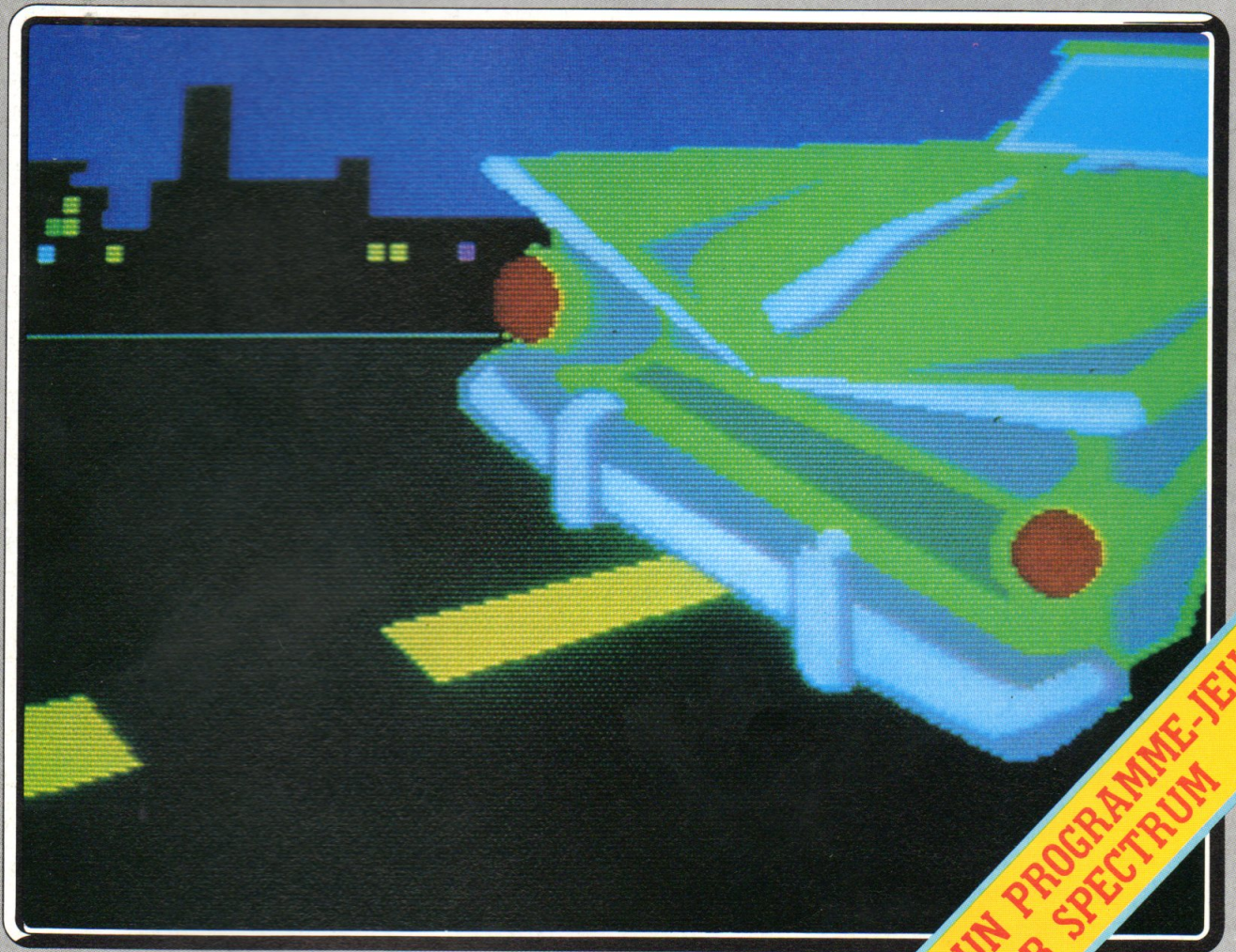
BASIC PLUS

ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS

30



FICHE : UN PROGRAMME-JEU POUR SPECTRUM

EDITIONS DU HENNIN

M 6268-30-12FF - 80FB - 4FS

BASIC PLUS

ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS

Parution hebdomadaire publiée par
les Editions du Hennin
20-22, rue de Clichy 75009 Paris.

Directeur général : Charles-Jean Pradelle.
Directeur délégué : Italo Milani.

Avec la collaboration de **Bull** et
Thomson Micro-Informatique Grand Public.

Direction éditoriale : Nathalie Perrin.

Traduction et adaptation
sous la direction de Martine Allain.

Titre original : Basic, Enciclopedia dell'Informatica dei Mini e Personal Computer.
Réalisé par le Département Grands Ouvrages d'Armando Curcio Editore, sous la direction technique de Sante Senni et la direction éditoriale de Gabriella Costarelli.

Dépôt légal : 3e trimestre 1984, ISSN : en cours.
Composition : Typelec, Paris. Imprimé en Belgique par Van Cortenbergh, Bruxelles.

Photo de couverture : Ecran de Vidéotex canadien réalisé par E. Bonnet.

Directeur de la publication : Maurice Brébart.

Administration
France : Éditions du Hennin, 20-22, rue de Clichy 75009 Paris. Tél. (1) 280.64.65.
Belgique : Femmes d'Aujourd'hui S.A., avenue Frans Van Kalken, 9 B-1070 Bruxelles, Tél. (02) 523.20.60.

Distribution
France : NMPP. Belgique : AMP S.A. Suisse : NAVILLE S.A.

Vente au numéro
Les fascicules peuvent être ontenus chez les marchands de journaux ou, à défaut, chez l'Éditeur, au prix en vigueur au moment de la commande. Pour toute commande par lettre, joindre le règlement (chèques bancaire ou postal) majoré de 10 % représentant la participation aux frais de port.

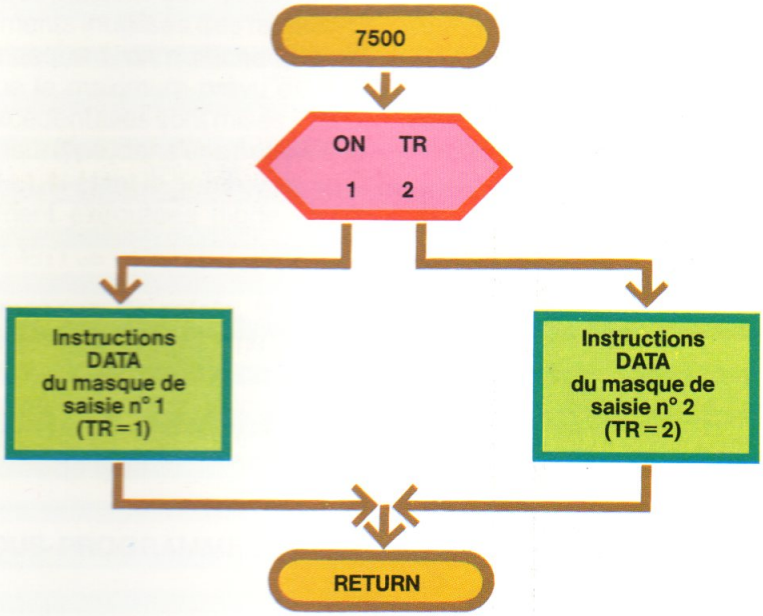
Vente par souscription
France : 72 fascicules + 6 reliures mobiles pour un prix global de 1.200 FF dont 108,00 FF de frais de participation au conditionnement et de port (pour la France métropolitaine).
Payable par prélèvement automatique en 24 mensualités de 50,00 FF. Envoi de 4 fascicules à la fois (x 18) sous étui cartonné. La première reliure mobile vous parviendra avant la parution du douzième fascicule, qui achève le premier volume. Adressez votre courrier à : AZ Distribution, 20-22, rue de Clichy, 75009 Paris.
N.B. Avec votre commande, n'envoyez pas d'argent (ni chèque, ni mandat). Toute souscription concerne l'intégralité de l'ouvrage et commence au fascicule n° 1.

Belgique : Basic Plus, avenue Frans Van Kalken, 9. B-1070 Bruxelles.

BASIC PLUS, première encyclopédie d'initiation à l'informatique et aux ordinateurs, se compose de 72 fascicules de 24 pages (+ 4 pages de couverture), destinés à être assemblés en 6 volumes de 288 pages à l'aide de reliures mobiles spécialement conçues à cet effet par l'Editeur et qui sont à la disposition de nos lecteurs chez les marchands de journaux.

En raison des vacances, BASIC PLUS ne paraîtra pas le 1er et le 15 août.
Dates de parution des numéros d'août : n° 30, le 8/8 - n° 31, le 22/8 - n° 32, le 29/8.

SOUS-PROG. D'AFFECTATION DES LIBELLES ET DES LONGUEURS DE CHAMPS



On définira de la même façon le type de chaque champ. A cet effet, on adoptera, par exemple, la convention suivante :

TC(I) = 1 pour une variable numérique ;
TC(I) = 2 pour une variable alphanumérique ;
TC(I) = 3 pour une variable faisant l'objet d'une simple présentation.

