTES PAPIER    1       0.79
00005  FOURCHETTES         5       0.39
00006  SALADE              2       0.49
00007  FROMAGE             1       1.79
00008  LAIT (1 L)          2       1.19
```

```
. LOCATE FOR PRIX>.70
ENREGISTREMENT : 00002
. CONTINUE
ENREGISTREMENT : 00003

. DISP ARTICLE
BOEUF
. CONTINUE
ENREGISTREMENT : 00004

. CONTINUE
ENREGISTREMENT : 00007
. CONTINUE
ENREGISTREMENT : 00008

. CONTINUE
FIN DE FICHIER
```

LOOP

L'utilisation de cette commande au sein d'une boucle DO WHILE, permet d'ignorer toutes les commandes situées après l'instruction LOOP tout en permettant d'estimer et d'exécuter à nouveau le DO WHILE. LOOP permet de diminuer le temps d'exécution d'une boucle DO WHILE, en « sautant » toutes les lignes de commandes inutiles. La commande LOOP est semblable à la commande ENDDO.

L'utilisation de LOOP dans un DO WHILE n'est pas une bonne logique de programmation et est déconseillée.
L'un des deux exemples suivants utilise l'instruction LOOP.

Exemple 1 :

```
STORE 1 TO INDEX
DO WHILE INDEX<10
  STORE INDEX+1 TO INDEX
  IF ARTICLE=' '                Lorsque ARTICLE  est égal à espace
    SKIP                        passez à l'enregistrement suivant
    LOOP                        et revenez au DO WHILE
  ENDIF
  DO TRAIT
ENDDO
```

Exemple 2 :

```
STORE 1 TO INDEX
DO WHILE INDEX < 10
  STORE INDEX + 1 TO INDEX
  IF ARTICLE = '    '
    SKIP
  ELSE
    DO TRAIT
  ENDIF
ENDDO
```

MODIFY

a. MODIFY STRUCTURE
b. MODIFY COMMAND [<fichier de commande>]

Le cas « a. » permet à l'utilisateur de modifier la structure d'un fichier. Toutes les modifications sont autorisées. Des champs peuvent être créés et effacés. Les différents paramètres (noms, types, dimensions, nombre de décimales) peuvent également être modifiés.

MODIFY fonctionne sur la base de données en cours d'utilisation. Les corrections sont effectuées directement à l'écran de la même façon qu'avec l'édition plein écran, mais avec deux exceptions : ctrl-N insère une ligne blanche à l'endroit où se trouve le curseur, ctrl-T efface la ligne où se trouve le curseur. Les autres commandes de contrôle sont identiques à celles décrites au Chapitre 9.

NOTE : La commande MODIFY STRUCTURE efface tous les enregistrements du fichier concerné avant de procéder aux modifications. Pour conserver les données précédemment saisies, tout en modifiant la structure, copiez la structure initiale dans un fichier de travail. Effectuez vos corrections, puis, après la sauvegarde de cette nouvelle structure, ajoutez les données de la base initiale. Effacez la base initiale si celle-ci est devenue inutile et changez le nom de la base nouvellement créée si cela est nécessaire. Voir l'exemple ci-dessous.

Le cas « b. » permet l'édition plein écran des fichiers de commandes (ou de tout autre fichier « texte »). Si le fichier de commandes existe, son contenu est affiché et l'utilisateur peut alors effectuer d'éventuelles modifications. Dans le cas contraire, le fichier de commandes est créé et la saisie des différentes lignes de commande est autorisée. Après édition d'un fichier de commandes, MODIFY COMMAND crée deux fichiers; le fichier modifié aura l'extension « .CMD », le fichier initial (sans les dernières modifications) aura l'extension « .BAK ».

Les commandes ctrl-N et ctrl-T donnent les mêmes résultats que ceux définis dans le second paragraphe. Ctrl-Q permet de ne pas prendre en compte les modifications. Ctrl-W sauvegarde sur disque le fichier de commandes avec prise en compte des différentes modifications.

Le cas « b. » de cette commande est, malgré tout, restrictif :
1) les lignes ne peuvent contenir que 77 caractères de long ou moins (y compris le retour chariot)
2) la tabulation est convertie en espace simple
3) le curseur ne peut se déplacer que sur un maximum de 4 000 caractères
4) des blocs ne peuvent pas être recherchés ni déplacés comme le permettent certains éditeurs de texte.

A l'exception des commandes suivantes, les commandes de contrôle du curseur en mode plein écran sont les mêmes pour MODIFY COMMAND :

ctrl-N	insère une ligne blanche à l'endroit où se trouve le curseur
ctrl-T	efface la ligne où se trouve le curseur et remonte les autres lignes
ctrl-W	sauvegarde sur disque le fichier avec ses corrections et sort mode MODIFY COMMAND
ctrl-Q	abandonne toute correction faite dans le fichier de commandes
ctrl-R	défilement vers le bas
ctrl-C	défilement vers le haut

Exemple :

```
. NOTE  -- EXEMPLE DE MODIFICATION DE STRUCTURE SANS --
. NOTE   PERDRE LES INFORMATIONS PRECEDEMMENT SAISIES
. USE STOCK
. COPY TO TRAVAIL
. USE TRAVAIL
. MODIFY STRUCTURE
. APPEND FROM STOCK
. DELETE FILE STOCK
. USE
. RENAME TRAVAIL TO STOCK
```

NOTE

a.. NOTE tous caractères
b. * tous caractères

Cette commande permet de placer des commentaires dans un fichier de commandes. A la différence de la commande REMARK, les commentaires de NOTE ne seront pas émis vers l'imprimante.

Exemple :

```
NOTE - dernière modification : 4 juillet 1985
* -- dernière modification ce jour
```

PACK

Cette commande efface physiquement les enregistrements marqués par la commande DELETE. Les enregistrements effacés par la commande PACK ne peuvent plus être restitués. Ils sont définitivement détruits.

Si nous exécutons la commande PACK sur un fichier indexé, le fichier index est mis à jour simultanément. Dans le cas de fichier indexé de grande dimension, il est préférable d'utiliser PACK uniquement sur le fichier de données puis de mettre à jour le fichier index.

Si la base de données est indexée par plus d'un fichier index, les autres fichiers doivent être réindexés.

Une autre technique d'utilisation de PACK consiste à copier l'ancien fichier dans un nouveau : les enregistrements logiquement effacés ne seront pas transférés. L'ancien fichier peut alors être effacé (ou sauvegardé) et le nouveau fichier renommé.

Sous le système d'exploitation CP/M, il est préférable d'utiliser la procédure de copie plutôt que la commande PACK, car l'espace disque initialement occupé reste le même après l'exécution de la commande PACK. Ceci est une limite de CP/M et non pas de dBASE II.

Exemples :

```
. USE B:TRAVAIL
. LIST
00001  CONSERVES            5       0.75
00002  PAIN                 2       0.97
00003  BOEUF                4       3.94
00004  ASSIETTES PAPIER     1       0.86
00005  FOURCHETTES          5       0.42
00006  SALADE               2       0.53
00007  FROMAGE              1       1.96
00008  LAIT                 2       1.30

. DELETE RECORD 8
00001 ENREGISTREMENT(S) EFFACE(S)
```

```
. LIST
00001  CONSERVES              5       0.75
00002  PAIN                   2       0.97
00003  BOEUF                  4       3.94
00004  ASSIETTES PAPIER       1       0.86
00005  FOURCHETTES            5       0.42
00006  SALADE                 2       0.53
00007  FROMAGE                1       1.96
00008 *LAIT                   2       1.30

. PACK
COMPACTAGE TERMINE 00007 ENREGISTREMENTS TRANSFERES

. LIST
00001  CONSERVES              5       0.75
00002  PAIN                   2       0.97
00003  BOEUF                  4       3.94
00004  ASSIETTES PAPIER       1       0.86
00005  FOURCHETTES            5       0.42
00006  SALADE                 2       0.53
00007  FROMAGE                1       1.96
```

Il n'est pas toujours nécessaire d'exécuter un PACK. Nous pouvons effacer quelques enregistrements sans toutefois les enlever de la base de données. Ces enregistrements ne seront pas copiés, ajoutés ou triés; ils seront toutefois comptés. Il est donc important de savoir si l'enregistrement traité est logiquement effacé. L'exemple suivant est une partie d'un fichier de commandes qui ignore tous les fichiers logiquement effacés pour ne traiter que les enregistrements normaux.

```
DO WHILE .NOT. EOF
  LOCATE FOR NATURE = "TLM"
  IF .NOT. *
    .
    commandes
    .
  ENDIF
  CONTINUE
ENDDO
```

PEEK

PEEK(< adresse >)

Cette commande permet d'effectuer une lecture à l' < adresse > mémoire spécifiée. Une valeur entière comprise entre 0 et 255 est alors renvoyée vers la console ou vers une variable mémoire.

Exemple :

```
. ? PEEK(304)                          Longueur de la chaîne de caractères
  1                                    d'effacement de l'écran de dBASE II.

. ? PEEK(305)                          Premier et dernier (dans notre cas)
  12                                   caractère de la chaîne pour l'effa-
                                       cement de l'écran.
. STORE PEEK(305) TO CODE
  12
. ? CODE
        12
```

(Voir la commande POKE).

POKE

POKE < adresse > , < valeur >

Cette commande permet de placer une < valeur > entière, comprise entre 0 et 255, à une < adresse > désignée de la mémoire. Cette commande rend par exemple possibles les modifications temporaires de la configuration du programme dBASE II.

Exemple : Modification temporaire de la chaîne de caractères utilisée pour l'effacement de l'écran.
Chaîne de caractères à installer : 0CH → 12 en décimal

- POKE 304,1 (longueur de la chaîne)
- POKE 305,12 (premier et dernier caractère de la chaîne dans notre cas)
- ERASE

Si l'effacement de votre écran utilise également le code 12 en décimal, après l'exécution des trois lignes précédentes, l'écran sera vierge de tout caractère. Cette commande doit être employée avec de grandes précautions car une erreur dans l'<adresse> ou dans la <valeur> peut entraîner des erreurs système et éventuellement la perte d'informations du (ou des) fichier(s) de données en cours d'utilisation.

(Voir la commande PEEK).

QUIT

QUIT [TO <liste de programmes>]

Cette commande ferme tous les fichiers de base de données, les fichiers de commandes, les fichiers alternatifs, et rend le contrôle au système. Le message *** FIN DE TRAITEMENT dBASE II *** est visualisé.

Si la phrase TO est spécifiée, tous les programmes dans <liste de programmes> sont exécutés en séquence par CP/M. Ceci vous permet de sortir de dBASE et d'enchaîner avec d'autres programmes.

Les programmes ne sont pas limités en nombre et les commandes CP/M peuvent être exécutées tant que l'on n'excède pas la limite des 254 caractères autorisés.

Exemple :

```
. QUIT TO 'DIR B:','PIP PRN:=ALTERNAT.TXT','dBASE FICHIER'
```

Dans cet exemple, nous sortons de dBASE, listons les fichiers se trouvant sur l'unité de disque B, exécutons le programme PIP pour copier un fichier en sortie, et rappelons dBASE avec un fichier de commandes (FICHIER).

READ

Cette commande permet, en mode plein écran, la modification et/ou la saisie de variables identifiées et visualisées par la commande « @ » suivie de l'instruction GET. Le curseur peut se déplacer sur toutes les positions d'une zone saisie, et également d'une zone de saisie à une autre. Les modifications de ces zones de saisie à l'écran sont enregistrées dans les champs de base de données appropriés ou dans les variables mémoire.

Si l'instruction SET FORMAT TO <fichier format> est utilisée, la commande READ exécute toutes les commandes « @ » formatant l'écran et permet la modification ou la saisie des différentes zones précédées de GET. Cette technique est une substitution de la commande EDIT.

Lorsque vous êtes en mode SET FORMAT TO SCREEN, une commande ERASE efface l'écran. Une série de commandes « @ » peut alors être émise pour formater l'écran. La commande READ vous permet alors de saisir ou de modifier les différentes zones précédées de GET.

Si d'autres commandes « @ » sont émises après une commande READ, cette dernière place le curseur sur la première variable du GET suivant le dernier READ. De cette façon, la présentation d'écran et l'ordre de priorité de la modification ou de la saisie de variables spécifiques dépendent des conditions posées par l'utilisateur.

Les variables utilisées avec la commande « @ » et modifiées par la commande READ doivent être soit des noms de champs du fichier en cours d'utilisation, soit des variables mémoire. Les variables mémoire doivent être définies avant l'utilisation de la commande « @ ». Lors de l'initialisation d'une variable mémoire, affectez autant d'espaces que le nombre maximum de caractères que vous désirez saisir.
(exemple, STORE ' ' to MEMVAR).

Voir le Chapitre 8 pour les instructions de saisie de données et de gestion du curseur.

La commande SET SCREEN ON doit être en activité (c'est l'option par défaut définie lors de l'installation sur dBASE).

Exemple :

```
STORE ' ' TO PTYPE
STORE '           ' TO ACCT
ERASE
@ 5,0 SAY 'Entrez un C pour payment comptant'
@ 6,0 SAY '    ou un D pour payment différé'
@ 8,10 GET PTYPE
READ
IF PTYPE='D'
   @ 10,10 SAY 'Entrez no.compte' GET ACCT PICTURE '999-99-9999'
   READ
ENDIF
```

Dans cet exemple, l'écran est effacé et les deux premières commandes « @ » exécutées. Le curseur est alors positionné entre deux caractères « : », définissant la position de saisie à l'écran de la variable PTYPE. Puisque la première commande STORE initialise la taille de PTYPE à un caractère, la frappe de tout caractère remplira PTYPE exécutant ainsi la première commande READ.

Si l'opérateur avait saisi un « D », la commande « @ » aurait demandé un numéro de compte.