La première zone de notre exemple sera alors de type 2, tandis que toutes les autres seront de type 1. On utilise le type 3 pour définir des champs dont le contenu doit être visualisé sans que l'opérateur puisse le modifier.
Certaines applications nécessitent l'affichage de plusieurs masques de saisie successifs : c'est le cas du programme que nous allons maintenant développer. Il sera judicieux de réutiliser les mêmes variables pour chacun des masques. Il suffit alors de prévoir plusieurs groupes d'instructions DATA et d'activer sélectivement celui qui contient les données voulues. On peut, au niveau du programme d'appel, mémoriser le numéro du masque à visualiser dans une variable (appelée par exemple NM), dont la valeur déterminera, dans le sous-programme, le choix des instructions d'affectation correspondantes. Le sous-programme définit également le nom et le numéro du fichier dans lequel les données doivent être écrites (représentés respectivement par les variables NF$ et NO). L'organigramme du haut de cette page correspond à un sous-programme dans lequel on n'a prévu que deux masques de saisie (NM peut prendre la valeur 1 ou 2). Pour lui conférer un domaine d'application plus large, il suffirait d'ajouter des groupes de DATA, des ordres de lecture et, bien sûr, de modifier en conséquence l'instruction de branchement multiple. Cette routine est listée page 698.
La structure de chaque nouveau groupe de DATA devra se calquer sur celle du bloc compris entre les lignes 7800 et 7824.
La première instruction (ligne 7530) sert à définir la position du masque à visualiser (KC=2, KL=3) ; l'exécution se poursuit ensuite en fonction du numéro de masque choisi (ligne 7540).
Dans chaque cas, on initialise tout d'abord les variables caractéristiques du masque : son nombre de champs (NC) et le numéro logique du fichier associé (NO). On positionne ensuite le pointeur de DATA, et on appelle la routine 7730, qui effectuera toutes les lectures et affectations (ces instructions sont en effet communes à tous les masques). Au retour, on mémorise le nom du fichier dans

AFFECTATION DES LIBELLES ET DES LONGUEURS DE CHAMPS

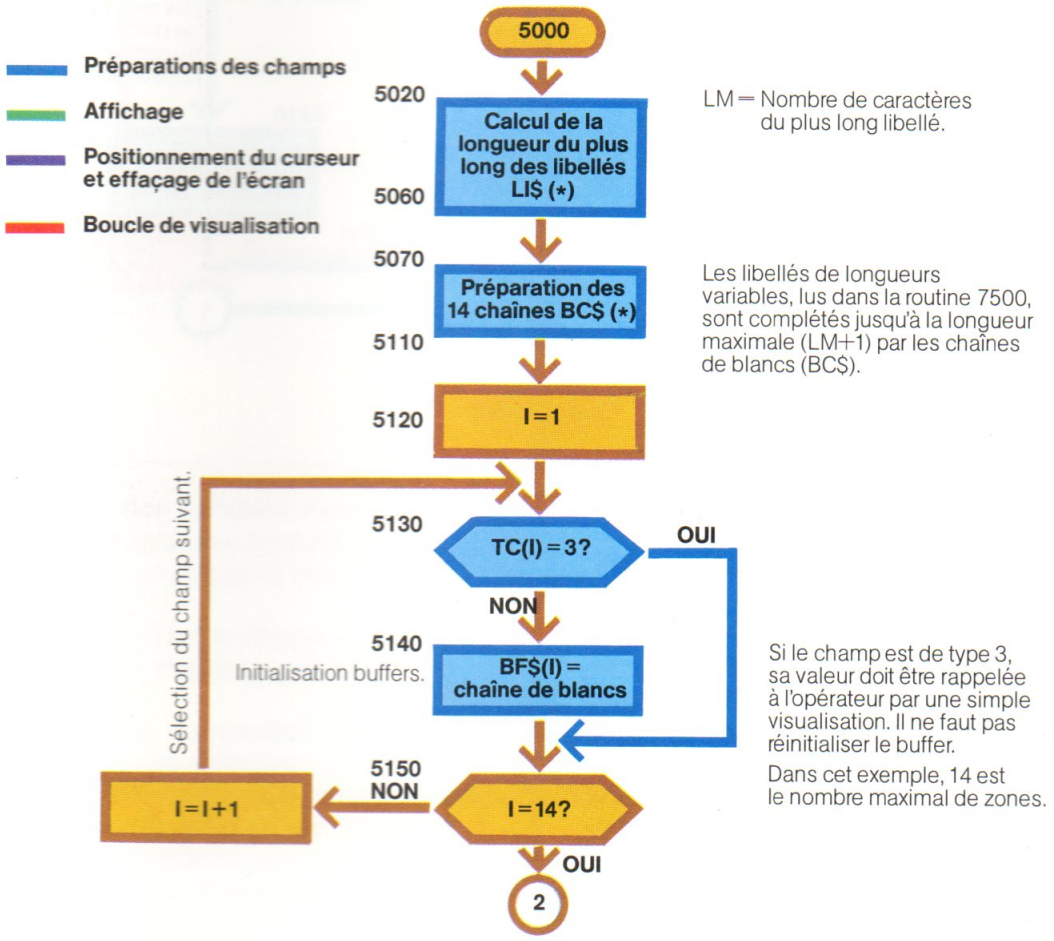
```
7500 '** SOUS-PROGRAMME DATA **
7502 '   FICHIER = ENCYCL
7504 '
7506 ' ENTREE :    NM = numero de masque
7508 ' SORTIES:    KC = position de la premiere colonne sur l'ecran
7510 '             KL = position de la premiere ligne
7512 '         NC(*) = nombre de champs
7514 '        LI$(*) = libelle des lignes
7516 '         TC(*) = type de champ
7518 '         PO(*) = pointeur vers le premier octet du champ
7520 '         DO(*) = pointeur vers le dernier octet
7522 '        NO = NM = numero logique du fichier
7524 '           NF$ = nom du fichier
7526 '            LN = nombre de caracteres d'un enregistrement du fichier
7528 '
7530 KC = 2 : KL = 3    ' Position du curseur
7540 ON NM GOTO 7570, 7640    ' Selection selon le numero de masque
7550 '
7560 ' * SI NM = 1
7570 NC(NM) = 10 : N = 10 : NO = 1    ' Initialisations
7580 RESTORE 7800                     ' Positionnement dans les DATA
7590 GOSUB 7740                       ' Lecture des DATA
7600 NF$ = " A:ACHVEN "               ' Nom du fichier
7610 GOTO 7690
7620 '
7630 ' * SI NM = 2
7640 NC(NM) = 9 : N = 9 : NO = 2
7650 RESTORE 7830
7660 GOSUB 7740
7670 NF$ = " A:MVTMAR "
7680 '
7690 N = N+1                          ' Reinitialisation des champs excedentaires
7700 FOR I = N TO 14 : LI$(I) = " " : TC(I) = 0 : PO(I) = 0 : DO(I) = 0 : NEXT I
7710 RETURN                           ' Sortie du sous-programme DATA
7720 '
7730 ' ** LECTURE DES DATA
7740 FOR I = 1 TO N : READ LI$(I) : NEXT I
7750 FOR I = 1 TO N : READ TC(I) : NEXT I
7760 FOR I = 1 TO N : READ PO(I) : NEXT I
7770 FOR I = 1 TO N : READ DO(I) : NEXT I
7780 LN = DO(N)
7790 RETURN                           ' Retour au sous-programme DATA
7795 '
7796 ' ** DATA
7798 ' * cas ou NM = 1
7800 DATA "  1 - Type marchandise "
7802 DATA "  2 - Taux T.V.A. "
7804 DATA "  3 - Unite de mesure "
7806 DATA "  4 - Cout Achat "
7808 DATA "  5 - Prix Vente "
7810 DATA "  6 - Cumul des Entrees "
7812 DATA "  7 - Valeur totale "
7814 DATA "  8 - Cumul des Sorties "
7816 DATA "  9 - Valeur totale "
7818 DATA " 10 - ............ "
7820 DATA 2!, 1!, 2!, 2!, 2!, 3!, 3!, 3!, 3!, 3!
7822 DATA 1!, 21!, 23!, 31!, 38!, 45!, 53!, 63!, 72!, 82!
7824 DATA 20!, 22!, 30!, 37!, 44!, 52!, 62!, 71!, 81!, 100!
7826 ' * cas ou NM = 2
7830 DATA " 1 - CODE MARCH. "
7832 DATA " 2 - DATE MOUVEMENT "
7834 DATA " 3 - QTE ENTREE "
7836 DATA " 4 - COUT ACHAT "
7838 DATA " 5 - QTE SORTIE "
7840 DATA " 6 - PRIX VENTE "
7842 DATA " 7 - ......... "
7844 DATA " 8 - ......... "
7846 DATA " 9 - transfert Mvt. "
7848 DATA 1!, 2!, 2!, 2!, 2!, 2!, 2!, 2!, 3!
7850 DATA 1!, 4!, 9!, 17!, 24!, 32!, 39!, 65!, 91!
7852 DATA 3!, 8!, 16!, 23!, 31!, 38!, 64!, 90!, 91!
```