```
USE CHECKS
SET FORMAT TO SCREEN
ACCEPT "Option" TO CHOIX
IF CHOIX $ 'Aa'
  ERASE
  DO WHILE numero # 0
    APPEND BLANK
    @ 5,0 SAY "Entrez le num. suivant" ;
          GET numero PICTURE '99999'
    @ 6,0 SAY "Entrez le bénéficiaire";
          GET benefiT  PICTURE 'XXXXXXXXXXXXXXXXXXXXXXXXX'
    @ 7,0 SAY "Entrez le montant";
          GET montant PICTURE '9999999999'
    @ 8,5 SAY est-ce déjà de retour ? " ;
          GET maison
    @ 8,30 SAY "le débitez-vous ? ";
           GET débit
    READ
  ENDDO
ENDIF
```

RECALL

RECALL [<étendue>] [FOR <exp>]

Cette commande restitue les enregistrements marqués pour effacement logique par la commande DELETE.

Exemples :

```
. USE DUPE3
. LIST
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, RENE         2468
00003  CASSIDY, BERNARD      3344
00004  CHANG, LEE            6743
00005  POST, WILEY           1011
00006  LANCASTER, RENE       6623
```

```
. 3
. DELETE NEXT 3
00003  ENREGISTREMENT(S) EFFACE(S)

. LIST
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, RENE         2468
00003 *CASSIDY, BERNARD      3344
00004 *CHANG, LEE            6743
00005 *POST, WILEY           1011
00006  LANCASTER, RENE       6623

. RECALL RECORD 4
00001  ENREGISTREMENT(S) RESTITUE(S)
. LIST
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, RENE         2468
00003 *CASSIDY, BERNARD      3344
00004  CHANG, LEE            6743
00005 *POST, WILEY           1011
00006  LANCASTER, RENE       6623

. RECALL ALL
00002 ENREGISTREMENT(S) RESTITUE(S)

. LIST
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, RENE         2468
00003  CASSIDY, BERNARD      3344
00004  CHANG, LEE            6743
00005  POST, WILEY           1011
00006  LANCASTER, RENE       6623
```

REINDEX

Cette commande permet la remise à jour des fichiers index, suite aux éventuelles modifications effectuées sur le fichier de données.

Exemples :

```
. USE LIVRES INDEX AUTEURS
. DISP STRU
STRUCTURE DU FICHIER            : LIVRES.DBF
NOMBRE D'ENREGISTREMENTS        : 00009
DATE DE LA DERNIERE MISE A JOUR : 18/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM    DECIMALE(S)
001     TITRE        C    025
002     AUTEUR       C    015
003     CAT:NUM      C    006
004     ARR:DTE      C    008
** TOTAL **            00055

. LIST
00001  Comment voler           Bird, I. M.      IMB001   02/29/04
00005  Procedures              Bird, I. M.      IMB002   07/25/06
00002  Peches et Chasse        Fish, U. R.      URF001   10/30/23
00008  Sante                   Knight and Gale  KG001    08/04/44
00006  Vacances en Europe      Knight and Gale  KG002    06/24/42
00004  Comment cuisiner        Lynch, I.        IL001    04/01/00
00003  Comment survivre        Lynch, M.        ML001    01/01/30
00009  Nana                    Zola, E.         LS002    03/24/73
00007  L'argent                Zola, E.         LS001    12/01/73

. APPEND BLANK
. REPLACE AUTEUR WITH 'Lune, Bleu'
00001 REMPLACEMENT(S)
. REPLACE TITRE WITH 'Les astres'
00001 REMPLACEMENT(S)
. REPLACE CAT:NUM WITH 'BM001'
00001 REMPLACEMENT(S)
. REPLACE ARR:DTE WITH '15/11/54'
00001 REMPLACEMENT(S)
. DISPLAY
00010  Les astres              Lune, Bleu       BM001    15/11/54

. SET INDEX TO TITRES
```

```
. REINDEX
. SET INDEX TO CATNUMS
. REINDEX
. SET INDEX TO AUTEURS
```

RELEASE

RELEASE [< memvar liste >]
[ALL]

Cette commande détruit tout ou partie des variables mémoire, l'emplacement mémoire ainsi libéré étant disponible pour de nouvelles variables.

Si ALL est spécifié, l'ensemble des variables mémoire est détruit.

REMARK

REMARK tous caractères.

Cette commande permet de visualiser des commentaires. Le libellé de ce commentaire est émis vers le périphérique actif (écran de visualisation, imprimante).

Exemple :

```
 REMARK ****** ESSAI D'EXPLICATION *******
****** ESSAI D'EXPLICATION *******
```

RENAME

RENAME < nom du fichier original > TO < nom du nouveau fichier >

Cette commande permet de changer le nom d'un fichier du répertoire CP/M. Si aucune extension n'est précisée, DBASE II considère qu'il s'agit d'un fichier de données dBASE II et prend par défaut l'extension « .DBF ». Voir le Chapitre 4 pour les types de fichiers utilisés par dBASE II.

Il est déconseillé d'utiliser cette commande sur un fichier en cours d'utilisation.

Exemple :

```
. RENAME STOCKNOU TO STOCKANC

. RENAME D:ETAT.FRM TO ETAT.BAK

. RENAME TYPELESS. TO TYPED.TYPE
```

REPLACE

REPLACE [<étendue>] [champ] WITH <exp> [,<champ2> WITH <exp2>], etc
[FOR <exp>]

Cette commande remplace le contenu des champs de données du fichier en cours d'utilisation par de nouvelles données. A la différence de STORE, REPLACE modifie le contenu des champs de variables alors que la commande STORE permet uniquement la modification des variables mémoire.

Si <étendue> ne figure pas dans la commande, les remplacements sont effectués sur l'enregistrement en cours.

Si le fichier en cours d'utilisation met en œuvre un fichier index, et si le contenu de la zone de clé d'un enregistrement est modifié, la commande REPLACE permet la prise en compte de la nouvelle information dans le fichier de données et effectue simultanément la mise à jour du fichier index. Aucune modification n'est apportée au(x) fichier(s) index non utilisé(s). La clause « NEXT n », ne doit pas être utilisée lors du remplacement du contenu d'une zone de clé d'un fichier indexé, car le fichier index est mis à jour après chaque remplacement, ce qui entraîne que l'enregistrement suivant celui modifié, sera différent de celui avant modification.

Exemples :

```
. USE COURSE
. NOTE PRISE EN COMPTE D'UNE AUGMENTATION DE PRIX DE 10%
. LIST
00001  CONSERVES              5       0.69
00002  PAIN                   2       0.89
00003  BOEUF                  4       3.59
00004  ASSIETTES PAPIER       1       0.79
00005  FOURCHETTES            5       0.39
00006  SALADE                 2       0.49
00007  FROMAGE                1       1.79
00008  LAIT (1 L.)            2       1.19

. REPLACE ALL PRIX WITH PRIX*1.1
00008 REMPLACEMENT(S)
. LIST
00001  CONSERVES              5       0.75
00002  PAIN                   2       0.97
00003  BOEUF                  4       3.94
00004  ASSIETTES PAPIER       1       0.86
00005  FOURCHETTES            5       0.42
00006  SALADE                 2       0.53
00007  FROMAGE                1       1.96
00008  LAIT (1 L.)            2       1.30

. USE B:COURSE
. COPY TO B:TRAVAIL
00008  ENREGISTREMENT(S) TRANSFERE(S)
. LIST
00001  CONSERVES              5       0.75
00002  PAIN                   2       0.97
00003  BOEUF                  4       3.94
00004  ASSIETTES PAPIER       1       0.86
00005  FOURCHETTES            5       0.42
00006  SALADE                 2       0.53
00007  FROMAGE                1       1.96
00008  LAIT                   2       1.30
```

```
. GOTO TOP
. REPLACE NEXT 5 PRIX WITH PRIX*1.1 FOR PRIX>.75
00003  REMPLACEMENT(S)
. LIST
00001  CONSERVES                5       0.75
00002  PAIN                     2       1.06
00003  BOEUF                    4       4.33
00004  ASSIETTES PAPIER         1       0.94
00005  FOURCHETTES              5       0.42
00006  SALADE                   2       0.53
00007  FROMAGE                  1       1.96
00008  LAIT                     2       1.30

. USE CHECKS

. DISP STRU
STRUCTURE DU FICHIER             : CHECKS.DBF
NOMBRE D'ENREGISTREMMENTS        : 00016
DATE DE LA DERNIERE MISE A JOUR  : 18/10/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYPE   DIM   DECIMALE(S)
001     NUMERO        N     005
002     BENEFIT       C     020
003     MONTANT       N     010    002
004     DISPO         L     001
005     DEBIT         L     001
** TOTAL **              00038
```

```
. LIST
00001      1 P.T.T.                104.89 .F. .T.
00002      2 Gaz                     4.15 .F. .T.
00003      3 Electricité           250.30 .F. .T.
00004      4 Epicerie             1034.45 .F. .T.
00005    134 Personnel             561.77 .T. .F.
00006      6 Banque                  4.00 .T. .T.
00007      7 Médecin               100.00 .T. .T.
00008      8 Assurances            100.00 .F. .T.
00009      9 Garage                500.01 .F. .T.
00010     10 Personnel             561.77 .T. .F.
00011     11 Voiture                50.02 .F. .T.
00012     12 Personnel             561.77 .T. .F.
00013     13 Personnel             750.03 .T. .F.
00014    234 Hotel                  14.00 .F. .T.
00015    237 Restaurant            650.00 .F. .T.
00016     30 Personnel             561.77 .T. .F.

. 11
. REPLACE DISPO  WITH T
00001 REMPLACEMENT(S)
. DISPLAY
00011      11 Voiture                50.02 .T. .T.
```

REPORT

REPORT [FORM <fichier format> [<étendue>] [FOR <exp>] [TO PRINT]

REPORT génère les états (à l'écran ou sur papier), des données du fichier en cours d'utilisation, suivant la présentation de votre choix. La présentation des états est composée d'un en-tête, de différentes colonnes avec éventuellement un titre, des totaux (et sous-totaux) de champs numériques.

Les données imprimées peuvent être :

— le contenu de champs du fichier ou de variables mémoire
— le résultat d'un calcul arithmétique sur des valeurs numériques
— la concaténation de chaînes de caractères
— une constante numérique
— une constante chaîne de caractère(s)

Caractéristiques d'utilisation :

— Chaque ligne de l'état peut être composée d'un maximum de 132 caractères.
— Un maximum de 24 champs de données peuvent être imprimés simultanément au sein d'un même état.
— Le nombre de champs « TOTAL » ou « SOUS-TOTAL » est limité à 16, des erreurs peuvent être rencontrées si vous dépassez cette limite.

La phrase FOR permet d'imprimer uniquement les informations répondant aux conditions définies dans <exp>. La phrase TO PRINT permet l'émission de l'état vers l'imprimante (en plus de l'écran). ALL est la valeur par défaut de <étendue>.

Lorsque vous utilisez la commande REPORT pour créer un nouvel état, un fichier format est créé. La création est effectuée en conversationnel, une suite de questions est soumise à l'utilisateur afin de définir l'en-tête, la position la dimension et le contenu des différentes colonnes, les titres de chaque colonne, etc (voir l'exemple ci-dessous).

Lorsque le <fichier format> existe, la ligne de commande REPORT FORM <fichier format>, entraîne immédiatement l'impression des données suivant la présentation précédemment définie. Si seule la commande REPORT est émise, le message « DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : » sera affiché. Après la saisie du nom du <fichier format>, l'impression sera exécutée.

L'exemple suivant vous présente pratiquement la totalité des options de définition d'un <fichier format>.

L'utilisateur peut modifier :

— la marge gauche (option 'M', 8 espaces par défaut).
— le nombre de lignes par page (option 'L', 57 lignes par défaut).
— la largeur de la page (option 'W', 80 caractères par défaut, cette précision est prise en compte uniquement pour le centrage des titres).

Exemple :

```
. REPORT FORM ACHAT
OPTIONS  M=MARGE GAUCHE,L=LIGNES/PAGE,W=DIMENSION DE PAGE M=5,W=65
DESIREZ-VOUS UN TITRE (Y/N) ? Y
DONNEZ LE TITRE : liste des achats pour le Pique-nique
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? N
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
DESIREZ-VOUS DES SOUS-TOTAUX (Y/N) ?  N
COLONNE  LONGUEUR  CONTENU
001      23,ARTICLE+'...'
DONNEZ LE TITRE : ARTICLE;======
002      10,NO
DONNEZ LE TITRE : >NOMBRE;======
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
003      13,PRIX
DONNEZ LE TITRE : >PRIX/ARTICLE;============
DESIREZ-VOUS DES TOTAUX (Y/N) ? N
004      10,NO*PRIX
DONNEZ LE TITRE : >PRIX;====
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
005      (cr)
```

Vous devez définir la dimension ainsi que le contenu de chaque colonne. La dimension qui vous est demandée n'a aucune relation avec la dimension réelle du champ. Par exemple, la dimension de la colonne ARTICLE est de 23 caractères, alors que le champ ARTICLE du fichier de données est composé de seulement 20 caractères. Notez que la première colonne est la concaténation du champ ARTICLE et de la chaîne '...'. Si le nombre de caractères à imprimer dans une colonne est supérieur à la dimension de celle-ci, les caractères en excédent sont placés à la ligne suivante.

La colonne 4 est un exemple de réalisation de calcul arithmétique. Lors de l'impression, le produit des champs NO et PRIX est effectué pour chaque enregistrement. Comme vous pouvez le constater ci-dessous, aucun des champs du fichier de données ne correspond à ce calcul.

```
. LIST
00001   CONSERVES           5       0.75
00002   PAIN                2       1.06
00003   BOEUF               4       4.33
00004   ASSIETTES PAPIER    1       0.94
00005   FOURCHETTES         5       0.42
00006   SALADE              2       0.53
00007   FROMAGE             1       1.96
00008   LAIT                2       1.30
```

Vous avez certainement remarqué l'utilisation de certains caractères spéciaux (« ; », « < », « > »), lors de la définition des titres de chaque colonne du <fichier format>.

— « ; » permet d'imprimer les caractères à droite du « ; » sous le titre de la colonne.
— « < » permet de cadrer à gauche le titre de la colonne.
— « > » permet de cadrer à droite le titre de la colonne.

Dans le cas où aucune spécification n'est fournie, le titre est centré en fonction de la dimension de la colonne concernée. Si la dimension d'un titre excède celle de la colonne, les caractères en excédent (après un espace, s'il en existe un), sont placés à la ligne suivante.

Des totaux (et sous-totaux) peuvent être effectués. Ils permettent la totalisation des zones numériques (issues du fichier de données ou du résultat d'un calcul). Si vous utilisez les fonctions de totalisation (totaux et de sous-totaux), vous avez la possibilité de n'imprimer qu'un résumé. Un état résumé imprime uniquement les totaux et les sous-totaux.

Un retour chariot termine la définition des différentes colonnes. L'impression commence immédiatement après la fin de définition. Si la phrase TO PRINT est incluse dans la commande initiale, l'impression s'effectue simultanément (en plus de l'écran) sur l'imprimante.

Les commandes « SET EJECT OFF », « SET HEADING TO » et « SET DATE TO » permettent de modifier certains paramètres de défaut. La commande SET EJECT OFF permet de ne pas effectuer de saut de page (option de défaut) avant la génération d'un état sur l'imprimante. La commande SET HEADING TO permet de définir un titre supplémentaire avant la génération d'un état. Cette commande n'a d'effet que pour l'état qui lui est consécutif, si elle est nécessaire, elle doit être précisée avant chaque état. Il en est de même pour la commande SET DATE TO. L'utilisation de cette commande

modifie la date en tête de l'état. Consultez la commande SET pour un complément d'informations.

Les commandes « @ » et SET FORMAT TO PRINT vous apportent une grande souplesse pour générer des états avec une présentation particulière. Voir la commande « @ ».

Exemple 1 :

```
. USE COURSE
. REPORT FORM ACHAT


 PAGE NO. 00001

          liste des achats pour le Pique-nique

        ARTICLE                NOMBRE  PRIX/ARTICLE        PRIX
        =======                ======  ============        ====

   CONSERVES            ...         5          0.75        3.75
   PAIN                 ...         2          1.06        2.12
   BOEUF                ...         4          4.33       17.32
   ASSIETTES PAPIER     ...         1          0.94        0.94
   FOURCHETTES          ...         5          0.42        2.10
   SALADE               ...         2          0.53        1.06
   FROMAGE              ...         1          1.96        1.96
   LAIT                 ...         2          1.30        2.60
   ** TOTAL **
                                   22                     31.85
 SET HEADING TO 14 Juillet 1985
. REPORT FORM ACHAT
```

PAGE NO. 00001 14 Juillet 1985

liste des achats pour le Pique-nique

ARTICLE		NOMBRE	PRIX/ARTICLE	PRIX
CONSERVES	...	5	0.75	3.75
PAIN	...	2	1.06	2.12
BOEUF	...	4	4.33	17.32
ASSIETTES PAPIER	...	1	0.94	0.94
FOURCHETTES	...	5	0.42	2.10
SALADE	...	2	0.53	1.06
FROMAGE	...	1	1.96	1.96
LAIT	...	2	1.30	2.60
** TOTAL **		22		31.85

Exemple 2 :

```
. USE ORDRES INDEX ORDRES
. LIST
00003  HARRIS, ARNOLD        11528    44
00013  ANDERSON, JEAN  REGI  11528    16
00007  JUAN, DON             21828     5
00001  SWARTZ, JOE           31415    13
00005  MACK, JAY             31415     3
00009  BARNETT, PIERRE       31415     6
00008  SALT, CLATA           70296     9
00002  SWARTZ, JOE           76767    13
00006  TERRY, HANS           76767     5
00010  NICHOLS, BILL         76767    17
00004  ADAMS, JEAN           89793    12
00011  MURRAY, CAROLINE      89793     4
00012  WARD, CHARLES A.      92653    15
```