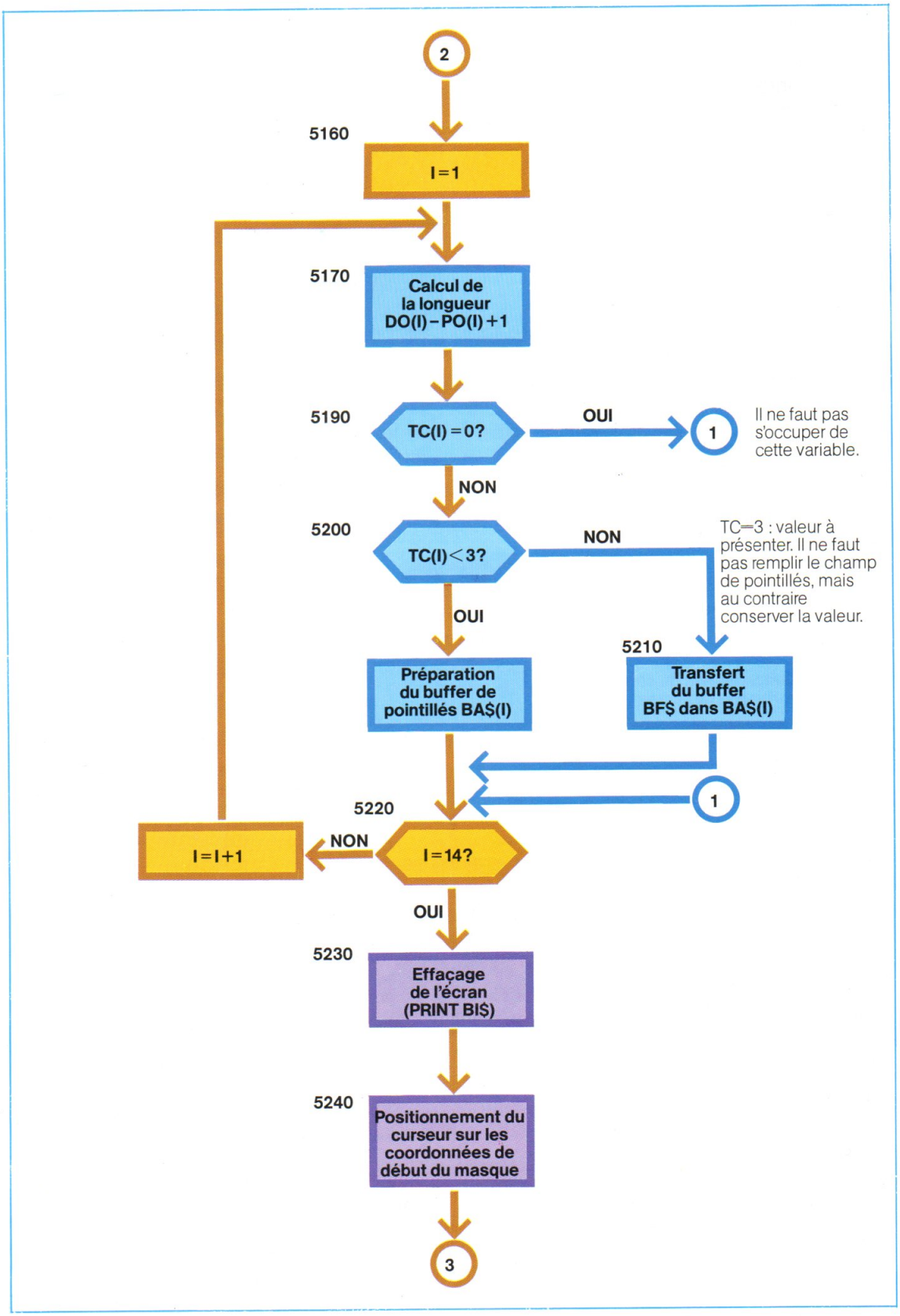
la chaîne NF$. Dans ce programme, le fichier correspondant au premier masque s'appelle ACHVEN et se trouve sur la disquette A.
Enfin, les dernières lignes (7690-7700) réinitialisent les éléments inutilisés des tableaux. Ainsi, avec le masque 1, on n'affiche que 10 zones, alors que le maximum prévu est 14. Les variables excédentaires sont mises à zéro ou à vide, selon leur type, dans une boucle qui commence à N+1, N étant le nombre de lectures effectivement exécutées (ligne 7570 pour le masque 1).

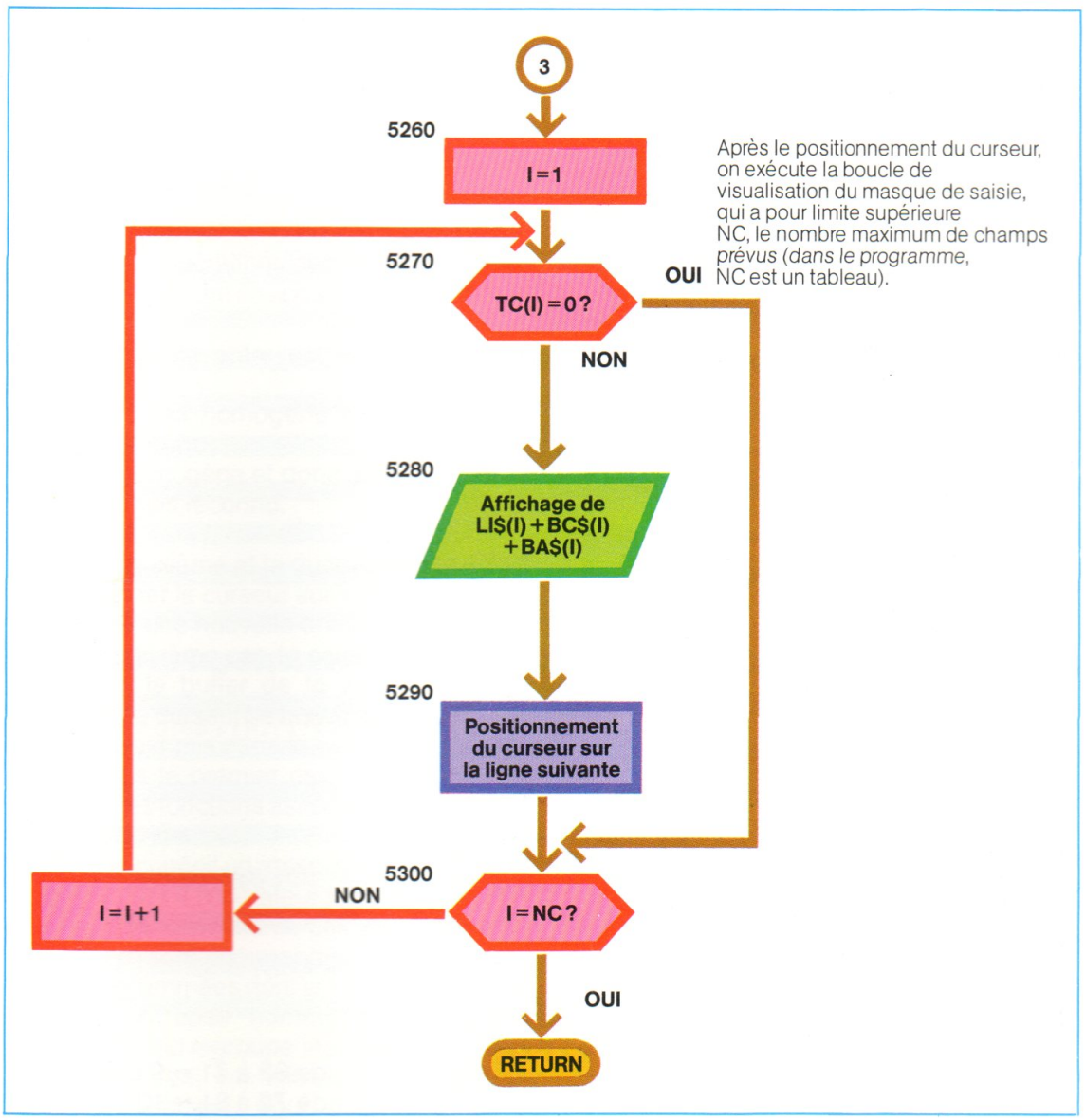
Affichage du masque de saisie. Une fois qu'on a appelé le sous-programme précédent, il faut visualiser le masque de saisie. Cette opération présente deux difficultés : le cadrage des zones et, éventuellement, la présence de variables de type 3.

Les différents libellés ne possèdent pas forcément la même longueur. Il ne faut donc pas afficher systématiquement les pointillés à la suite du libellé correspondant, car ils ne seraient alors pas alignés les uns avec les autres. Pour éviter cette mauvaise présentation, on procédera de la façon suivante : on cherchera la longueur du plus grand libellé, et on complètera chaque libellé par un nombre de blancs égal à la différence entre la longueur maximale et celle du libellé lui-même. En fait, il vaudra mieux prévoir chaque fois un espace supplémentaire, afin que les pointillés ne soient pas accolés au libellé le plus long.
L'organigramme du sous-programme de visualisation des masques de saisie commence page 699 et se poursuit pages 700 et 701. Le listing correspondant se trouve page 702.

SOUS-PROGRAMME D'AFFICHAGE DES MASQUES DE SAISIE




2
5160
I=1
5170
Calcul de la longueur DO(I)-PO(I)+1
5190
TC(I)=0?
OUI
1
Il ne faut pas s'occuper de cette variable.
NON
5200
TC(I)<3?
NON
TC=3 : valeur à présenter. Il ne faut pas remplir le champ de pointillés, mais au contraire conserver la valeur.
OUI
5210
Préparation du buffer de pointillés BA$(I)
Transfert du buffer BF$ dans BA$(I)
1
5220
I=I+1
NON
I=14?
OUI
5230
Effaçage de l'écran (PRINT BI$)
5240
Positionnement du curseur sur les coordonnées de début du masque
3



Après le positionnement du curseur, on exécute la boucle de visualisation du masque de saisie, qui a pour limite supérieure NC, le nombre maximum de champs *prévus (dans le programme,* NC est un tableau).

Certaines des variables mentionnées dans ce sous-programme sont initialisées dans le programme principal, et notamment :

BL$ = chaîne d'espaces blancs
BP$ = chaîne de pointillés

La recherche du nombre de caractères du plus long libellé s'effectue par les instructions 5020 à 5060, tandis que la boucle suivante (5070 à 5110) construit les chaînes BC$ de blancs, qui viendront compléter les différents libellés LI$. Les chaînes BC$ sont générées par prélèvement de N caractères dans BL$ (instruction 5100).