```
. REPORT
DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : ORDRES
OPTIONS, M=MARGE GAUCHE, L=LIGNES/PAGE, W=DIMENSION DE PAGE  M=5,W=65
DESIREZ-VOUS UN TITRE (Y/N) ? Y
DONNEZ LE TITRE : LISTE DES COMMANDES PAR NUMERO DE REFERENCE
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? N
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
DESIREZ-VOUS DES SOUS-TOTAUX (Y/N) ? Y
DONNEZ LE CHAMP DE SOUS-TOTAL : PART:NO
DESIREZ-VOUS UNIQUEMENT UN RESUME DU RAPPORT (Y/N) ? N
CHANGEMENT DE PAGE APRES LES SOUS-TOTAUX (Y/N) ? N
ENTREZ  LE TITRE POUR LES SOUS-TOTAUX :  Cde(s) pour  la référence
COLONNE  LONGUEUR  CONTENU
001      20,CLIENT
DONNEZ LE TITRE : <NOM DU CLIENT
002      10,MONTANT
DONNEZ LE TITRE : >QUANTITE COMMANDEE
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
003      <enter>


    PAGE NO. 00001

    LISTE DES COMMANDES PAR NUMERO DE REFERENCE

    NOM DU CLIENT           QUANTITE
                            COMMANDEE

    * Cde(s) pour  la référence 11528
    HARRIS, ARNOLD               44
    ANDERSON, JEAN  REGI         16
    ** SOUS-TOTAL **
                                 60

    * Cde(s) pour  la référence 21828
    JUAN, DON                     5
    ** SOUS-TOTAL **
                                  5
```

```
* Cde(s) pour  la référence 31415
SWARTZ, JOE                  13
MACK, JAY                     3
BARNETT, PIERRE               6
** SOUS-TOTAL **
                             22


* Cde(s) pour  la référence 70296
SALT, CLARA                   9
** SOUS-TOTAL **
                              9

* Cde(s) pour  la référence 76767
SWARTZ, JOE                  13
TERRY, HANS                   5
NICHOLS, BILL                17
** SOUS-TOTAL **
                             35


* Cde(s) pour  la référence 89793
ADAMS, JEAN                  12
MURRAY, CAROLINE              4
** SOUS-TOTAL **
                             16


* Cde(s) pour  la référence 92653
WARD, CHARLES A.             15
** SOUS-TOTAL **
                             15

** TOTAL **
                            162
```

Exemple 3 :

```
. DISP STRU
STRUCTURE DU FICHIER              : CARTES.DBF
NOMBRE D'ENREGISTREMENTS          : 00016
DATE DE LA DERNIERE MISE A JOUR : 17/09/85
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE   DIM   DECIMALE(S)
001     DATE          C    008
002     LISA          N    003
003     ANNA          N    003
004     PASCAL        N    003
** TOTAL **              00018

. REPORT
DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : CARTE
OPTIONS, M=MARGE GAUCHE, L=LIGNES/PAGE, W=DIMENSION DE PAGE M=5,W=40
DESIREZ-VOUS UN TITRE (Y/N) ? Y
DONNEZ LE TITRE : Tournoi de jeu de cartes
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? N
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
DESIREZ-VOUS DES SOUS-TOTAUX (Y/N) ? N
COLONNE  LONGUEUR,CONTENU
001     10,DATE
DONNEZ LE TITRE : Date du;Jeu
002     6,LISA
DONNEZ LE TITRE :  Point;Lisa
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
003     6,ANNA
DONNEZ LE TITRE  : Point;Anna
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
004     6,PASCAL
DESIREZ-VOUS DES TOTAUX : Point;Pascal
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
005     5,LISA+ANNA+PASCAL
DONNEZ LE TITRE : Jeu;Total
DESIREZ-VOUS DES TOTAUX (Y/N) ? Y
006     <enter>
```

Note : la colonne 5 de l'état ci-dessus est un nouvel exemple d'utilisation d'expression arithmétique. Dans notre cas, la somme des points des différents joueurs est calculée lors de l'impression de chaque enregistrement.

```
PAGE NO. 00001

      Tournoi de jeu de cartes

Date du   Point  Point Point   Jeu
  Jeu      Lisa   Anna Pascal Total

26/05/85     29     75     53   157
27/05/85     45     48     63   156
28/05/85     50     56     74   180
29/05/85     86     24     72   182
05/06/85     43     12     75   130
12/06/85     42      9     27    78
26/06/85     84     35     63   182
06/07/85     33     71     26   130
19/08/85     37     55     38   130
15/09/85     19     57     54   130
16/09/85     15      7    108   130
17/09/85     59     13     58   130
** TOTAL **
            715    698    875  2288
```

Vous pouvez visualiser uniquement une partie du fichier de données, comme ci-dessous.

```
. GOTO RECORD 7
. REPORT NEXT 4 FORM CARTE

PAGE NO. 00001

      Tournoi de jeu de cartes

Date du   Point  Point Point   Jeu
  Jeu      Lisa   Anna Pascal Total

26/06/85     84     35     63   182
06/07/85     33     71     26   130
19/08/85     37     55     38   130
15/09/85     19     57     54   130
** TOTAL **
            173    218    181   572
```

Vous pouvez également obtenir un état sélectif.

```
. REPORT FORM CARTE FOR PASCAL < 50

PAGE NO. 00001

      Tournoi de jeu de cartes

Date du   Point  Point Point   Jeu
  Jeu      Lisa   Anna Pascal Total

12/06/85     42      9     27    78
06/07/85     33     71     26   130
19/08/85     37     55     38   130
** TOTAL **
            112    135     91   338
```

RESET

Cette commande est utilisée pour réinitialiser le système après le remplacement d'une disquette. Dans le cas du remplacement d'une disquette, CP/M n'autorise l'écriture sur celle-ci que si une réinitialisation est effectuée (CTRL C sous CP/M). RESET essaie d'ouvrir à nouveau tous les fichiers ouverts avant le changement de disquette. Si un fichier précédemment ouvert ne se trouve pas sur la nouvelle unité de disquette active, RESET ferme le fichier.

ATTENTION : Si un remplacement est effectué et qu'un fichier précédemment ouvert existe également (même nom) sur le nouveau support magnétique, la commande RESET ne ferme pas ce fichier. Afin d'éviter toute erreur, il est recommandé de fermer tous les fichiers avant d'émettre la commande RESET après le remplacement d'un disque souple. La commande USE sans spécification de nom de fichier ferme le fichier en cours d'utilisation. Une commande CANCEL ferme tous les fichiers de commandes éventuellement ouverts.

Une commande RESET émise sans changement de disque sera sans effet.

RESTORE

RESTORE FROM <fichier>

Cette commande effectue la lecture d'un fichier de variables mémoire. Ce fichier doit être préalablement créé en utilisant la commande SAVE MEMORY TO <fichier>. Toutes les variables mémoire existantes sont détruites lors de l'exécution de la commande RESTORE.

Exemples :

```
. DISPLAY MEMORY
UNE         (N)    1.0000
ALPHABET    (C)    ABCDEFGHIJKL
CHARS       (C)    ABCDEFGHIJKL
** TOTAL **        03 VARIABLES USED  00030 BYTES USED

. SAVE TO MEMFICHE

. RELEASE ALL

. DISPLAY MEMORY
** TOTAL **        00 VARIABLES USED  00000 BYTES USED

. RESTORE FROM MEMFICHE

. DISPLAY MEMORY
UNE         (N)    1.0000
ALPHABET    (C)    ABCDEFGHIJKL
CHARS       (C)    ABCDEFGHIJKL
** TOTAL **        03 VARIABLES USED  00030 BYTES USED
```

RETURN

Cette commande, dans un fichier de commandes, permet le renvoi au fichier de commandes appelant. La commande RETURN est équivalente à la rencontre de la fin d'un fichier de commandes.

Les fichiers de commandes ont habituellement une commande RETURN en dernière ligne de programme.

SAVE

SAVE TO <fichier>

Cette commande sauvegarde sur disque toutes les variables mémoire existantes lors de son exécution. Ces variables mémoire peuvent ensuite être restituées par la commande RESTORE.

Exemples :

```
. DISPLAY MEMORY
UNE         (N)    1.0000
ALFABET     (C)   ABCDEFGHIJKL
CHARS       (C)   ABCDEFGHIJKL
** TOTAL **        03 VARIABLES USED  00030 BYTES USED

. SAVE TO MEMFICHE

. RELEASE ALL

. DISPLAY MEMORY
** TOTAL **        00 VARIABLES USED  00000 BYTES USED

. RESTORE FROM MEMFICHE

. DISPLAY MEMORY
UNE         (N)    1.0000
ALFABET     (C)   ABCDEFGHIJKL
CHARS       (C)   ABCDEFGHIJKL
** TOTAL **        03 VARIABLES USED  00030 BYTES USED
```

SELECT

SELECT [PRIMARY]
[SECONDARY]

Cette commande affecte l'une des deux zones mémoire disponibles à un fichier de données. L'utilisateur peut ainsi réaliser des traitements interactifs (lecture, écriture) sur deux fichiers de données.

Si aucune spécification particulière n'est utilisée, seule la zone primaire est active. La zone secondaire sera active dès l'émission de la commande SELECT SECONDARY et le restera tant que SELECT PRIMARY ne sera pas rencontrée. Sélectionner une zone déjà active est sans effet.

Lorsque chacune des deux zones est affectée à un fichier de données, les champs de variables de l'une ou de l'autre zone peuvent être exploités simultanément. Si deux fichiers possèdent un champ de même nom, il est nécessaire, afin d'éviter toute erreur, de faire précéder ce nom par « P. » si le champ concerné est celui de la zone primaire, par « S. » pour la zone secondaire.

Les commandes GOTO, SKIP, REPORT, SORT, COPY, LIST et DISPLAY, effectuent uniquement leur traitement sur les données du fichier sélectionné. La commande SET LINKAGE ON permet aux commandes séquentielles (commandes contenant des paramètres <étendue>) de se positionner sur les bases de données secondaires et primaires (voir commande SET). La commande REPLACE affecte uniquement les variables de la base de données en cours. La commande DISPLAY STRUCTURE visualise uniquement la structure de la base de données en cours.

Exemples :

```
. USE COURSE

. LIST
00001  CONSERVES            5       0.75
00002  PAIN                 2       1.06
00003  BOEUF                4       4.33
00004  ASSIETTES PAPIER     1       0.94
00005  FOURCHETTES          5       0.42
00006  SALADE               2       0.53
00007  FROMAGE              1       1.96
00008  LAIT                 2       1.30
```

```
. NOTE OUVRONS UN FICHIER DE DONNEES DANS LA ZONE SECONDAIRE

. SELECT SECONDARY

. USE TARIFS

. LIST
00001   800104      31.38
00002   800111      45.69
00003   800118      51.18
00004   800124      48.19
00005   800201      55.82
00006   800209      12.04
00007   800229      12.04

. SELECT PRIMARY

. SUM PRIX
 11.29

. SELECT SECONDARY

. APPEND

RECORD 00008
DATE       : 800303
MONTANT    : 11.29

RECORD 00009
DATE       : (cr)

. SUM MONTANT
 267.63

. NOTE NOUS POUVONS ACCEDER AUX VARIABLES DES DEUX BASES

. DISP OFF PRIX,MONTANT,ARTICLE,DATE
1.30  11.29  lait                 800303

. NOTE LE MEME FICHIER PEUT ETRE UTILISE DANS LES 2 ZONES
```

```
. USE COURSE

. NOTE SOYONS PRUDENTS SI DES NOMS DE CHAMPS SONT COMMUNS
. NOTE AUX DEUX FICHIERS DE DONNEES
```

SET

a. SET <param1> [ON]
[OFF]
b. SET <param2> TO <option>

Cette commande modifie la configuration de dBASE. La commande SET offre deux possibilités. Utilisée comme dans le cas « a. », elle rend active ou inactive l'option définie dans <param1>. Dans le cas « b. », elle définie l'<option> à utiliser pour les <param2>.

Les valeurs par défaut sont en gras :

<param1>	Fonction	Signification
ECHO	ON	Toutes les commandes issues d'un fichier de commandes sont visualisées à l'écran.
	OFF	Pas de visualisation.
STEP	ON	Après exécution d'une commande, dBASE arrête le programme et attend la décision de l'utilisateur : poursuivre l'exécution de la commande suivante, quitter (touche escape) le fichier de commandes, ou saisir une nouvelle commande au clavier. (STEP est utilisée pour localiser les erreurs dans un fichier de commandes).
	OFF	Exécution continue des opérations.
TALK	**ON**	Visualise les résultats des commandes à l'écran.
	OFF	Pas de visualisation.
PRINT	ON	Tous les caractères émis, sont envoyés vers l'imprimante.
	OFF	Pas de listage.

CONSOLE	**ON**	Tous les caractères émis, sont envoyés vers l'écran de visualisation.
	OFF	Pas de visualisation.
ALTERNATE	ON	Tous les caractères émis, sont stockés sur un fichier disque.
	OFF	Pas de sauvegarde.
SCREEN	**ON**	Active les opérations plein écran pour les commandes APPEND, INSERT, EDIT et CREATE.
	OFF	Pas d'opération plein écran.
LINKAGE	ON	Permet aux commandes séquentielles (LIST, REPORT, SUM et les commandes utilisant l'option <étendue>) de se positionner sur les fichiers de données primaires et secondaires.
	OFF	Les bases de données primaires et secondaires sont indépendantes.
COLON	**ON**	Les zones de saisie de l'instruction GET sont délimitées par « :'.
	OFF	Les « : » sont omis.
BELL	**ON**	Emet un « bip » sonore lorsqu'une zone de saisie est remplie, ou si des caractères interdits sont saisis.
	OFF	Pas de signal sonore.
ESCAPE	**ON**	Un caractère escape (1B en hexa) abandonne l'exécution du fichier de commandes.
	OFF	Pas d'abandon possible.

EXACT	ON	Exige la complète similitude de deux chaînes (sauf les espaces à droite) lors d'une comparaison ou d'une recherche par l'instruction FIND.
	OFF	Les chaînes peuvent être de longueurs différentes. Exemple : « ABCDEF » = « ABC » est vrai.
INTENSITY	**ON**	Le mode plein écran utilise les attributs vidéo (vidéo inversée, demi-intensité, couleur).
	OFF	Les attributs vidéo ne sont pas utilisés.
DEBUG	ON	Tous les messages générés par les commandes ECHO et STEP vers l'imprimante, car lors de l'exécution pas à pas d'un fichier de commande utilisant les fonctions plein écran il est très difficile de suivre l'évolution de l'application (effacement d'écran, superposition de messages, etc.).
	OFF	Pas d'émission vers l'imprimante.
CARRY	ON	Transfère les données de l'enregistrement précédent dans le nouvel enregistrement lorsque vous ajoutez des enregistrements en mode plein écran.
	OFF	Pas de transfert.
CONFIRM	ON	En mode Edition plein écran, la saisie de chaque champ doit se terminer par un retour chariot, même si toutes les positions de saisie, d'un champ, sont remplies.
	OFF	Passage automatique au champ suivant, lorsque toutes les positions du champ en cours de saisie sont remplies.
EJECT	**ON**	Un saut de page est effectué avant la génération d'un état sur l'imprimante (commande REPORT).
	OFF	Suppression du saut de page.

RAW	ON	Lors de l'utilisation des commandes LIST et DISPLAY, l'espace séparant deux champs d'un même enregistrement est ignoré sauf le nom du ou des champs à visualiser est précisé.
	OFF	Un espace sépare deux champs d'un même enregistrement visualisé par LIST ou DISPLAY.
DELETE	ON	Les enregistrements marqués pour effacement logique (commande DELETE) sont ignorés par la commande FIND et celles pouvant utiliser l'<étendue> NEXT (LIST, LOCATE, COUNT etc.).
	OFF	Les enregistrements marqués pour effacement logique peuvent être localisés et visualisés (mais ni copiés ou ajoutés).

SET HEADING TO <chaîne>

Cette commande imprime une <chaîne> de 60 caractères au maximum sur la ligne d'en-tête d'un état. (Voir la commande REPORT).

SET FORMAT TO [SCREEN]
[PRINT]
[<fichier de format>]

Les deux premières options définissent le périphérique vers lequel doivent être émises toutes les données précédées de l'instruction «@». La dernière option indique le nom du <fichier de format> contenant la présentation à utiliser. (Voir les commandes « @ » et READ).

SET DEFAULT TO <unité de disque>

Cette commande définit l'unité de disque à utiliser par défaut. Tous les fichiers (index, de commandes, de données, etc.), pour lesquels aucune unité de disque n'est précisée, sont lus ou modifiés sur l'unité active précédemment définie. Cette commande peut être mise en œuvre par le biais d'une macro-instruction.

Le paramètre <unité de disque> peut être libellé sous deux formes :
1. « B »
2. « B : »

NOTE : La commande SET n'affecte en aucune façon l'unité de disque par défaut, reconnu par le système d'exploitation. La commande SET DEFAULT est exclusivement utilisée par dBASE II.

Exemple :