Deux formes de présentation sont prévues dans le sous-programme : l'une pour les valeurs à saisir au clavier et l'autre pour celles qui doivent seulement être affichées. En effet, le type de champ (TC), lu dans le sous-programme 7500 prend les valeurs 1, 2 ou 3. Dans les deux premiers cas, on affichera une chaîne de pointillés (BA$). Pour le type 3, au contraire, la chaîne BA$ à afficher devra contenir la valeur lue et affectée au buffer BF$. Ainsi, pour obtenir la visualisation d'un masque de saisie, on appellera successivement le module 7500 qui contient les DATA, puis le module 5000. Pour visualiser le premier masque, on exécutera les instructions suivantes :

AFFICHAGE DES MASQUES DE SAISIE

```
5000 ' AFFICHAGE DES MASQUES DE SAISIE
5002 ' ** Sous-programme visuel **
5004 '       FICHIER = ENCYCL
5006 '
5008 ' ENTREES : LI$(*),TC(*),PO(*),DO(*),NC(*),KC,KL de la routine 7500
5010 '           BF$(*)                                  de la routine 1200
5012 '           NH                                      numero de masque
5014 ' SORTIE  : LC(*)=longueur des champs
5016 '           BF$(*)=buffer des champs, corrige a la bonne longueur
5020 LM=0
5030 FOR I=1 TO 14              ' Recherche de la longueur maximale
5040 N=LEN(LI$(I))              ' des libellés
5050 IF N>LM THEN LM=N
5060 NEXT I
5065 '
5070 FOR I=1 TO 14              ' Chaines de blancs pour completer
5080 N=LEN(LI$(I))              ' les libellés
5090 NB=LM-N+1
5100 BC$(I)=LEFT$(BL$,NB)
5110 NEXT I
5115 '
5120 FOR I=1 TO 14
5130 IF TC(I)=0 OR TC(I)=3 GOTO 5150
5140 BF$(I)=BL$                 ' Reinitialisation du buffer
5150 NEXT I
5155 '
5160 FOR I=1 TO 14
5170 N=DO(I)-PO(I)+1            ' Longueur du champ
5180 LC(I)=N
5190 IF TC(I)=0 GOTO 5220
5200 IF TC(I)<3 THEN BA$(I)=LEFT$(BP$,N) : BF$(I)=LEFT$(BF$(I),N) : GOTO 5220
5210 BA$(I)=BF$(I)              ' Transfert de la donnee a afficher
5220 NEXT I
5225 '
5230 PRINT BI$              ' Efface l'ecran et emet un bip
5235 '
5240 X=KC : Y=KL : GOSUB 6900       ' Position curseur
5250 NE=NC(NM)                ' Nombre de lignes a afficher
5260 FOR I=1 TO NE
5270 IF TC(I)=0 GOTO 5300
5280 PRINT LI$(I)+BC$(I)+BA$(I)
5290 Y=Y+1 : GOSUB 6900             Incrementation d'une ligne
5300 NEXT I
5310 RETURN
```

```
NM=1
GOSUB 7500
GOSUB 5000
```

Dans le module 7500, les cinq dernières zones du premier masque sont définies en type 3. Or, ces champs n'ont pas encore fait l'objet d'une affectation. Ils peuvent donc être vides, ou contenir des valeurs erronées provenant de traitements antérieurs. Si l'on veut afficher des valeurs correctes, il faut les initialiser avant d'appeler le sous-programme 5000. Ces zones portent les numéros 6, 7, 8, 9 et 10. Leurs libellés figurent aux lignes 7810 à 7818, et voici leurs longueurs respectives :

champ 6	de 45 à 52 = 8 caractères
champ 7	de 53 à 62 = 10 caractères
champ 8	de 63 à 71 = 9 caractères
champ 9	de 72 à 81 = 10 caractères
champ 10	de 82 à 100 = 19 caractères

Si l'on transfère des chaînes de caractères dans les buffers BF$ correspondants, celles-ci seront visualisées en même temps que le reste du masque de saisie. Ecrivons, par exemple :

```
NM=1
GOSUB 7500
BF$(9)="0123456789"
GOSUB 5000
```

L'exécution de ces instructions entraîne l'affichage de la chaîne "0123456789" sur la neuvième ligne du masque.

Entrée des données. Après le module 5000, on parvient en phase de saisie des données. Les opérations à exécuter sont les suivantes :

— positionnement du curseur ;
— lecture d'un caractère (sans écho) ;
— contrôle du caractère lu.

La suite des opérations dépend de l'issue de ce contrôle. Quatre cas peuvent se présenter :

1 - le caractère entré est un caractère de commande ;
2 - il n'est pas homogène avec le type de variable attendu ;
3 - il est homogène et donc accepté ;
4 - il n'est pas reconnu.

Dans le deuxième et le quatrième cas, il faut repositionner le curseur sur le même point et demander une nouvelle entrée (correction). Dans le troisième cas, le caractère est transféré dans le buffer de la variable ; si, par exemple, le curseur se trouve sur la ligne 4, le caractère est mis dans le quatrième buffer. Enfin, dans le premier cas, la machine doit exécuter les fonctions associées au caractère de commande.
Quand on conçoit un masque de saisie, il faut toujours offrir à l'opérateur la possibilité d'effectuer des corrections. Les déplacements du curseur sont commandés par quatre touches programmées dont la frappe déclenche l'envoi d'un code numérique spécial. Le tableau TF, qui regroupe les différents codes de commande, est initialisé dans le programme principal. Pour déterminer si la touche frappée est associée à une fonction, il suffit de balayer ce tableau. Si le nombre généré s'y trouve, on va se brancher à un endroit du programme défini par sa valeur (lignes 6450-6462). Ici, les codes associés aux touches programmées sont 10, 8, 9 et 11 (pour les déplacements), ainsi que 16, 17 et 13. Ces touches de fonction se répartissent en deux groupes : le premier commande des fonctions internes au masque de saisie et le second permet de sortir du sous-programme. La gestion des touches du premier groupe présente quelques difficultés. Prenons, par exemple, le cas des deux touches de déplacement vertical. Voici les opérations qu'elles réalisent et les codes correspondants :

— déplacement d'une ligne vers le haut, code 11 (ligne 6590) ;
— déplacement d'une ligne vers le bas, code 10 (ligne 6470).

On peut résumer ainsi les opérations effectuées par ces touches :

— incrémentation (ou, selon le sens du déplacement, décrémentation) du compteur de lignes CL ;
— incrémentation (ou décrémentation) de la coordonnée verticale Y ;
— remise à zéro du compteur de caractères (CC) ;
— positionnement horizontal en début de zone (X=X0).

Supposons, par exemple, que le curseur se trouve sur la ligne 3 (CL=3). Les données qu'on entre à ce moment sont transférées dans le buffer numéro 3 (BF$(3)). L'activation du code 11 devra entraîner :

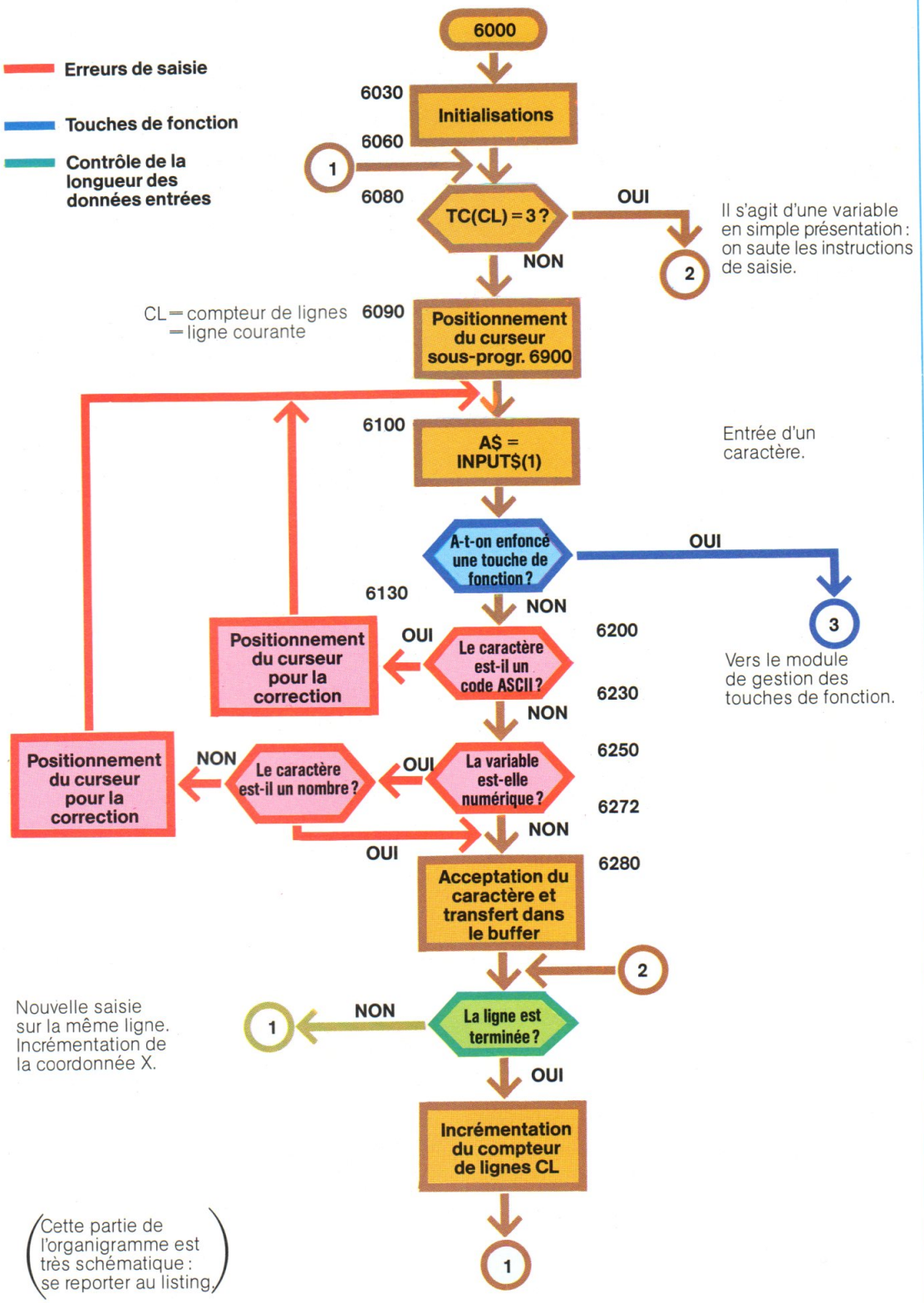
— la décrémentation du compteur de lignes (CL=3–1=2) qui doit désormais pointer vers le buffer numéro 2 ;
— la décrémentation de la position (Y=Y–1) pour que le curseur revienne sur la ligne précédente ;
— la remise à zéro du compteur de caractères (CC=0). Ainsi, le curseur ira se placer au début de la zone réservée à la donnée, et tout se passera comme si aucun caractère n'avait encore été transféré dans le buffer numéro 2.

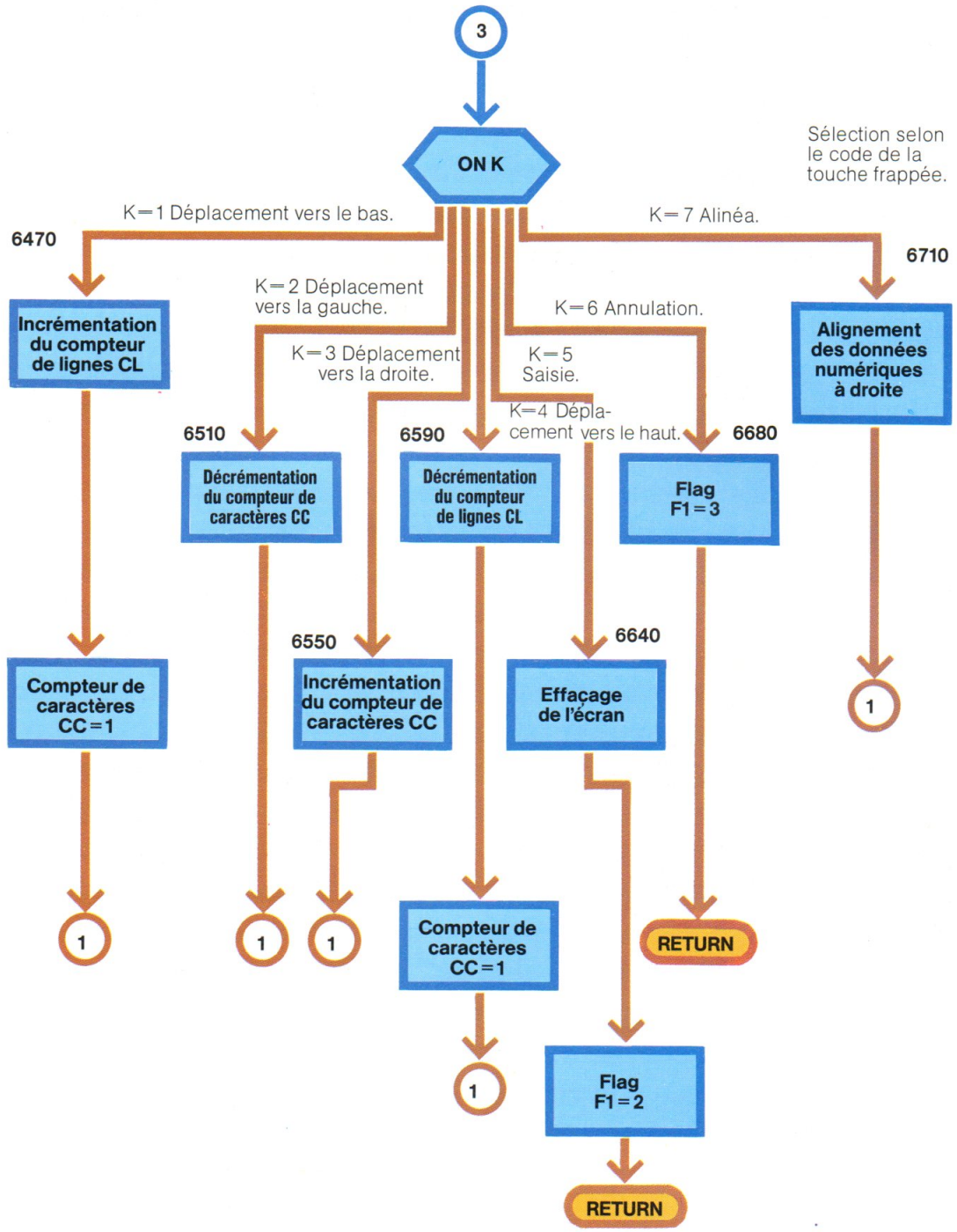
Les autres touches de fonction internes commandent les opérations suivantes :

— déplacement d'une position vers la droite, code 9, ligne 6550 ;
— déplacement d'une position vers la gauche, code 8, ligne 6510 ;
— justification, code 13, ligne 6710 (touche RT).

Le mécanisme des fonctions de déplacement horizontal est semblable à celui des fonctions de déplacement vertical, mais le compteur de lignes reste inchangé, alors que le compteur de caractères est incrémenté (ou décrémenté).

SOUS-PROGRAMME D'UTILISATION DES MASQUES DE SAISIE





On remarquera qu'il n'est possible de sortir du sous-programme qu'en frappant les touches de fonction associées à la saisie ou à l'annulation. Selon le cas, le flag est positionné à 2 ou à 3. Sa valeur est ensuite testée dans le programme principal, où elle sert d'aiguillage vers la saisie (2) ou l'annulation suivie d'une nouvelle saisie (3).

UTILISATION DES MASQUES DE SAISIE

```
6000 '
6002 ' UTILISATION DES MASQUES DE SAISIE
6003 '
6004 ' ** SOUS-PROGRAMME SAISIE **
6005 '    FICHIER = ENCYCL
6006 '
6007 ' ENTREES :      TC(*),KC,KL              de la routine 7500
6008 '               LM,LC(*),BF$(*)          de la routine 5000
6009 '               NM = numero du masque
6010 '               TF(20) = codes des touches programmées (pr. principal)
6011 '    code       ASCII      Signification standard      fonction programmée
6012 '     10        CTRL+J     line feed                   curseur bas
6013 '      8        CTRL+H     backspace                   curseur gauche
6014 '      9        CTRL+I     horizontal tabulation       curseur droite
6015 '     11        CTRL+K     vertical tabulation         curseur haut
6016 '     16        CTRL+P     DEL                         saisie des donnees
6017 '     17        CTRL+Q     DC4                         annulation des donnees
6018 '     13        CTRL+M     carriage return             justification
6020 ' SORTIES :   F1=flag  saisie/annulation
6021 '
6022 ' CL = compteur de lignes = ligne courante
6023 ' CC = compteur de caractères
6024 '
6025 '
6030 X0=KC+LM+1 : Y0=KL
6040 CL=1 : CC=0
6050 F1=0                    ' Flag de saisie/annulation
6060 X=X0 : Y=Y0
6065 '
6070 IF TC(CL)=0 THEN GOSUB 6850  ' Ligne de masque inexistante : on cherche la suivante
6080 IF TC(CL)=3 GOTO 6400        ' Valeur affichee : pas de saisie
6090 GOSUB 6900                   ' Position du curseur en (X,Y)
6095 '
6100 A$=INPUT$(1)                 ' Saisie d'un caractere sans echo
6110 GOSUB 6900
6120 V=ASC(A$)                    ' Conversion
6125 '
6130 K=0                          ' La touche frappée est-elle programmee?
6140 FOR I=1 TO 20
6150 IF TF(I)=0 GOTO 6170
6160 IF V=TF(I) THEN K=I
6170 NEXT I
6180 IF K<>0 GOTO 6440            ' Gestion des touches de fonction
6190 PRINT A$
6195 '
6200 ' Controle du caractere frappé
6210 IF V<123 AND V>31 GOTO 6240      ' Caractere ASCII
6220 PRINT CHR$(7) : GOSUB 6950       ' On signale l'erreur a l'operateur
6230 GOTO 6090                        ' Vers une nouvelle saisie
6235 '
6240 ' Controle du type de caractere
6250 IF TC(CL)=2 GOTO 6280            ' Caractere alpha
6260 IF V>47 AND V<58 GOTO 6280       ' Caractere numerique correct
6270 PRINT CHR$(7)
6272 GOTO 6090
6275 '
6280 ' Caractere accepte : insertion
6290 N=LC(CL)                         ' Longueur totale du champ
6300 NG=CC                            ' Nombre de caracteres deja memorises
6310 CC=CC+1                          ' Caractere en cours de saisie
6320 BG$=LEFT$(BF$(CL),NG)
6330 ND=N-CC                          ' Nombre de caracteres encore a saisir
6340 IF ND=0 THEN BD$="" : GOTO 6360
6350 BD$=RIGHT$(BF$(CL),ND)
6360 BF$(CL)=BG$+A$+BF$
```

```
6370 IF CC=N THEN CL=CL+1 : CC=0 : Y=Y+1 : X=X0-1      ' La ligne est complete
6380 IF CL>NC(NM) THEN CL=1 : Y=Y0                     ' Retour en debut de masque
6390 X=Y+1                                             ' Deplacement vers la droite
6392 GOTO 6070
6395 '
6400 ' Cas d'une valeur affichee a ne pas modifier (TC(CL)=3)
6405 '
6410 CL=CL+1 : Y=Y+1              ' On passe simplement a la ligne
6420 IF CL>NC(NM) THEN CL=1 : Y=Y0 : CC=0 : X=X0       ' Retour en debut de masque
6430 GOTO 6070
6434 '
6436 '
6440 ' ** Gestion des touches programmees **
6445 '
6450 ON K GOTO 6470, 6510, 6550, 6590, 6640, 6680, 6710
6455 '
6456 ' Erreur : code fonction non attribue
6460 ER=5 : PU=5
6462 RETURN          ' Sortie du sous-programme
6465 '
6470 ' * U=10        deplacement vers le bas
6480 CL=CL+1 : CC=0
6485 X=X0 : Y=Y+1
6490 IF CL>NC(NM) THEN CL=1 : Y=1          ' Retour en debut de masque
6500 GOTO 6070
6505 '
6510 ' * U=8         deplacement vers la gauche
6520 IF CC=0 THEN GOSUB 6950 : GOTO 6090       ' On ne fait rien
6530 CC=CC-1 : X=X-1
6540 GOTO 6090
6545 '
6550 ' * U=9         deplacement vers la droite
6560 IF CC>LC(CL) THEN GOSUB 6950 : GOTO 6090
6570 CC=CC+1 : X=X+1
6580 GOTO 6090
6585 '
6590 ' * U=11        deplacement vers le haut
6600 IF CL=1 THEN GOSUB 6950 : GOTO 6090
6610 CL=CL-1 : Y=Y-1
6615 CC=0 : X=X0
6620 IF TC(CL)=0 GOTO 6600
6625 ' Si la ligne du masque n'est pas valide : on remonte encore d'une ligne
6626 ' Lorsqu'on se deplace en descendant, la recherche d'une ligne valide
6627 ' s'effectue dans la subroutine 6850
6630 GOTO 6080
6635 '
6640 ' * U=16        touche de saisie
6650 PRINT BI$
6660 F1=2                    ' Flag de saisie
6670 RETURN                  ' Sortie du sous-programme
6675 '
6680 ' * U=17        touche d'annulation
6690 F1=2                    ' Flag d'annulation
6700 RETURN                  ' Sortie
6705 '
6710 ' * U=13        justification a droite des donnees numeriques
6720 IF TC(CL)=2 GOTO 6810           ' Donnee alpha
6730 A$=""
6740 N=LC(CL)-CC             ' Nombre de zeros a ajouter a gauche
6750 IF N=0 GOTO 6810
6760 FOR I=1 TO N
6764 A$=A$+"0"
6767 NEXT I
6770 X=X0 : GOSUB 6900
6780 IF CC<>0 THEN A$=A$+LEFT$(BF$(CL),CC)
6790 PRINT A$
6800 BF$(CL)=A$
6805 '
```

```
6810  CL=CL+1 : CC=0
6815  X=X0 : Y=Y+1
6820  IF CL>NC(CL) THEN CL=1 : Y=Y0
6830  GOTO 6070
6835  '
6840  '
6845  '
6850  ' ** Routine de recherche d'une ligne de masque valide **
6855  '
6860  CC=0 : X=X0
6870  CL=CL+1
6880  IF CL>NC(CL) THEN CL=1 : Y=Y0        ' Retour en debut de masque
6890  IF TC(CL=0 GOTO 6870                 ' On continue la recherche
6895  RETURN
6896  '
6897  '
6898  '
6900  ' ** Routine de positionnement du curseur **
6905  '
6910  PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
6920  RETURN
6925  '
6930  '
6940  '
6950  ' ** Routine de signalement des erreurs **
6960  '
6970  PRINT CHR$(7)
6980  PRINT CHR$(27)+CHR$(61)+CHR$(51)+CHR$(51);    ' Curseur en 20,20
6990  INPUT "ERREUR : tapez un caractere ";S$
6995  PRINT CHR$(27)+CHR$(61)+CHR$(51)+CHR$(51); : PRINT " 30 "
6996  RETURN
```