```
. SET DEFAULT TO B:

. USE DATEVSYR      (Le fichier de données DATEVSYR est lu sur l'unité "B")
```

SET ALTERNATE TO [<fichier>]

Cette commande permet d'écrire sur le disque toutes les informations qui, en temps normal, sont affichées à l'écran (listes, messages d'erreur, caractères saisis au clavier, etc.). Si le <fichier> existe, son ancien contenu est détruit ; dans le cas contraire, il sera créé. Une commande SET ALTERNATE ON lance le traitement.

Exemple :

```
SET ALTERNATE TO B:DOCUMENT
SET ALTERNATE ON
  .
  .
  .
toutes commandes
  .
  .
  .
SET ALTERNATE TO n'importe-quel-fichier
```

Dans cet exemple, toutes les informations émises vers l'écran ou l'imprimante sont stockées dans le fichier B :DOCUMENT.TXT. Lorsque l'extension du fichier de destination n'est pas précisée, « .TXT » est utilisée par défaut. Ce fichier peut ensuite être imprimé ou traité par un progiciel de traitement de texte.

SET DATE TO jj/mm/aa

Le format peut être aa/mm/jj, mm/jj/aa, etc. « m », « j » ou « a » peuvent être n'importe quel nombre. Cette commande permet d'initialiser la date à n'importe quel moment. Aucun contrôle de validité n'est effectué.

```
SET DATE TO 12,10,76
```

SET INDEX TO <index principal> [,<fichier index>,..<fichier index>]

Cette commande peut identifier et créer jusqu'à sept fichiers index. Si un fichier index est en cours d'utilisation lors de l'émission de cette commande, l'ancien fichier index est alors fermé et le nouvel index principal est utilisé.

Note : Quand un nouveau fichier index est utilisé, le pointeur d'enregistrement reste à sa position initiale. Il est donc nécessaire d'utiliser une commande FIND ou GOTO, avant tout traitement (ex : NEXT), afin que le pointeur soit en relation avec le nouveau fichier index. Le premier fichier index cité est considéré comme l'index principal.

Une commande SET INDEX TO (sans spécification de nom de fichier index) ferme tous les fichiers index. Les enregistrements du fichier en cours d'utilisation sont alors traités séquentiellement.

SET MARGIN TO n

Cette commande permet à l'utilisateur de positionner la marge gauche lors de l'impression d'un état. L'impression de chaque ligne commencera à la colonne « n » + 1. « n » doit être un entier compris entre 1 et 254.

SKIP

SKIP [+][<exp>]
[—]

Cette commande déplace le pointeur sur un enregistrement situé avant ou après l'enregistrement en cours de traitement.

Exemple :

```
. USE STOCK

. LIST
00001  136928  13  1673 ADJ. INTERIM      7.13 189    9    0    9.98
00002  221679   9  1673 SM. COUTURE       5.17 173    4    1    7.98
00003  234561   0    96 PLASTIQUE         2.18 27   112   53    4.75
00004  556178   2   873 ADJ. INTERIM     22.19 117    3    0   28.50
00005  723756  73    27 COMPTEUR         19.56 354    6    1   29.66
00006  745336  13    27 PERCEUSE         12.65 63     7    2   15.95
00007  812763   2  1673 GLOBE            5.88 112    5    2    7.49
00008  876512   2   873 PRISE            3.18 45     7    3    4.25
00009  915332   2  1673 SUPPORT          1.32 97     7    3    1.98

. 5

. SKIP -2
ENREGISTREMENT : 00003

. SKIP
ENREGISTREMENT : 00004

. SKIP 3
ENREGISTREMENT : 00007
```

SORT

SORT ON <champ> TO <fichier> [ASCENDING]
[DESCENDING]

Cette commande permet à l'utilisateur de copier dans le <fichier> de destination les données du fichier en cours d'utilisation en organisant les enregistrements en fonction de l'évolution logique d'un champ. Cet ordre logique peut être ascendant ou descendant. Aucune modification n'est effectuée sur le fichier d'origine (fichier en cours d'utilisation).

Les enregistrements logiquement effacés sont ignorés. La commande SORT n'effectue des tris que sur une seule clé. Cependant vous pourrez réaliser des tris multicritères en effectuant des tris successifs. Commencez votre tri sur la clé la moins significative, puis

terminez par la clé principale. dBASE ne changera l'ordre des enregistrements que si nécessaire. Si nous ne spécifions pas l'ordre ascendant ou descendant, le tri se fera par ordre ascendant.

Le tri utilise la valeur ASCII de chaque caractère. Par exemple : la chaîne 'DURAND' est « plus petite » que la chaîne 'Durand'.

Il est nécessaire de noter que la commande INDEX permet d'obtenir des résultats semblables à ceux de la commande SORT, en bénéficiant d'une plus grande souplesse et d'un temps de réalisation bien inférieur. Lors de l'utilisation de SORT, l'emplacement disponible sur disque doit être au moins le même que celui occupé par le fichier d'origine.

Exemple :

```
. USE COURSE

. LIST
00001  CONSERVES            5       0.75
00002  PAIN                 2       0.97
00003  BOEUF                4       3.94
00004  ASSIETTES PAPIER     1       0.86
00005  FOURCHETTES          5       0.42
00006  SALADE               2       0.53
00007  FROMAGE              1       1.96
00008  LAIT (1 L.)          2       1.30

. SORT ON ARTICLE TO TRIFICHE
*** TRI TERMINE ***

. USE TRIFICHE
. LIST
00004  ASSIETTES PAPIER     1       0.86
00003  BOEUF                4       3.94
00001  CONSERVES            5       0.75
00005  FOURCHETTES          5       0.42
00007  FROMAGE              1       1.96
00008  LAIT                 2       1.30
00002  PAIN                 2       0.97
00006  SALADE               2       0.53
```

STORE

STORE <exp> TO <memvar>

Cette commande calcule la valeur d'une expression et stocke cette valeur dans une variable mémoire. Si cette variable n'existe pas avant l'émission de cette commande, celle-ci est automatiquement créée.

Notez que la commande STORE modifie uniquement les variables mémoire. Le contenu des champs d'un fichier de données est modifié par la commande REPLACE.

Exemple :

```
.
. RELEASE ALL

. STORE 1 TO UN
  1

. STORE 'ABCDEFGHIJKL' TO ALPHABET
ABCDEFGHIJKL

. STORE ALPHABET+' ENVIRON' TO CHAINE
ABCEDFGHIJKL ENVIRON

. STORE UN*1.0000 TO UN
 1.0000

. DISPLAY MEMORY
EOF         (L)  .T.
UN          (N)  1.0000
ALPHABET    (C)  ABCDEFGHIJKL
CHAINE      (C)  ABCDEFGHIJKL ENVIRON
** TOTAL **       04 VARIABLES USED  00038 BYTES USED
```

SUM

SUM <champ> [,<champ>] [TO <memvar list>]
[<étendue>] [FOR <exp>]

La commande SUM totalise les expressions numériques du fichier en cours d'utilisation suivant les clauses <étendue> et FOR. Nous pouvons totaliser cinq expressions au maximum. Si la clause TO est présente, le résultat du calcul est stocké dans une variable mémoire. Si la clause <étendue> n'est pas précisée, tous les enregistrements sont traités. Les enregistrements logiquement effacés sont ignorés.

Exemples :

```
. USE COURSE

. LIST
00001  CONSERVES            5       0.75
00002  PAIN                 2       0.97
00003  BOEUF                4       3.94
00004  ASSIETTES PAPIER     1       0.86
00005  FOURCHETTES          5       0.42
00006  SALADE               2       0.53
00007  FROMAGE              1       1.96
00008  LAIT                 2       1.30
00009  CHARBON              2       0.75

. SUM PRIX
 11.48

. SUM PRIX FOR NO=1
 2.82

. SUM PRIX,NO
 11.48  24

. SUM PRIX TO MSUM
 11.48

. ? MSUM
 11.48

. DISPLAY MEMORY
MSUM        (N)    11.48
** TOTAL **       01 VARIABLES USED  00006 BYTES USED
```

```
. ? MSUM*1.10
12.6280

. SUM NO*PRIX,NO,PRIX,PRIX/NO
 31.53   24   11.48 5.81
```

TEXT

TEXT
<lignes de texte>
ENDTEXT

Toutes les lignes de texte, suivant la commande TEXT, sont imprimées directement sur l'écran ou sur l'imprimante, en respectant l'état des options modifiables par la commande SET. Les macro-instructions (« & ») ne sont pas reconnues mais traitées comme du texte.

La commande TEXT est une solution très souple à l'insertion d'un bloc de texte, devant être imprimé ou affiché dans un fichier de commandes sans utiliser les commandes « ? » ou « à SAY ».

Exemple :

TEXT
Tous les caractères inclus entre les commandes TEXT/ENDTEXT, seront émis vers l'écran ou l'imprimante, sans transiter par l'interpréteur des commandes de dBASE II. La seule exception, est l'interprétation du « ; ». Si le caractère « ; » est le dernier d'une ligne de texte, lors de l'impression, aucun retour chariot ne sera généré et l'impression se poursuivra sur la même ligne.
ENDTEXT

TOTAL

TOTAL ON <clé> TO <base de données> [CHAMP <liste>] [FOR <expression>

La fonction de la commande TOTAL est semblable à celle du sous-total de la commande REPORT, excepté que les sous-totaux sont placés dans une base de données au lieu d'être imprimés. Ceci nous permet d'éliminer les détails et de condenser nos données.

Note : La base de données utilisée doit être soit triée soit indexée sur une clé.

Si la clause TO utilise un fichier existant, la structure de ce fichier est laissée intacte et utilisée pour définir le champ à totaliser arithmétiquement.

Si le fichier de destination n'existe pas, la structure du fichier de données en cours d'utilisation est copiée dans ce nouveau fichier. Cette commande est plus sélective lorsque la base de données réceptrice existe et que la phrase FIELDS est comprise dans la commande. Dans ce cas, seuls les champs numériques de FIELDS sont totalisés, sinon tous les champs numériques sont totalisés.

TOTAL peut également être utilisé pour localiser et enlever les enregistrements en double d'une base de données.

Exemple :

```
. USE ORDRES INDEX ORDRES

. DISPLAY STRU
STRUCTURE DU FICHIER            : ORDRES.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM   DECIMALE(S)
001     CLIENT       C    020
002     PART:NO      C    005
003     MONTANT      N    005
** TOTAL **            00031

. LIST
00003  HARRIS, ARNOLD     11528    44
00007  JUAN, DON          21828     5
00001  SWARTZ, JOE        31415    13
00005  MACK, JAY          31415     3
00008  SALT, CLARA        70296     9
00002  SWARTZ, JOE        76767    13
00006  TERRY, HANS        76767     5
00004  ADAMS, JEAN        89793    12

. TOTAL ON PART:NO TO APPELS
00006  ENREGISTREMENT(S) TRANSFERE(S)
```

```
. USE APPELS

. DISP STRU
STRUCTURE DU FICHIER            : APPELS.DBF
NOMBRE D'ENREGISTREMENTS        : 00006
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM   DECIMALE(S)
001     PART:NO      C    005
002     MONTANT      N    005
** TOTAL **            00011

. LIST
00001  11528    44
00002  21828     5
00003  31415    16   <-- deux enregistrements son totalisés
00004  70296     9
00005  76767    18   <-- deux enregistrements son totalisés
00006  89793    12
```

UPDATE

UPDATE FROM <base de données> ON <clé> [ADD <champ liste>]
[REPLACE <champ liste>]

La commande UPDATE utilise les données d'un second fichier pour mettre à jour le fichier en cours d'utilisation. Les articles mis à jour peuvent être soit totalisés soit remplacés complètement. Un champ de clé du fichier d'origine est comparé à un champ du fichier de destination (fichier en cours d'utilisation). La mise à jour est effectuée lorsque les clés sont identiques. Les champs de clés sont spécifiés par la phrase ON.

Note : Le fichier en cours d'utilisation doit être soit trié soit indexé sur la <clé>. Le fichier d'origine doit être trié sur la <clé>.

La lecture des deux fichiers est effectuée à partir du premier enregistrement. Lorsque les clés des fichiers d'origine et de destination sont identiques l'ajout et/ou le remplacement sont effectués. Si le nombre de caractères significatifs de la clé du fichier en cours d'utilisation est différent de celui du fichier d'origine aucune modification n'est effectuée.

Exemple :

```
. USE STOCKMAJ
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : STOCKMAJ.DBF
NOMBRE D'ENREGISTREMENTS        : 00003
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM    DECIMALE(S)
001     PART:NO      C    005
002     DISPO        N    005
003     PRIX         N    010      002
** TOTAL **             00021

. LIST
00001  21828     77       35.88
00002  30296      0      250.00
00003  89793      2   134999.00

. USE STOCK INDEX REF
. DISPLAY STRUCTURE
STRUCTURE DU FICHIER            : STOCK.DBF
NOMBRE D'ENREGISTREMENTS        : 00008
DATE DE LA DERNIERE MISE A JOUR : 00/00/00
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM         TYPE  DIM    DECIMALE(S)
001     ARTICLE      C    020
002     PRIX         N    010    002
003     PART:NO      C    005
004     DISPO        N    005
** TOTAL **             00041
```

```
.DISP ALL
00008  BANDE MAGNETIQUE          22.00 11528    16
00005  CASSETTE VIDEO            34.72 21828    77
00001  CASSETTE SONO              9.99 24776     1
00002  POCHETTE                   1.67 31415    18
00007  ETUI MAGNETOSCOPE        200.00 70296     5
00006  ACCUMULATEUR             198.37 76767    76
00004  MAGNETOSCOPE/CAMERA   134999.00 89793     5
00003  CONNECTEUR                16.33 92653     7

(Notez que le fichier est indexé sur le champ PART:NO)