Quant à la touche RT, on l'utilise pour aligner les données. En effet, lors de la saisie, le curseur se place au début de chaque zone, puis se décale d'une position vers la droite chaque fois qu'un caractère est saisi. Les données s'*alignent donc toutes à gauche*, quel que soit leur type, alors que les valeurs numériques doivent être cadrées à droite.
Les instructions 6710 à 6830 testent la nature de la variable (numérique ou alphanumérique selon la valeur de TC) et, éventuellement, rectifient l'alignement en ajoutant une chaîne de zéros à gauche du champ numérique. Elles sont activées par la frappe de la touche de justification.
Enfin, la fonction de positionnement du curseur est traitée séparément, dans un sous-programme d'une seule ligne (6900).
Quant aux fonctions du deuxième groupe, nous avons vu qu'elles permettent de sortir du module de visualisation. Au nombre de deux, elles se différencient par la valeur que prend le flag F1, selon qu'on frappe la touche associée à l'une ou à l'autre. On peut ainsi savoir, dans le programme principal, si les données sont correctes (touche 1, F1=2) ou si l'opérateur a décidé leur annulation (touche 2, F1=3).

Pour définir d'autres fonctions en programmant de nouvelles touches, il suffit de faire figurer leur code dans le tableau TF et d'ajouter au sous-programme de saisie les instructions nécessaires.

La phase de mémorisation sur disque

Lorsque la condition F1=2 est vérifiée (données correctes), les données doivent être écrites sur le disque. On peut encore concevoir un sous-programme d'emploi général.
Voici les quatre opérations qu'on peut effectuer sur un fichier :

- ouverture et définition de la zone tampon pour E/S ;
- lecture des enregistrements ;
- écriture des enregistrements ;
- fermeture du fichier.

Une instruction non paramétrée (CLOSE n) suffit à fermer un fichier. Il vaut donc mieux la placer dans le programme principal, plutôt que l'inclure dans la routine.
Les trois autres fonctions sont, au contraire, paramétrées. On devra donc, avant d'appeler

le sous-programme 1200 de gestion du disque, définir les paramètres suivants :

OP = flag indiquant l'opération désirée (ouverture, lecture, écriture) ;
NF$ = nom du fichier à traiter ;
NO = numéro d'unité logique de ce fichier ;
NE = numéro d'enregistrement, en lecture ou en écriture ;
LN = longueur de cet enregistrement (tableau calculé dans la routine 7500).

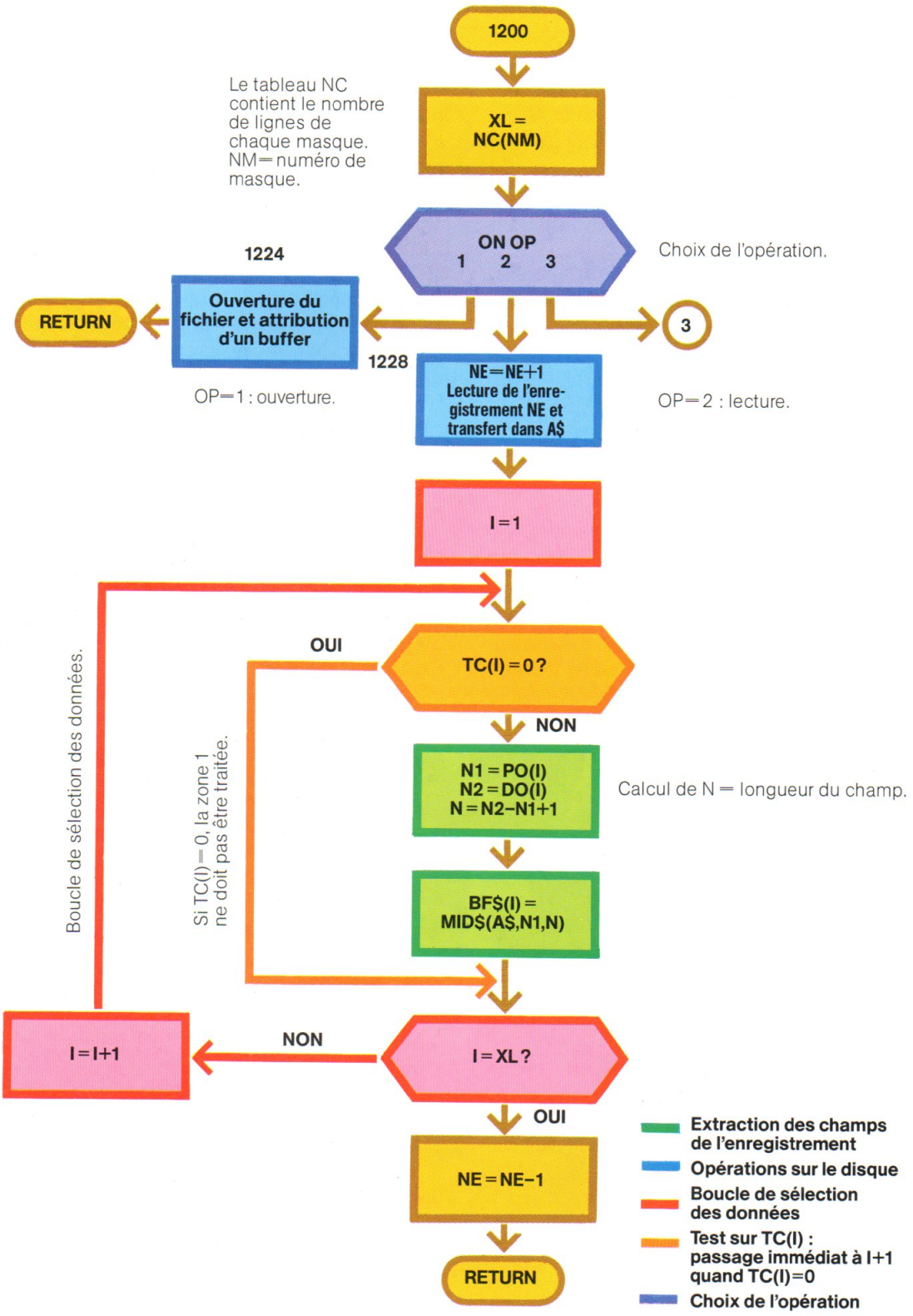
En outre, il faut dimensionner un tableau alphanumérique (DX$ par exemple), qui jouera le rôle de buffer d'entrées/sorties, et dont chaque élément sera associé au fichier de même numéro (voir ligne 1226). Ainsi, on posera NO=3, LN=100, NF$="A:ESSAI" et OP=1 pour ouvrir, sur le disque A, un fichier de nom ESSAI, de mémoire tampon DX$(3) et contenant des enregistrements de 100 caractères. Les opérations de lecture et d'écriture présentent une certaine complexité, et nous allons les étudier de façon plus détaillée.

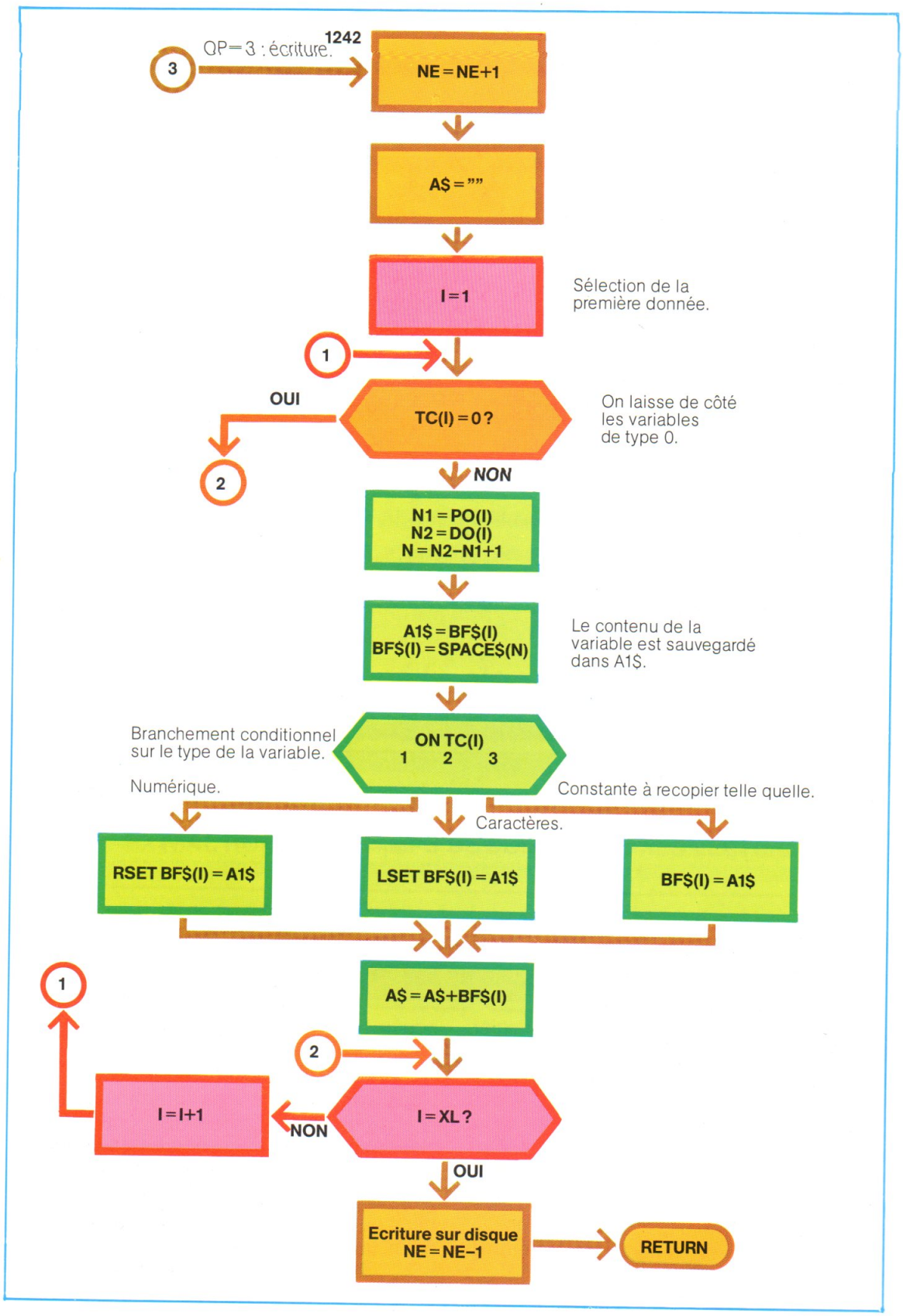
L'écriture (instructions 1242 à 1272). Conformément à la méthode que nous avons plusieurs fois mise en œuvre, le numéro du dernier enregistrement écrit est mémorisé dans le premier enregistrement, qui ne représente donc pas une donnée.
Ce décalage risquerait de provoquer des erreurs si l'utilisateur devait le contrôler lui-même, en ajoutant 1 avant de lire ou d'écrire une donnée et en s'en abstenant pour le premier enregistrement. On préférera donc le gérer automatiquement à l'intérieur même du sous-programme. Pour cela, on donnera le numéro 0 au premier enregistrement (NE=0), ce qui permettra d'ajouter systématiquement 1 pour accéder à n'importe quel enregistrement. Il faut naturellement soustraire 1 à NE avant de sortir du sous-programme, afin de lui rendre sa valeur au moment de l'appel.
Avant d'écrire un enregistrement, on doit regrouper les zones BF$ qui contiennent les données saisies à l'aide du masque.
On prépare les enregistrements de la façon suivante : après avoir cadré le contenu des buffers de saisie (BF$), on les concatène dans un ordre déterminé de manière à créer une chaîne A$, comme le montre le schéma de la page 713 et les instructions 1246 à 1263. On transfère ensuite cette chaîne A$ dans le

Techniciens vérifiant le fonctionnement du matériel dans un centre informatique.

K. Reese/Marka

SOUS-PROGRAMME PARAMETRE DE GESTION DU DISQUE




3
QP=3 : écriture.
1242
NE=NE+1
A$=""
I=1
Sélection de la première donnée.
1
OUI
TC(I)=0?
On laisse de côté les variables de type 0.
2
NON
N1=PO(I)
N2=DO(I)
N=N2-N1+1
A1$=BF$(I)
BF$(I)=SPACE$(N)
Le contenu de la variable est sauvegardé dans A1$.
Branchement conditionnel sur le type de la variable.
ON TC(I)
1 2 3
Numérique.
Caractères.
Constante à recopier telle quelle.
RSET BF$(I)=A1$
LSET BF$(I)=A1$
BF$(I)=A1$
A$=A$+BF$(I)
1
2
I=I+1
NON
I=XL?
OUI
Ecriture sur disque
NE=NE-1
RETURN

SOUS-PROGRAMME PARAMETRE DE GESTION DU DISQUE

```
1200 ' ** SOUS-PROGRAMME DISQUE **
1201 '    Lecture/ecriture sur disque
1202 '    FICHIER = ENCYCL
1203 '    ENTREES :
1204 ' NE = numero de l'enregistrement
1205 ' NO = numero du fichier = numero du buffer ; NF$ = nom du fichier
1206 ' OP = code operation
1207 '     1 : ouverture
1208 '     2 : lecture
1209 '     3 : ecriture
1210 ' LN,PO(*),DO(*),NC(*),TC(*)    parametres de la routine 7500
1218 '
1219 '
1220 XL=NC(NM)          ' Nombre de lignes du masque
1221 '
1222 ON OP GOTO 1224,1228,1242
1223 '
1224 ' **** OUVERTURE ****
1226 OPEN "R",NO,NF$,LN : FIELD NO,LN AS DX$(NO) : RETURN
1227 '
1228 ' **** LECTURE ****
1229 ' NE=NE+1     ' Incrementation pour obtenir le veritable numero d'enregistrement
1230 GET NO,NE : A$=DX$(NO)
1232 FOR I=1 TO XL : IF TC(I)=0 GOTO 1236
1234 N1=PO(I) : N2=DO(I) : N=N2-N1+1 : BF$(I)=MID$(A$,N1,N)      ' Extraction
1236 NEXT I
1237 NE=NE-1          ' Decrementation pour retrouver la valeur initiale
1238 RETURN
1240 '
1242 ' **** ECRITURE ****
1243 NE=NE+1
1244 A$=""
1246 FOR I=1 TO XL : IF TC(I)=0 THEN GOTO 1264
1248 N1=PO(I) : N2=DO(I) : N=N2-N1+1
1250 A1$=BF$(I)
1252 BF$(I)=SPACE$(N)
1254 ON TC(I) GOTO 1256,1258,1260
1256 RSET BF$(I)=A1$ : GOTO 1262
1258 RSET BF$(I)=A1$ : GOTO 1262
1260 BF$(I)=A1$
1262 A$=A$+BF$(I)
1263 NEXT I
1264 '
1265 PRINT "A=";A$ : S$=INPUT$(1)
1268 LSET DX$(NO)=A$
1270 PUT NO,NE
1271 NE=NE-1
1272 RETURN
```

buffer d'entrées/sorties associé au fichier (ligne 1268), et ce dernier sera recopié sur disque à la place qui lui revient (ligne 1270).