 .UPDATE ON PART:NO FROM STOCKMAJ ADD DISPO REPLACE PRIX
 .LIST
00008  BANDE MAGNETIQUE          22.00 11528    16
00005  CASSETTE VIDEO            35.88 21828   154 <-+
00001  CASSETTE SONO              9.99 24776     1   I
00002  POCHETTE                   1.67 31415    18   I
00007  ETUI MAGNETOSCOPE        250.00 70296     5 <-+
00006  ACCUMULATEUR             198.37 76767    76   I
00004  MAGNETOSCOPE/CAMERA   134999.00 89793     7 <-+
00003  CONNECTEUR                16.33 92653     7   I
                                                     I
Ces trois enregistrements ont été mis à jour  >------+
```

USE

a. USE [<fichier de données>]
b. USE <fichier de données> INDEX <index principal>
[,<fichier index>,...<fichier index>]

Exemple :

```
. USE FICHIER INDEX NOM,VILLE,PART:NO,VENDEUR
```

Dans le cas « a. », la commande USE spécifie le nom du fichier de données utilisé. Si le <fichier de données> n'est pas précisé, la commande USE ferme le fichier précédemment ouvert.

Le cas « b. » permet de définir les noms des différents fichiers index associés au fichier de données. Un maximum de sept fichiers index peuvent être utilisés au même instant. Le premier des fichiers index cités est considéré comme l'index principal, c'est lui qui sera utilisé pour les éventuelles recherches (FIND) et visualisations. L'index principal ainsi que les index secondaires seront automatiquement mis à jour si leur champ de clé est modifié (APPEND, EDIT, REPLACE ou BROWSE)

Exemple :

```
. USE EXEMPLE

. USE TRACE INDEX TRACE
```

WAIT

WAIT [TO <memvar>]

Cette commande arrête temporairement l'exécution des commandes de dBASE et attend la saisie d'un caractère du clavier. Le message « → » est affiché. Dès la frappe d'un caractère, l'exécution de nouvelles commandes peut alors être effectuée. Si la clause TO est spécifiée, le caractère saisi est placé dans une variable mémoire avant la reprise du traitement.

L'option TO est très utilisée lorsqu'une sélection (menu) ne nécessite qu'un seul caractère. A la différence des commandes ACCEPT et INPUT, le retour chariot n'est pas nécessaire pour la saisie.

Si un caractère non imprimable (inférieur à 20H) est frappé lors de l'utilisation de la commande WAIT, un espace (20H) est stocké dans la variable mémoire de destination, si elle existe.

Exemple :

```
. RELEASE ALL

. WAIT TO ACTION
*** VEUILLEZ ENTRER VOTRE CHOIX : 1

. DISP MEMO
ACTION      (N)     1
** TOTAL **        01 VARIABLES USED  00006 BYTES USED
```

ANNEXE A

EXEMPLE DE FICHIER DE COMMANDES

Les exemples qui suivent vous montrent l'utilisation des fichiers de commandes dans la pratique. Ces fichiers sont utilisés comme des programmes classiques. Ils peuvent contenir des groupes de commandes exécutant des séries de fonctions telles que SORT.

Notre programme est un calcul de balance et de maintenance de livre de comptes. Il est composé de six fichiers de commandes : le fichier de contrôle, XMENU, et cinq autres fichiers subordonnés, XNOUVENT, XDEPOT, XCANNUL, XDANNUL et XBALANCE. La solution du problème peut être structurée de différentes façons. Notre exemple a pour but de vous montrer l'utilisation des commandes dBASE dans les fichiers de commandes.

Ces fichiers de commandes ont été créés par un éditeur de texte avec une extension .CMD afin de faciliter leur utilisation. Le programme est exécuté par l'utilisation de SET ALTERNATE. Voir commande SET pour cette technique.

Avant de résoudre tout problème de base de données, nous devons définir les champs à utiliser. Dans cet exemple, pour illustrer notre fichier de commandes, les champs suivants sont nécessaires :

NO	numéro du chèque
TO	bénéficiaire du chèque
AMT	montant du chèque en dollar
CAN	statut du chèque (annulé ou en cours)
DATE	date d'écriture du chèque

La structure de la base de données est ensuite créée par la commande CREATE de dBASE.

```
. CREATE
DONNEZ LE NOM DU FICHIER : CHECKREG
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP    NOM  TYPE DIM DECIMALE(S)
 001     NO,N,4
 002     TO,C,30
 003     AMT,N,10,2
 004     CAN,L
 005     DATE,C,10
 006     (cr)
DESIREZ-VOUS COMMENCER LA SAISIE (Y/N) ?N
```

FICHIER DE COMMANDES XMENU

```
NOTE - MODULE DE MENU DU PGM DE TENUE DE COMPTES CHEQUES
*
*
*  - pas de visualisation des resultats des commandes a l'ecran
SET TALK OFF
*  - l'utilisateur donne le nom du fichier des donnees
ACCEPT "Entrez le nom du fichier" to DBF
*  - acces au fichier
USE &DBF
*  - Designation du disque contenant les fichiers de commandes
ACCEPT "Quelle est l'unite de disque contenant les fichiers  'A:' ou 'B:'";
 to DISK
SET DEFAULT TO &DISK
```

```
*  - le texte suivant est visualise a l'ecran
DO WHILE T
* DO WHILE T  signifie EXECUTEZ CE PGM TANT QUE LA CONDITION EST VRAIE
* Le DO WHILE pourra etre arreter par une commande CANCEL
  ?
  ?
  ?
  ? "      MENU DE SORTIE DE BALANCE D'UN COMPTE"
  ?
  ? '         0 - FIN DE TRAITEMENT'
  ? '         1 - ENTREE DE NOUVEAUX CHEQUES'
  ? '         2 - ENTREE DE NOUVEAUX DEPOTS'
  ? '         3 - ANNULATION DE CHEQUES'
  ? '         4 - ANNULATION DE DEPOTS'
  ? '         5 - BALANCE'
  ?
  ? ' ENTREZ LE NUMERO DE VOTRE CHOIX'
 *  - Lecture de la reponse de l'utilisateur
 WAIT TO ACTION
 DO CASE

  *  - test de fin de traitement
    CASE ACTION='0'
       SET TALK ON
       *  - retour sous dBASE
       CANCEL

  *  - test pour entrer de nouveaux cheques
    CASE ACTION='1'
       *  - saisie des donnees dans le nouveau cheque
       DO XNOUVENT
```

```
  *  - test pour entrer de nouveaux depots
     CASE ACTION='2'
        *  - saisie des donnees dans les nouveaux depots
        DO XDEPOT

  *  - test d'annulation de cheques
     CASE ACTION='3'
        *  - saisie des donnees pour annulation des cheques
        DO XCANNUL

  *  - test d'annulation de depots
     CASE ACTION='4'
        *  - saisie des donnees pour annulation des depots
        DO XDANNUL

  *  - test pour la balance du compte
     CASE ACTION='5'
        *  - calcul de balance finale
        DO XBALANCE

  * - cas de saisie de donnees incorrectes dans le menu
     OTHERWISE
        ? 'Entree incorrecte, re-entrez'

  ENDCASE

ENDDO
RETURN
```

FICHIER DE COMMANDES XDEPOT

```
NOTE - CE PROGRAMME EST UTILISE POUR ENTRER DE NOUVEAUX DEPOTS
*
REMARK ENTREZ ZERO DANS LE MONTANT DU DEPOT POUR SORTIR DU PROGRAMME
*  - initialisation de la date
STORE '01/01/86' TO C:DAT
DO WHILE T
  ?
  ?
  *  - lecture des depots entres
  INPUT "MONTANT DU DEPOT              " TO C:AMT
  *  - Si montant du depot = 0, sortie du fichier de commandes
  IF C:AMT=0
     RETURN
  ENDIF
  STORE C:DAT TO ANC:DATE
  *  - lecture de la date du depot
  ACCEPT "DATE DU DEPOT SOUS FORME MM/JJ/AA  " TO C:DAT
  *  - Si date = espace, utilisation de la date en cours
  IF C:DAT=' '
    STORE ANC:DATE TO C:DAT
  ENDIF
  *  - Verification de la validite des champs
  INPUT "TOUS LES CHAMPS SONT-ILS CORRECTS ?       " TO GO:NOGO
  *  - Verification de la validite des donnees
  IF .NOT.GO:NOGO
    LOOP
  ENDIF
  *  - insertion d'un enregistrement vide dans le fichier
  APPEND BLANK
  *  - saisie des donnees du depot
  REPLACE NO WITH 0,TO WITH 'DEPOSIT',AMT WITH C:AMT
  REPLACE DATE WITH C:DAT,CAN WITH F
ENDDO
```

FICHIER DE COMMANDES XNOUVENT

```
NOTE - CE PROGRAMME EST UTILISE POUR ENTRER DE NOUVEAUX CHEQUES
*
*  - Effacement de l'ecran
ERASE
*  -  Allons-nous nous positionner à la fin du fichier
GO BOTTOM
*  - initialisation de la date
STORE '01/01/86' TO CH:DAT
DO WHILE T
  * initialisation des variables memoires
  STORE '                                   TO CH:TO
  STORE 0.00 TO M:AMT
  STORE ' ' TO CH:MEMO
  STORE NO+1 TO M:NO
  *  - Lecture des donnees
  @ 5,0  SAY 'ENTREZ LE NUM.DU CHEQUE , ENTREZ ZERO DANS LE NUM.POUR SORTIR'
  @ 6,0  SAY "SI NUM.CHEQUE = 0     FAITES RETURNS POUR D'AUTRES ENTREES"
  @ 7,0  SAY 'NUMERO DU CHEQUE          ' GET M:NO
  @ 8,0  SAY "PAYER A L'ORDRE DE        " GET CH:TO
  @ 9,0  SAY 'MONTANT DU CHEQUE         ' GET M:AMT
  @ 10,0 SAY 'DATE DU CHEQUE            ' GET CH:DAT PICTURE '9999'
  @ 11,0 SAY 'CAUSE DU CHEQUE'
  @ 13,0 SAY 'A - PUBLICITES          B - FRAIS BANQUE      C - LOCATION CARS'
  @ 14,0 SAY 'D - FRAIS + PUBLICITES  F - TRANSPORT         I - ASSURANCE'
  @ 15,0 SAY 'L - DEPENSES LEGALES    S - FOURNITURES       P - COURRIER'
  @ 16,0 SAY 'R - LOCATION            T - TELEPHONE         X - TAXES'
  @ 18,0 SAY 'ENTREZ LA LETTRE CORRESPONDANTE' GET CH:MEMO
  READ
  IF M:NO=0
    RETURN
  ENDIF
  *  - insertion d' un enregistrement vide dans le fichier
  APPEND BLANK
  *  - saisie des donnees du cheque
  REPLACE NO WITH M:NO,TO WITH CH:TO,AMT WITH M:AMT;
    DATE WITH CH:DAT,CAN WITH F,MEMO WITH !(CH:MEMO)
ENDDO
```

FICHIER DE COMMANDES XCANNUL

```
NOTE - CE PROGRAMME EST UTILISE POUR ANNULER LES CHEQUES
*
*  - message visualise a l'ecran
REMARK ENTREZ UN NUM. DE CHEQUE = 0 POUR SORTIR DU PROGRAMME
*  - execution du pgm tant que num. du cheque n'est pas egal a zero
DO WHILE T
  ?
  INPUT "ENTREZ LE NUM. DU CHEQUE A ANNULER " TO C:CAN
  *  - test pour  num. du cheque = zero
  IF C:CAN=0
     RETURN
  ENDIF
  *  - allons nous positionner a la fin du fichier
  GO TOP
  *  - recherche du num. du cheques pour annulation
  LOCATE FOR C:CAN=NO
  *  - test de fin de fichier
  IF .NOT.EOF
     *  - si la fin du fichier n'est pas atteinte
     ?
     *  - visualisation des donnees du cheque
     DISP OFF 'Paye a ',TO,' on ',DATE
     DISP OFF 'Le montant du cheque est de ',AMT
     *  - Verification du cheque
    INPUT 'Est-ce le bon cheque? (Y/N)' to REPONSE
     IF REPONSE
        *  - changement de  la valeur logique des cheques annules
        REPLACE CAN WITH T
     ENDIF
  ELSE
     *  - message signalant a l'utilisateur que le cheque n'existe pas.
     DISP OFF 'Le cheque ',C:CAN," n'existe pas"
  ENDIF
ENDDO
RETURN
```

FICHIER DE COMMANDES XDANNUL

```
NOTE - CE PROGRAMME EST UTILISE POUR ANNULATER DES DEPOTS
*
*  - allons a la fin du fichier
GO TOP
DO WHILE T
*  - cherchons le depot a annuler
  LOCATE FOR NO=0.AND..NOT.CAN NEXT 65535
  *  - Est-ce la fin du champ de caracteres
  IF .NOT.EOF
     ?
     *  - visualisation de la date et du montant du depot
     *  - assurons-nous que c'est le bon depot pour annulation
     DISP OFF 'Depose le ',DATE,' Montant = ',AMT
     INPUT ' Voulez-vous annuler celui-ci? (Y/N)' TO REPONSE
     *  - Verifions si ce depot est a annuler
     IF REPONSE
        *  - changons l'etiquette logique des depots annules
        REPLACE CAN WITH T
     ENDIF
  ELSE  * FIN DE FICHIER ATTEINTE
     ?
     *  - Signalons a l'utilisateur qu'il ne reste plus de depots a annuler
     ? 'Il ne reste plus de depots a annuler'
     RETURN
  ENDIF * CE N'EST PAS ENCORE LA FIN DU FICHIER
  ?
  INPUT "Desirez-vous annuler d'autres depots ? (Y/N)" to REPONSE
  IF .NOT.REPONSE
     RETURN
  ENDIF
ENDDO
RETURN
```

FICHIER DE COMMANDES XBALANCE

```
NOTE - CE PROGRAMME CALCULE LA BALANCE
*
SUM AMT TO OUTSTAND FOR .NOT.CAN.AND.NO>0
SUM AMT TO T:OUT FOR .NOT.CAN.AND.NO=0
?
?
*  - visualise le total des cheques impayes
DISP OFF ' TOTAL DES CHEQUES IMPAYES     = $',OUTSTAND
?
*  - visualise le total des depots
DISP OFF ' TOTAL DES DEPOTS  = $'T:OUT
?
*  - demande le solde de la balance precedente
INPUT "DONNEZ LE SOLDE DE LA BALANCE" TO BEGIN
DISP OFF 'BALANCE ACTUELLE = $',BEGIN+T:OUT-OUTSTAND
WAIT
RETURN
```

ANNEXE B

LISTE DES COMMANDES

? <exp> [,<exp>]

@ <coordonnées> [SAY <exp> [USING '<picture>']] [GET <variable> [PICTURE '<picture>']]

ACCEPT ''<cchaîne>''] TO <memvar>

APPEND [FROM <fichier> [SDF] [DELIMITED] [FOR <exp>]]
or [BLANK]

BROWSE

CANCEL

CHANGE FIELD <liste> [<étendue>] [FOR <exp>

CLEAR [GETS]

CONTINUE

COPY TO <fichier> [<étendue>] [FIELD <liste>] [FOR <exp>]
[SDF] [DELIMITED [WITH <délimiteur>]] ou [STRUCTURE]

COUNT [<étendue>] [FOR <exp>] [TO <memvar>]

CREATE [<nom de fichier>]

DELETE [<étendue>] [FOR <exp>]

DELETE FILE <fichier>

DISPLAY [<étendue>] [FOR <exp>] [<exp liste>] [OFF]

DISPLAY STRUCTURE

DISPLAY MEMORY

DISPLAY FILES [ON <unité de disque>] [LIKE <squelette>]

DO <fichier>

DO WHILE <exp>

EDIT

EJECT

ELSE

ENDDO

ENDIF
ERASE
FIND <clé>
GO ou GOTO [RECORD], ou [TOP], ou [BOTTOM], <n>
IF <exp>
INDEX ON <chaîne de caractères expression> TO <nom de fichier index>
INPUT ["<cchaîne>" TO <memvar>
INSERT [BEFORE], ou [BLANK]
JOIN TO <fichier> FOR <expression> [FIELDS <liste de champs>]
LIST
LOCATE [<étendue>] [FOR <exp>]
LOOP
MODIFY STRUCTURE
MODIFY COMMAND <fichier de commande>
NOTE ou *
PACK
QUIT [TO <liste de commandes CP/M ou fichiers.COM>]
READ
RECALL [<étendue>] [FOR <exp>]
REINDEX
RELEASE [<memvar liste>, ou [ALL]
REMARK
RENAME <nom du fichier en cours> TO <nom du nouveau fichier>
REPLACE [<étendue>] <champ> WITH <exp> [AND <champ> WITH <exp>]
REPORT [<étendue>] [FORM <fichier de format>] [TO PRINT] [FOR <exp>]
RESET
RESTORE
RETURN
SAVE TO <fichier>
SELECT [PRIMARY ou SECONDARY]
SET <param> [ON], ou [OFF]
SET ALTERNATE TO <fichier>
SET DEFAULT TO <unité de disque>

SET DATE TO <chaîne>
SET FORMAT TO <fichier de format>
SET HEADING TO <chaîne>
SET INDEX TO <fichier index>
SET MARGIN TO <n>
SKIP <+/—> [<n>]
SORT ON <champ> TO <fichier> [ASCENDING], ou [DESCENDING]
STORE <exp> TO <memvar>
SUM <champ> [<étendue>] [TO <memvar liste>] [FOR <exp>]
TOTAL TO <fichier> ON <variable> [FIELDS <liste de champs>]
UPDATE FROM <fichier> ON <variable> [ADD <liste de champs>]
[REPLACE <liste de champs>]
USE <fichier> [INDEX <nom du fichier index>]
WAIT [TO <memvar>]

LES FONCTIONS

@(<chaîne1>,<chaîne2>)	fonction AT
*	effacement (logique) d'enregistrement
#	numéro enregistrement
!(<chaîne de caractères>)	conversion en majuscule
$(<chaîne de caractères>,<début>,<longueur>)	fonction de sous-chaînes
<chaîne1>$<chaîne2>	recherche de sous-chaînes
CHR(<expression numérique>)	donne la valeur ASCII d'un nombre
DATE()	fonction de date
EOF	fonction de fin de fichier
FILE(<fichier>)	fonction test d'existence de fichier
INT(<expression numérique>)	fonction nombre entier
LEN(<chaîne de caractères>)	fonction de longueur
STR(<expression numérique>,<largeur>[,<décimales>])	fonction de chaînes
VAL(<chaîne de caractères>)	fonction de valeur
TRIM(<chaîne de caractères>)	rapprochement de chaînes
TYPE(<exp>)	type de données

ANNEXE C

CONTRAINTES ET LIMITES

Nombre de champs par enregistrement	32 au maximum
Nombre de caractères par enregistrement	1 000 au maximum
Nombre d'enregistrements par base de données	65 535 au maximum
Nombre de caractères par chaîne de caractères	254 au maximum
Précision des champs numériques	10 nombres
Le plus grand nombre	1.8×10^{63} approx.
Nombre de variables mémoires	64 au maximum
Nombre de caractères par ligne de commande	254 au maximum
Nombre d'expressions dans une commande SUM	5 au maximum
Nombre de caractères dans un titre de rapport	254 au maximum
Nombre de caractères dans une clé index	99 au maximum
Nombre de GETs en suspens	64 au maximum
Nombre de fichiers ouverts à un moment donné	16 au maximum

ANNEXE D

LES MESSAGES D'ERREUR

*** DEFINITION INCORRECTE DU NOMBRE DE DECIMALE(S) ***

*** NOM DE FICHIER INCORRECT ***
La syntaxe du nom de fichier est erronée.

*** DEFINITION INCORRECTE DU NOM DU CHAMP ***

*** DEFINITION INCORRECTE DU TYPE DE CHAMP ***
Le champ doit être de type C, N, ou L.

*** DEFINITION INCORRECTE DE LA DIMENSION DU CHAMP ***

*** FICHIER VIDE : INSERTION IMPOSSIBLE ***
utilisez plutôt la commande APPEND

*** LE FICHIER NE PEUT ETRE OUVERT ***
erreur interne , veuillez contacter votre fournisseur

*** CE FICHIER DE COMMANDE N'EST PAS SUR L'UNITE ACTIVE ***
vérifiez l'orthographe

*** AUCUN ENREGISTREMENT NE CORRESPOND A VOTRE SELECTION ***

*** LA BASE DE DONNEES N'UTILISE PAS DE FICHIER INDEX ***
FIND ne peut être utilisé que sur des fichiers indexés

*** DEPASSEMENT DU NOMBRE MAXIMUM DE FICHIERS SUR DISQUE ***
dépassement de la capacité du disque CP/M.

*** DEPASSEMENT DE LA CAPACITE DU DISQUE ***

*** FIN DE FICHIER INATTENDUE ***
le format de la base de données en USE n'est pas correct.
Si tous les enregistrements sont présents et corrects, exécutez un PACK et re-INDEXez la base de données

*** PHRASE INCOMPLETE : "FIELD" MANQUANT ***

*** CE FICHIER EXISTE DEJA ***

*** CE FICHIER N'EXISTE PAS ***

*** FICHIER EN COURS D'UTILISATION ***
exécutez une commande USE ou CLEAR pour fermer le fichier

*** LE FICHIER DE FORMAT NE PEUT ETRE OUVERT ***

*** LE FICHIER DE FORMAT N'A PAS ETE CREE ***

*** TYPE DE DONNEES INCORRECT ***

*** NUMERO D'ENREGISTREMENT INCORRECT ***

*** NOM DE VARIABLE INCORRECT ***
seuls les ' :' et l'alphanumérique sont permis dans les variables et noms de champs

*** L'INDEX NE CORRESPOND PAS A LA BASE DE DONNEES ***
dBASE ne peut connecter la clé avec la base de données. Essayez un autre fichier index

*** FICHIER INDEX NON TROUVE ***
vérifiez l'orthographe ou INDEXez la base de données

*** COMMANDE "JOIN" NE PEUT GENERER PLUS DE 65,534 ENREG.***
vérifiez la clause FOR. Dépassement de la capacité de dBASE (65 534 enregistrements maximum)

*** LES CLES N'ONT PAS LA MEME DIMENSION ***

*** LA MACRO-INSTRUCTION N'EST PAS UNE CHAINE DE CARACTERES ***
&macros doit être une chaîne de caractères

*** ERREUR : TOTALISATION DE PLUS DE 5 CHAMPS ***

*** DEPASSEMENT DU NOMBRE MAXIMUM D'IMBRICATIONS ***

*** INEXISTENCE DE L'EXPRESSION A TOTALISER :***

*** PHRASE INCOMPLETE : "FOR" MANQUANT ***

*** PHRASE INCOMPLETE : "FROM" MANQUANT ***

*** ENREGISTREMENT NON TROUVE ***
dBASE ne peut trouver la clé

*** EXPRESSION NON-NUMERIQUE ***

*** FICHIER NON TROUVE ***

*** PHRASE INCOMPLETE : "ON" MANQUANT ***

*** DEPASSEMENT DE LA PLACE ALLOUEE AUX VARIABLES MEMOIRE ***
Réduisez le nombre ou la taille des variables mémoire

*** LA DIMENSION DE L'ENREGISTREMENT DEPASSE 1 000 CARACTERES ***

*** L'ENREGISTREMENT N'EST PAS INDEXE ***
le fichier index n'a pas été mis à jour après ajout d'un enregistrement. Re-indexez

*** ENREGISTREMENT NON SITUE ***
le numéro de l'enregistrement est supérieur au nombre d'enregistrements de la base de données. L'enregistrement n'existe pas.

*** ERREUR INTERNE CONTACTEZ VOTRE FOURNISSEUR ***

*** ORIGINE ET DESTINATION DES TYPES DE DONNEES DIFFERENTES **

*** ERREUR DE SYNTAXE ***

*** ERREUR DE SYNTAXE DANS LA SPECIFICATION DU FORMAT ***

*** ERREUR DE SYNTAXE VEUILLEZ CORRIGER ***

*** PHRASE INCOMPLETE : "TO" MANQUANT ***

*** TROP DE CARACTERES ***

*** TROP DE FICHIERS SONT OUVERTS ***
nous pouvons ouvrir simultanément 16 fichiers

*** DEPASSEMENT DU NOMBRE MAXIMUM DE VARIABLES MEMOIRE ***
nous pouvons disposer d'un maximum de 64 variables mémoire

*** TROP DE COMMANDES "RETURN" RENCONTREES ***
erreur probable dans la structure des fichiers de commandes

*** PHRASE INCOMPLETE : « WITH » MANQUANT ***

*** ERREUR INTERNE CONTACTEZ VOTRE FOURNISSEUR ***

*** CETTE COMMANDE NE PEUT ETRE UTILISEE

*** VARIABLE EST INCORRECTE ***
Créez la variable, ou vérifiez l'orthographe

ZIP

Depuis son introduction dans les micro-ordinateurs, dBASE II est la base de données relationnelle la plus puissante et la plus facile à utiliser. Incorporé à ce logiciel, nous avons un programme appelé ZIP.

CARACTERISTIQUES DE ZIP

- Préparation d'un format de sortie à l'imprimante aussi bien qu'un format de saisie à l'écran. ZIP peut gérer jusqu'à 88 lignes.
- Création automatique des codes exécutants : ZIP écrit la commande READ à la fin de chaque fichier qui utilise des commandes GET ou après 64 GET dans de longs fichiers. ZIP valide les noms de variables, écrit les fichiers .FMT ou écrit un fichier entier .CMD avec des commandes ERASE, SET FORMAT TO PRINT/SCREEN, SET MARGIN TO xxx et RETURN.
- Insertion des commandes de dBASE II dans vos fichiers de format : sauvegarde du format et des fonctions par les capacités d'édition de ZIP.
- A la fin de votre écran, une ligne vous signale la position de votre curseur (rang et colonne), vous permettant ainsi de dessiner rapidement et facilement votre format de saisie.
- Pendant la création d'un format de sortie ou d'un écran formaté, vous pouvez changer les marqueurs verticaux et horizontaux, la tabulation, la dimension de la page et la marge de l'imprimante.
- Les marqueurs horizontaux et verticaux dessinent et effacent les lignes, facilitant ainsi la création des cases, le dessin des lignes, l'effacement de la totalité des rangs ou des colonnes.
- L'opération est d'autant plus rapide qu'il n'y a pas de touche de contrôle à appuyer. Votre clavier fonctionne comme d'habitude. Les touches <Return>, <Delete>, <Backspace> et <Tab> font exactement ce que vous attendez d'elles.
- Vous avez accès aux valeurs par défaut aussi pouvez-vous installer ZIP selon votre désir.

```
***** File B:ESSAI ***
                          +---------+
                          I  ESSAI  I
                          +---------+

**********************
 ESSAI DE FACTURATION
**********************

 FACTURE @factnbr                    DATE #date

  CLIENT @client

-------------------------------------------------------------------------------
      [USE B:GETPRIX]![STORE 0 TO Sum]
      [DO WHILE .NOT. EOF]@[? Job:Nmbr,Descrip, Taxe]@[Sum = Sum + Taxe]
      [ENDDO]

-------------------------------------------------------------------------------
                                          TOTAL FACTURE @Sum        37,79
```

```
* ESSAI.CMD
SET FORMAT TO PRINT
SET MARGIN TO  40
@  0,23 SAY [+--------------+]
@  1,23 SAY [I   ESSAI      I]
@  2,23 SAY [+--------------+]
@  7, 5 SAY [**********************]
@  8, 7 SAY [ESSAI DE FACTURATION ]
@  9, 5 SAY [**********************]
@ 12, 5 SAY [FACTURE]
@ 12,13 SAY [factnbr]
@ 12,40 SAY [DATE]
@ 12,45 SAY date
@ 15, 6 SAY [CLIENT]
@ 15,13 SAY client
@ 17, 0 SAY [-----------------------------------------;
-------------------]
```

```
USE B:GETPRIX
STORE 0 TO Sum
DO WHILE .NOT. EOF
? Job:nmbr,Descrip,Taxe
Sum = Sum + Taxe
ENDDO
@ 35, 0 SAY [----------------------------------------;
-------------------]
@ 37,42 SAY [TOTAL FACTURE]
@ 37,58 SAY Sum
@ 37,75 SAY [37,79]
SET FORMAT TO SCREEN
RETURN
```

RESUME DES FONCTIONS DE ZIP

Sur la page précédente, vous avez pu voir comment ZIP écrit un fichier de commandes (extension .CMD) à partir d'un écran d'entrée.

La case a été créée par l'utilisation des commandes de dessin et d'effacement des marqueurs horizontaux et verticaux insérant automatiquement un signe « + » à l'intersection de deux lignes. Les symboles des marqueurs peuvent être changés pendant la création des formats.

Le texte, les noms de variables (précédés de « @ » ou #) et l'insertion des commandes (entourées de crochets) sont simplement tapés sans l'utilisation des fonctions de contrôle. Lorsque le format satisfait l'opérateur, ce dernier tape « /S ».

ZIP imprime ce format si l'opérateur répond affirmativement à la question : Voulez-vous imprimer ce format ? ZIP initialise aussi les marges de l'imprimante aux valeurs spécifiées par l'opérateur.

ZIP balaie le format et écrit automatiquement toutes les commandes : à <Rang,Colonne> SAY <texte>. Lorsqu'il rencontre un symbole de DISPLAY (@), ZIP écrit : @ <Rang,Colonne> SAY <variable>.

ZIP écrit aussi les commandes GET même si c'est un fichier d'impression. Lorsqu'il rencontre un symbole de GET (#), ZIP demande si c'est délibéré et l'opérateur répond affirmativement. Si la réponse est négative, ZIP met fin à l'exécution du fichier.CMD puis positionne le curseur sur le symbole erroné.

Dans l'exemple précédent, l'opérateur a également inséré des commandes de dBASE II : ces commandes sont entourées de crochets, ainsi le fichier Essai.CMD est une combinaison de format et de fonctions.

ZIP termine ce fichier en écrivant SET FORMAT TO SCREEN et RETURN. Ce fichier peut maintenant être utilisé dans n'importe quel autre fichier de commandes de dBASE II en ajoutant simplement **DO Essai** dans ce dernier.

L'écran pourrait être sauvegardé comme un fichier de format (extension.FMT) lors de la spécification du type de fichier désirée.

Pour un fichier de format, ZIP écrit uniquement les phrases **à...SAY** et tous les articles entourés de crochets. Les phrases **SET** ne sont pas incluses et le premier commentaire devrait être : * **ESSAI.FMT**.

Souvenez-vous que seuls les commentaires peuvent être insérés dans les fichiers de format. ZIP ne vérifie pas la validité des commentaires.

Nous allons vous montrer, dans les pages suivantes, la facilité d'utili sation de ZIP.

COMMENT TRAVAILLER AVEC ZIP

Tapez :

```
ZIP
```

ZIP commence par vous montrer un écran d'aide contenant toutes les commandes et les valeurs par défaut du système.

Pour l'instant, peu importe si vous avez mémorisé ou non toutes les commandes; vous pouvez les vérifier à nouveau lorsque vous travaillez avec ZIP en pressant la touche de commande deux fois (ou n'importe quel autre symbole installé). Pressez n'importe quelle touche pour continuer.

L'écran s'efface et la dernière ligne de votre écran devrait être :

```
<NEW> or <OLD> file (Q to Quit)?
```

ZIP ne commence à fonctionner qu'au moment où vous lui direz si vous désirez créer un nouveau fichier (en tapant la lettre N) ou désirez mettre à jour un ancien fichier (en tapant la lettre O). Vous pouvez changer d'avis et revenir au système en tapant la lettre Q. Puisque nous n'avons encore rien écrit, tapons : N

ZIP nous demande maintenant le nom du nouveau fichier :

FILE NAME (unité de disque optionnelle):

Si vous ne spécifiez pas l'unité de disque (un caractère suivi de deux-points), les fichiers seront sauvegardés sur l'unité de disque active.

Les noms de fichiers peuvent avoir un maximum de 8 caractères (en plus des 2 caractères spécifiant l'unité de disque) et contenir les caractères suivants :

A-Z (a-z), 0123456789, $« @ » et les deux-points (:)

ZIP n'accepte aucun autre caractère dans un nom de fichier, ce qui signifie que vous ne pouvez entrer vos propres extensions de fichier. ZIP sauvegardera votre fichier de travail comme <nom>.ZIP et écrira les fichiers de commandes comme <nom>.CMD.

Les caractères alphabétiques peuvent être entrés en majuscules ou en minuscules; ZIP les convertira en majuscules.

CP/M requiert que les « : » figurent en tant que deuxième caractère dans le nom. Si vous avez saisi un « : » dans n'importe quelle autre position, ZIP intercepte le nom et vous demande de le corriger.

Lorsque vous avez entré le nom du fichier et appuyé sur la touche <enter>, le curseur est initialisé à 0,0 et ZIP vous précise sa position.

Maintenant essayons la touche Tabulation ou le retour chariot. Si votre touche se répète automatiquement, maintenez-la enfoncée et le curseur se déplace ligne par ligne jusqu'à ce qu'il atteigne la fin de l'écran.

Il existe cependant un moyen plus rapide pour accéder au début de l'écran. Tapez les trois commandes suivantes (en majuscules ou en minuscules) :

```
/T
/B
/M
```

Ces trois clés vous permettent de vous déplacer rapidement à l'intérieur de l'écran.

Les commandes Début et Fin restent sur la même colonne afin de vous faciliter l'alignement de vos entrées. La commande Milieu déplace le curseur au milieu d'un rang vous permettant de centrer vos titres.

Peu importe le moment où vous démarrez une nouvelle session ZIP, la tabulation est automatiquement mise à la valeur que vous avez installée (5) et la dimension de la page est mise à la dimension de votre écran. La tabulation et la dimension de la page peuvent être changées à n'importe quel moment pendant une session ZIP. Tapez les trois commandes suivantes (après la première commande, veuillez attendre l'écran d'aide) :

```
//
T          (La commande marqueur n'est pas utilisée ici)
9
```

Pressez n'importe quelle touche pour revenir à votre écran de travail puis pressez la touche Tabulation. ZIP vous dira que la tabulation est maintenant mise par multiple de 9 à travers l'écran. Pendant une session de ZIP, vous pouvez changer cette tabulation aussi souvent que vous le désirez en retour nant à l'écran d'aide.

La dimension de la page peut être modifiée de la même façon que la tabulation. A partir de l'écran d'aide, vous pouvez choisir n'importe quelle dimension de page en commençant par la dimension d'un écran (minimum de ZIP) jusqu'à 88 lignes. Ce qui vous permet de préparer un rapport à partir de votre console ou d'imprimer 8 lignes par pouce sur du papier normal.

NOTE : Pendant une session ZIP, si vous raccourcissez la dimension d'une page, toute information au-delà de la fin de la nouvelle page est effacée.

Maintenant, positionnez le curseur au sommet gauche de votre écran (tapez <**enter**>, /T) et tapez :

```
 /H
 /V
<Tabulation deux fois>
 /V
<Tabulation deux fois>
 /V
 /H
```

Les mêmes commandes peuvent dessiner et effacer les lignes horizontales et verticales. A l'intersection des deux lignes, un symbole « + » est automatiquement ajouté, facilitant ainsi le dessin de votre écran et de votre format selon votre désir.

Si votre curseur se trouve sur le symbole d'un marqueur horizontal, lorsque vous entrez la commande horizontale, ZIP efface le reste de la ligne à droite.

Si le curseur se trouve sur un symbole « + », ZIP peut vous répondre de différentes façons.

Si le curseur se trouve sur un symbole « + » et a d'autres caractères à sa gauche, le signe « + » est laissé tel qu'il était mais le reste de la ligne horizontale à droite est effacé. S'il ne se trouve aucun caractère immédiatement à gauche du curseur et le symbole suivant à droite n'est pas un « + », la ligne horizontale est effacée et le symbole « + » est changé par la valeur courante du marqueur vertical.

Si le symbole est entouré par d'autres symboles « + », ils sont tous laissés intacts. Seule la ligne horizontale à droite, au-delà du plus éloigné symbole « + » est effacée. Dans ce cas, ZIP ne retrace pas la portion de cette ligne horizontale à moins que vous ne déplaciez le curseur à droite du symbole « + ».

La commande du marqueur vertical utilise la même logique.

Ces commandes sont beaucoup plus faciles à utiliser qu'elles ne paraissent puisque les décisions sont prises par ZIP.

NOTE : En utilisant la commande Marqueur deux fois (/H,/H ou /V,/V), vous pouvez effacer rapidement la totalité des rangs ou des colonnes.

Maintenant, tapez les commandes qui suivent (veuillez attendre l'écran d'aide) :

```
//
P
40
.          (valeur courante de votre marqueur vertical)
*
```

Ces commandes changent la dimension de la page à 40 lignes et le marqueur vertical en (*). Pressez n'importe quelle touche pour revenir à votre écran de travail sous ZIP.

Frappez plusieurs fois le retour chariot pour laisser quelques lignes. Tapez /H puis /V. Au lieu d'effacer la ligne verticale, la dernière commande va changer tous les symboles en (*) du fait qu'elle n'était pas au début du marqueur vertical courant.

Maintenant tapez :

```
/N         (veuillez attendre un nouvel écran)
B
```

Ces commandes vous placent à la fin de l'écran suivant. ZIP vous signale : « Rang 39, Colonne 0 » (la dimension de notre page est de 40 lignes numérotées de 0 à 39). Notez que la dernière ligne verticale tracée va jusqu'à la fin de la page même si elle n'est pas visible à l'écran.

Avec des écrans plus longs, vous pourriez désirer utiliser :

/F pour vous positionner à l'intérieur du premier écran de votre format avec Rang 0 au début de l'écran; ou
/L pour vous positionner à l'intérieur du dernier écran de votre format avec le dernier rang à la fin de votre écran.

Afin de sauvegarder ce que vous avez entré, tapez : /S. ZIP vous pose la question suivante :

```
SAVE <nom> as CMD or as FMT file (C, F or stop)?
```

Si vous ne répondez pas par « C » ou « F », ZIP vous ramène le fichier de travail.

Si vous tapez « C » ZIP vous demande :

Is this form to be printed (Y or N)?

Toute lettre autre que « Y », (minuscule ou majuscule) est équivalente à « Non ».

ZIP vous donne ensuite une chance de changer le nom du fichier (qu'il soit de type .CMD et .FMT) en posant la question :

Fichier <nom> : do you want to change its name (Y or N)?

Si vous répondez par « Y », ZIP vous demande alors le nom du nouveau fichier. Vous pouvez aussi spécifier une unité de disque différente dans le nouveau nom afin de sauvegarder votre fichier sur une autre unité de disque. Si vous pressez n'importe quelle autre touche, ZIP utilise le nom que vous avez donné au départ.

ZIP commence par sauvegarder une image de l'écran ou du format d'impression que vous avez créé. Les différentes étapes vous seront spécifiées par :

Writing screen image <nom>.ZIP.

Lorsqu'il a sauvegardé l'image de l'écran, ZIP écrit aussi une copie imprima ble.Il nous le signale par :

Writing printable file <name>.ZPR.

Lorsqu'il écrit un fichier de commande de dBASE II, ZIP insère d'abord le nom du fichier en commentaire (* <NOM>.CMD ou * <NOM>.FMT. Si c'est un fichier de type .CMD, ZIP écrira :

ERASE (si c'est un format d'écran)

ou

SET FORMAT TO PRINT
SET MARGIN TO xx (si c'est un format destiné à l'impression).

Pour les deux fichiers de type .CMD et .FMT, ZIP écrit toutes les phrases @...SAY et GET. ZIP insère aussi un **READ** après 64 GET. Pour ce fait, il vous signale le rang d'occurence.

Juste avant le **RETURN** final à la fin du fichier de commandes, ZIP termine par un **READ**. Si c'est pour des fichiers d'impression, il dira **SET FORMAT TO SCREEN**.

Après avoir sauvegardé votre fichier de travail et le programme, il vous est toujours possible de récupérer l'écran original, de le modifier et de le sauvegarder sous un autre nom de fichier.

Lorsque ZIP trouve un GET dans un format d'impression, il vous pose la question suivante :

"GET" in PRINTOUT: Okay (Y ou N)?

Si le GET a été utilisé délibérément, tapez « Y » et ZIP continue d'écrire le fichier <nom>.CMD.

Si le GET avait été utilisé par erreur, appuyez sur n'importe quelle touche. ZIP termine le fichier .CMD en nous notifiant **INCOMPLETE COMMAND FILE**, puis positionne votre curseur sur le symbole GET erroné afin que vous puissiez soit le changer soit l'enlever.

NOTE : Puisque l'exécution de votre programme a été abandonnée, vous devez sauvegarder à nouveau votre fichier (/S).

ZIP valide vos noms de variables pour s'assurer que ces fichiers fonction nent dans vos programmes dBASE II.

Si, par inadvertance, vous commencez vos noms de variables avec un caractère interdit par dBASE II (tout caractère autre que alphabet, nombre et deux-points), ZIP vous dit :

NO VARIABLE: continue (Y or N)?

En pressant toute lettre autre que « Y » (majuscule ou minuscule), la sauve garde se termine et votre curseur se positionne sur l'erreur dans votre fichier de travail.

Lorsque vous commencez ou terminez un nom de variable avec deux-points, il vous est demandé :

```
HANGING COLON:  continue (Y or N)?
```

L'appui de n'importe quel caractère autre que « Y », termine votre sauvegarde et positionne le curseur sur l'erreur afin que vous puissiez la corriger rapidement et facilement. La sauvegarde devra se faire à nouveau.

Dans la plupart des cas, vous n'avez pas besoin de regarder les fichiers générés par ZIP à moins que vous ne vouliez ajouter des codes pour la détection des erreurs, etc. Ajoutez seulement dans votre programme principal :

```
DO <nom>
```

ou

```
SET FORMAT TO <nom>
```

puis exécutez-le. Si votre programme fournit les valeurs aux variables que vous avez entrées par ZIP, vous aurez exactement l'écran ou le format que vous désirez.

Pour commencer un nouveau fichier, effacez d'abord l'ancien en tapant :

```
/E
```

puis pressez la touche « Y » lorsque ZIP vous demande :

```
Erase everything (Yes or No)?
```

Il vous est ensuite demandé si vous désirez un nouveau ou un ancien fichier. Si vous voulez vous arrêter, tapez Q lorsqu'il vous sera demandé :

```
<NEW> or <OLD> file (Q to QUIT)?
```

Tapez « Y » lorsque ZIP vous demande :

```
QUIT to system (Yes or No)?
```

Il vous est aussi possible d'abandonner le travail en cours en tapant :

```
/Q
```

A nouveau, ZIP vous demande de confirmer que vous voulez vous arrêter avant d'effacer votre travail et de fermer vos fichiers.

Lors de votre arrêt, si votre fichier était plus long que votre écran, ZIP vous donne une dernière et rapide revue de la totalité du fichier puis vous ramène au système.

NE JAMAIS DIRE .@. SAY..GET A NOUVEAU

Utilisons maintenant ZIP pour écrire quelques commandes « @ »SAY et GET. C'est beaucoup plus facile que vous ne le pensiez.

Vous désirez peut-être créer des formats ou des écrans de saisie afin d'avoir un fichier que vous pourrez exécuter après avoir terminé cette section.

Lorsque vous commencez votre travail sur un nouveau fichier, ZIP initialise automatiquement la dimension de la page à un écran simple. Si vous faites appel à un fichier déjà existant, ZIP ajuste la dimension de la page à la dimension de ce fichier.

Si votre fichier est destiné à l'impression, la dimension de votre page peut atteindre jusqu'à 88 lignes. La signalisation des rangs et colonnes sur votre écran vous permet de positionner vos titres, dates, commentaires et variables exactement là où vous les désirez.

Pour positionner une variable, tapez le symbole @ (pour SAY une variable) ou le symbole # (pour GET une variable), à la position où vous désirez l'occurence de votre champ.

Maintenant, entrez-y le nom de la variable. Puisque ZIP connaît les règles pour les noms de variables dBASE II, il ne vous est pas besoin d'entrer un caractère de fin de nom. ZIP met fin au nom après avoir lu 10 caractères acceptables ou lorsqu'il rencontre un

caractère non permis par dBASE II (espace compris). Si le nom de votre variable fait 10 caractères de long, vous pouvez utiliser la position suivante sans séparateur entre le nom de la variable et le reste de la ligne.

Vous trouverez ci-dessous les caractères que dBASE II accepte dans un nom de variable :

A-Z, a-z, 0123456789 et insertion des deux-points (:)

ZIP vous laisse taper un « : » à n'importe quelle position dans un nom de variable, mais il intercepte une entrée incorrecte à l'écriture du fichier <**nom**>**.CMD** et vous donne une chance de la corriger (voir Save au chapitre précédent).

Si un nom de variable est entré avec un caractère non permis par dBASE II, ZIP tronque le nom et assume que vous désirez commencer votre message ou titre avec le caractère inacceptable.

Maintenant, regardons à nouveau l'écran d'aide en tapant la commande marqueur deux fois (//). Vous trouverez probablement les commandes **/D** (ou votre touche Delete/Del), **/I**, **/K** et **/A** utiles avant d'avoir terminé avec votre format.

COMMENT INSERER DES COMMANDES dBASE DANS VOTRE FICHIER DE FORMAT

Afin de réduire le besoin d'un éditeur externe, ZIP vous permet d'insérer autant de commandes dBASE II qu'il vous est possible d'ajouter dans votre format. Il existe cependant des règles à suivre :

1) Les crochets doivent entourer la commande.
2) Les crochets doivent figurer sur une seule ligne (utilisez le « ; » pour des commandes plus longues).
3) Les commandes entre crochets doivent être séparées du texte et des autres commandes entre crochets.

Pour les séparer, utilisez un caractère de commande simple juste en face du crochet gauche ou deux caractères de commande (@@,##,@# ou #@). Il n'est pas besoin de séparer les commandes entre crochets des noms de variables les précédant.

Ci-dessous, vous trouverez un écran d'entrée et une écriture de fichier de commandes, générés par ZIP.

```
[RESTORE FROM B:CONSTANT]

     **********************
      ESSAI DE FACTURATION    @@[ANOTHER dBASE II COMMAND]
     **********************

     FACTURE @:factnbr  [SAVE TO A] **  DATE @date[DO DateTest]

     CLIENT @client  [Do Verif]
-------------------------------------------------------------------------
[USE B:GETPRIX]*[STORE 0 TO Sum]
[DO WHILE .NOT. EOF]@[? Job:Nmbr,Descrip, Taxe]*[Sum = Sum + Taxe]
[ENDDO]

                                                                    HZIN
Row 22, Col 75
-------------------------------------------------------------------------
CHAR MODE




-------------------------------------------------------------------------

                                          TOTAL FACTURE  @Sum        HIZN
Row 37, Col 75
-------------------------------------------------------------------------
CHAR MODE

* B.CMD
SET FORMAT TO PRINT
SET MARGIN TO 40
RESTORE FROM B:CONSTANT
@  7, 5 SAY [**********************]
@  8, 7 SAY [ESSAI DE FACTURATION]
ANOTHER dBASE II COMMAND
@  9, 5 SAY [**********************]
@ 12, 5 SAY [FACTURE]
@ 12,13 SAY factnbr
SAVE TO A
```

```
@ 12,40 SAY [DATE]
@ 12,45 SAY date
DO DateTest
@ 15, 6 SAY [CLIENT]
@ 15,13 SAY client
Do Verif
@ 17, 0 SAY [-----------------------------------------------;
-----------------------------]
USE B:GETPRIX
STORE 0 TO Sum
DO WHILE .NOT. EOF
? Job:Nmbr,Descrip, Taxe
Sum = Sum + Taxe
ENDDO
@ 35, 0 SAY [-----------------------------------------------;
-----------------------------]
@ 37,41 SAY [TOTAL FACTURE]
@ 37,57 SAY Sum
SET FORMAT TO SCREEN
RETURN
```

QUESTION : Comment pouvez-vous visualiser les résultats des expressions ou montrer plusieurs champs en même temps quand leurs noms occupent les positions où vous les désirez ?

REPONSE : Si le fait de pouvoir insérer des commandes de dBASE II dans votre fichier ne peut vous aider, vous trouverez là un des cas pour lequel vous voudriez visualiser le fichier de commande <nom>.CMD écrit par ZIP.

ZIP va vous faciliter la tâche car il peut gérer jusqu'à 40 symboles GET et SAY dans une ligne de 80 caractères.

Il vous suffit d'entrer uniquement les symboles aux positions où vous voudrez visualiser ou entrer vos champs. Puis, lorsque ZIP vous demande : **NO VARIABLE — continue (Y or N) ?**, tapez « Y ».

Cette procédure a l'avantage de vous aider à trouver les positions pour entrer des expressions ou des noms de variables car ZIP aurait écrit dans le fichier <nom>.CMD :

```
   @  xx,yy SAY
ou @  xx,yy GET          (le reste de la ligne est rempli de blancs)
```

Il vous faudrait répondre à beaucoup de questions et de messages provenant de ZIP, d'où l'inconvénient de cette procédure. Afin de pallier cette perte de temps, vous pourrez entrer uniquement des nombres simples ou des lettres dans ZIP puis les remplacer ensuite par les noms et expressions en éditant le fichier <nom>.CMD.

Lorsque vous avez terminé avec votre écran ou format, sauvegardez votre fichier en tapant /S, puis /Q pour vous arrêter.

ATTENTION

Ne touchez pas au fichier image-écran <nom>.ZIP. Ce fichier est en code ASCII, aussi pouvez-vous utiliser la fonction « TYPE » de CP/M pour regarder ce fichier (utilisez control-S pour arrêter ou commencer le défilement du fichier). Toutefois, selon la façon dont le fichier a été sauvegardé, un éditeur de texte pourra probablement le détruire.

Si vous désirez une impression de votre écran, utilisez votre traitement de texte pour imprimer le fichier **<nom>.ZPR** ou appuyez sur ctrl-P puis utilisez la commande TYPE de CP/M.

Si vous le désirez, vous pouvez utiliser votre éditeur de texte pour regar der le fichier de commandes de dBASE II **<nom>.CMD** ou **<nom>.FMT** écrit par ZIP. Vous pouvez utiliser la commande TYPE de CP/M pour une rapide vérification avant d'exécuter votre programme.

Maintenant appelez dBASE et assurez-vous qu'il existe des valeurs pour les variables que vous avez décrites dans votre format, puis tapez :

```
DO <nom>.
ou SET FORMAT TO <nom>
```

POUR dBASE II :

@ indique la position pour SAY une variable
# indique la position pour GET une variable

Il ne vous est pas nécessaire de spécifier un caractère de fin de champ car ZIP assume automatiquement que le nom de la variable est complet lorsqu'il atteint le premier

caractère non reconnu par dBASE II (ou 10 caractères maximum). Les caractères utilisables pour les noms de variables de dBASE II sont :

A - Z, a - z, 0123456789 et insertion de deux-points (:)

ZIP vérifie les « : », les noms de variables manquants et les GET dans un fichier d'impression, puis vous laisse les corriger, ou les ignore. Après 64 GET, ZIP écrit une commande READ avec un marqueur dans le fichier .CMD et un message à l'écran pour vous signaler l'occurence de cette commande.

Vous pouvez exécuter les fichiers de commandes **<nom>.CMD** et **<nom>.FMT** écrits par ZIP sans autre correction. Pour un fichier d'écran, ZIP écrira ERASE au début du fichier. Pour un fichier d'impression, ZIP écrira au début du fichier SET FORMAT TO PRINT et SET MARGIN TO XX puis, juste avant la fin, il écrira SET FORMAT TO SCREEN et RETURN.

ZIP sauvegardera aussi une image de votre écran comme **<nom>.ZIP** et une copie imprimable **<nom>.ZPR**.

ZIP se réserve les crochets pour délimiter ses textes. Les crochets peuvent être utilisés uniquement pour insérer des commandes de dBASE II et des commentaires dans vos fichiers.

LES VALEURS DYNAMIQUES

- est considéré comme un marqueur vertical mais il peut être changé en « T » ou autre caractère lors de l'installation de ZIP. Il peut également être changé temporairement, à n'importe quel moment lorsque vous travaillez sur un format avec ZIP.
- est le marqueur horizontal et peut être changé pendant l'installation ou dynamiquement lors d'un travail avec ZIP (fonctionnement identique au marqueur vertical).

TABULATION est initialisée à 5 espaces mais peut être changée à vos propres valeurs de 1 à 9 lors de l'installation de ZIP, ou peut être dynamiquement changée à volonté pendant votre travail avec ZIP.

DIMENSION DE PAGE peut être dynamiquement initialisée à n'importe quelle valeur à partir d'un écran simple jusqu'à 88 lignes lors d'un travail avec ZIP.

MARGE DE L'IMPRIMANTE peut être posée à n'importe quelle valeur de 0 à 127.

AUTRES COMMANDES ET SYMBOLES DE ZIP

/ est la commande marqueur, mais pourrait être posée en « ! » ou quelque autre caractère que vous n'utiliserez pas comme texte dans vos écrans et formats de sortie.

// Pendant votre travail, si vous appuyez la commande marqueur deux fois, ZIP vous visualise un résumé des commandes et vous laisse modifier le marqueur horizontal, le marqueur vertical, la tabulation, la dimension de la page et la marge de l'imprimante. Afin de modifier par exemple la dimension de la page à 47, tapez :

```
/
P
47
```

Pour utiliser les commandes ci-dessous, tapez en premier la commande marqueur que vous avez installée puis le caractère.

H est la commande de dessin ou d'effacement de la ligne horizontale. Pour l'utiliser, tapez : /H.
Si le curseur se trouve sur n'importe quel caractère à l'exception du caractère que vous utilisez comme marqueur horizontal, il vous sera dessiné une ligne à partir de la position du curseur jusqu'au côté droit de votre écran. Si le curseur se trouve sur le caractère utilisé comme marqueur horizontal, tout ce qui se trouve à droite du curseur sera effacé.

V est la commande de dessin ou d'effacement de ligne verticale. Pour l'utiliser, tapez : /V.
Si le curseur se trouve sur n'importe quel caractère à l'exception de celui que vous avez utilisé comme marqueur vertical, il vous sera dessiné une ligne à partir de la position du curseur jusqu'à la fin de votre page. Si vous avez posé une valeur plus grande que 23 comme dimension de page, la ligne sera étendue au-delà de la fin de votre écran. Si votre curseur se trouve sur un caractère utilisé comme marqueur vertical, tous les caractères partant du curseur seront effacés jusqu'à la dernière ligne de votre page.
Un symbole « + » sera automatiquement effacé ou dessiné à l'intersection des lignes horizontales et verticales.

/H/H et /V/V peuvent être utilisés respectivement pour effacer les rangs et colonnes.

T et B (précédés par un caractère de commande) déplacent le curseur au début ou à la fin de l'écran en gardant le curseur sur la même colonne.

M déplace le curseur au centre de la ligne.

N montre l'écran suivant : visualisation de votre travail écran par écran jusqu'à la dernière ligne de votre page.

P montre l'écran précédent en revenant en arrière écran par écran jusqu'à ce qu'il trouve Rang 0 au début de votre écran.

F montre le premier écran; c'est un moyen rapide pour vous positionner au début.

L montre le dernier écran; c'est un moyen rapide pour vous positionner à la fin du format.

I insère un espace à la position du curseur. Tout caractère poussé au-delà du côté droit de l'écran est perdu.

D efface le caractère sous le curseur.

A ajoute une ligne à la position du curseur poussant ainsi le texte vers le bas. Tout texte poussé au-delà de la dimension de la page posée est perdu.

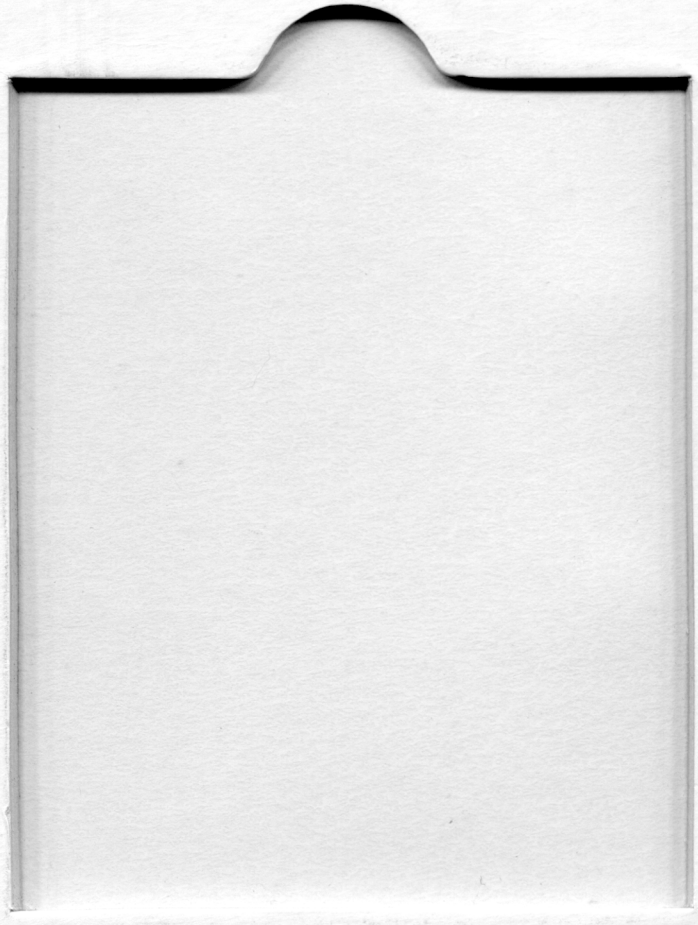
K efface le curseur ligne et termine le texte.

E efface votre fichier de travail et vous permet de commencer sur un autre fichier.

S sauvegarde votre fichier sous <nom>.ZIP et <nom>.ZPR, puis écrit le fichier de commande pour votre écran ou pour l'impression <nom>.CMD ou <nom>.FMT (selon votre choix).

Q vous ramène au système.

Imprimé en France
Décembre 1985
DM Impressions
1, place de l'Église
BP 38 92215 Saint-Cloud
(1) 47 71 25 90



dBASE II
AMSTRAD
Version 2.4
Serie No A 100999

GARANTIE

Le Produit est garanti contre tous vices de fabrication durant 30 jours après l'acquisition et sera échangé gratuitement en cas de défaut.
Étant donné le prix réduit du Produit, il ne sera pas fourni de support technique pour aide à son utilisation.

OUVRAGE COMPLÉMENTAIRE

Initiation aux bases de données pour micro-ordinateurs.
Application à dBASE II pour AMSTRAD CPC 6128 et PCW 8256 (par Robert A. Byers).

328 pages, 250 FF TTC franco de port.

Envoyer votre commande et règlement à :
La Commande Électronique
7, rue des Prias,
27920 SAINT-PIERRE-DE-BAILLEUL

Prix 790 FF TTC
(Manuel d'utilisation et disquette)

La Commande Electronique

ISBN : 2-905350-05-9

ASHTON-TATE

dBASE II POUR AMSTRAD

Manuel d'utilisation et disquette

dBASE™ II

Ashton-Tate

SUR AMSTRAD
CPC 6128 et PCW 8256

CONTRAT A NOUS RETOURNER OBLIGATOIREMENT

CONTRAT DE CESSION LIMITÉE DE LICENCE D'EXPLOITATION DE dBASE II A UN UTILISATEUR FINAL

Article 1. — Le client achète le droit d'exploitation de **dBASE II** limitée à un micro-ordinateur. Il s'interdit formellement d'utiliser le logiciel sur plusieurs micro-ordinateurs, de le vendre, ou de le fournir à des tiers même sans paiement. En aucune manière le client ne peut se considérer comme ayant une propriété quelconque sur **dBASE II** si ce n'est comme contractant d'exploitation sur une seule machine.

Article 2. — La garantie s'applique durant 30 jours à dater de la date d'achat. Le contrat présent doit nous avoir été retourné signé. La garantie s'applique à tout vice de fabrication. La disquette sera changée gratuitement.

Article 3. — La responsabilité d'Ashton-Tate ne saurait être engagée des suites de l'utilisation de son produit. En tout état de cause, sa responsabilité ne saurait excéder le montant de cession de la licence d'exploitation.

Article 4. — Dans le cas où l'une des parties engage une action judiciaire, la partie qui gagne son procès sera en droit d'obtenir le remboursement de ses frais et honoraires dans les mesures raisonnables.

Article 5. — Le présent contrat sera régi par le Droit International et interprété conformément à ce même droit. Dans le cas où des dispositions quelconques du présent Contrat seraient déclarées nulles par un tribunal compétent, ou en vertu de la législation applicable, la validité des autres dispositions du présent contrat ne s'en trouvera pas affectée pour autant.

Article 6. — Ce contrat constitue l'engagement complet de licence entre vous et Ashton-Tate.

Article 7. — En dépit de tous les soins apportés à la réalisation de ce logiciel et du manuel associé, étant donné la complexité du produit, il peut exister des erreurs mineures qui ne mettent pas en cause la validité générale du produit selon les usages généraux de la profession.

Article 8. — En rapport avec le prix auquel ce logiciel est vendu, il ne sera pas fourni d'assistance technique pour son utilisation. Il est conseillé au néophyte de s'adresser à une société de formation spécialisée.

Le/la soussigné(e) reconnaît qu'en signant le présent document, il/elle devient partie de ce même Contrat d'Utilisateur Final et accepte d'être lié(e) par toutes les clauses, conditions et obligations de ce Contrat.

Signature de l'Utilisateur Final	Support	Numéro de série
	AMSTRAD	A 100877 VERSION 2.4

Dénomination Sociale de l'Utilisateur Final *(s'il s'agit d'une société)* : _____

Activité de votre société : _____

Nom de l'Utilisateur Final : _____ Date : _____

Adresse : _____ Téléphone : _____

Code Postal : _____ Ville : _____

Nom et adresse du vendeur : _____

_____

Date de l'achat : _____

Cachet du vendeur

En rapport avec le prix auquel ce logiciel est vendu, il ne sera pas fourni d'assistance technique pour son utilisation. Il est conseillé au néophyte de s'adresser à une société de formation spécialisée.

La Commande Electronique
7, RUE DES PRIAS — 27920 SAINT-PIERRE DE BAILLEUL

Document numérisé
avec amour par :

AMSTRAD
CPC

MÉMOIRE ÉCRITE

http://amstradcpc.fredisland.net/