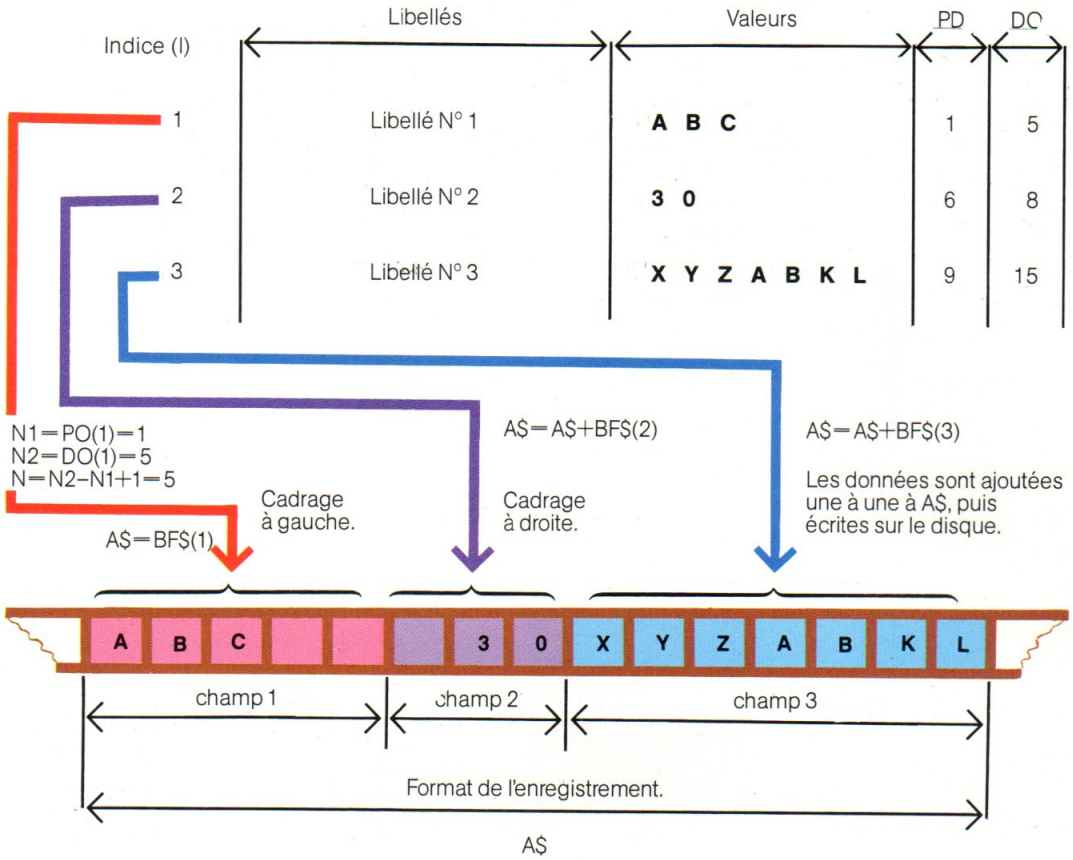
La lecture (lignes 1228 à 1238). Son mécanisme est exactement l'inverse du précédent. Après avoir extrait du disque l'enregistrement cherché (sans oublier d'ajouter 1), on transfère le contenu du buffer d'E/S du fichier dans la chaîne A$, puis on répartit les différents champs dans leurs buffers respectifs (BF$). Ce processus d'extraction est illustré par le schéma du bas de la page 713.

Le programme principal

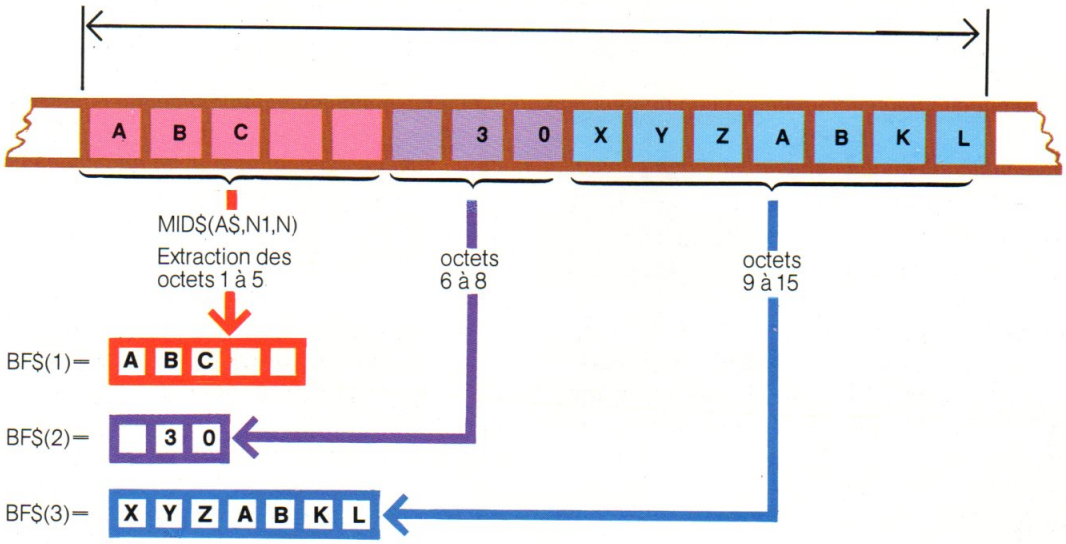
Nous avons conçu un programme principal afin de tester nos différentes routines. Son organigramme se trouve page 704 et son listing pages 715 et 716, suivi de celui du sous-programme 9000 de programmation des touches de fonction. Les codes correspondants à ces touches sont initialisés par les instructions 240 et 250 de DATA. On en a défini vingt, en prévision de développements futurs.

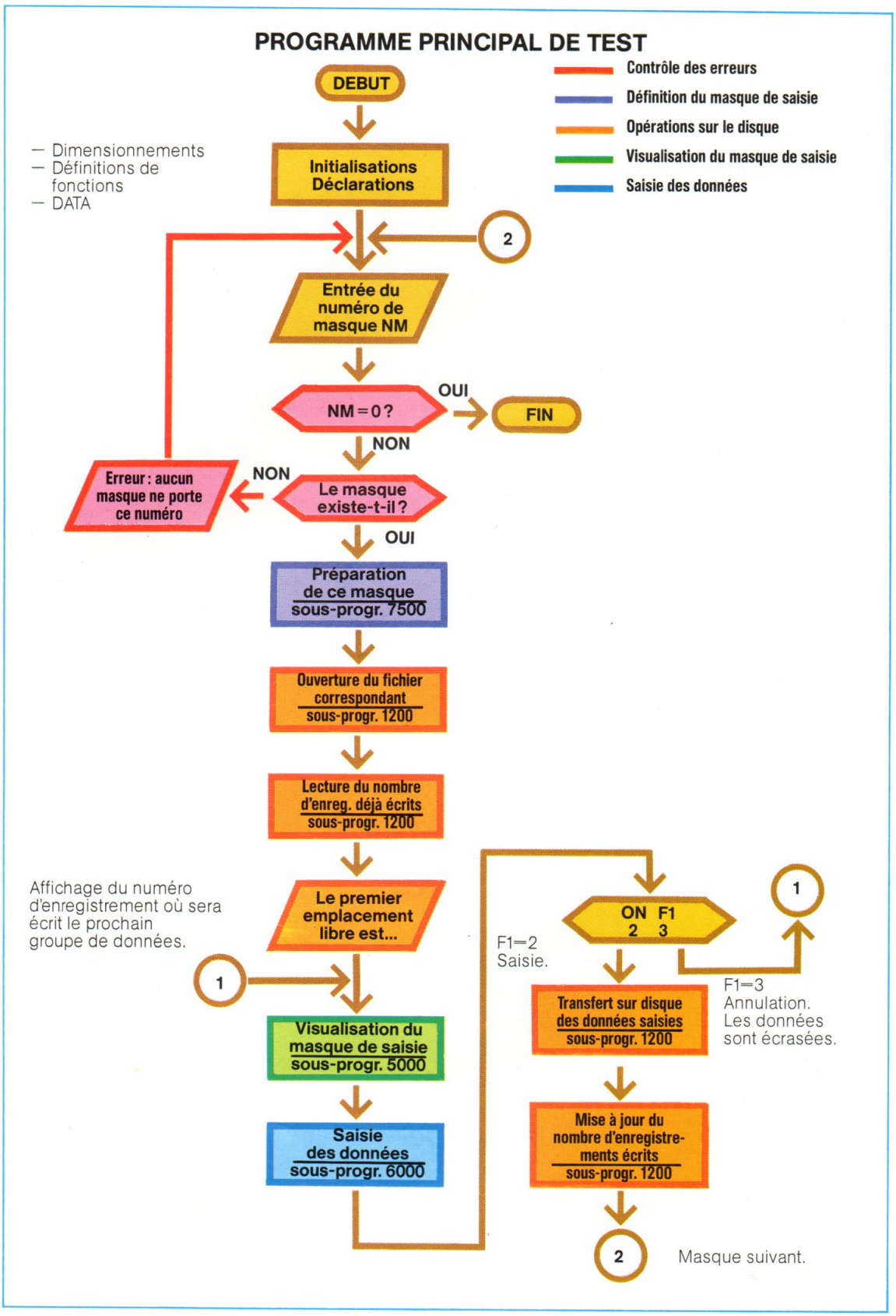
PREPARATION D'UN ENREGISTREMENT AVANT ECRITURE

On construit un enregistrement en concaténant les différents champs par l'instruction A$=A$+BF$(I).



EXTRACTION DES CHAMPS DE DONNEES APRES LECTURE




PROGRAMME PRINCIPAL DE TEST
Contrôle des erreurs
Définition du masque de saisie
Opérations sur le disque
Visualisation du masque de saisie
Saisie des données
DEBUT
– Dimensionnements
– Définitions de fonctions
– DATA
Initialisations Déclarations
2
Entrée du numéro de masque NM
NM = 0 ?
OUI
FIN
NON
Erreur : aucun masque ne porte ce numéro
NON
Le masque existe-t-il ?
OUI
Préparation de ce masque sous-progr. 7500
Ouverture du fichier correspondant sous-progr. 1200
Lecture du nombre d'enreg. déjà écrits sous-progr. 1200
Affichage du numéro d'enregistrement où sera écrit le prochain groupe de données.
Le premier emplacement libre est...
1
Visualisation du masque de saisie sous-progr. 5000
Saisie des données sous-progr. 6000
ON F1 2 3
F1=2 Saisie.
1
F1=3 Annulation. Les données sont écrasées.
Transfert sur disque des données saisies sous-progr. 1200
Mise à jour du nombre d'enregistrements écrits sous-progr. 1200
2
Masque suivant.

PROGRAMME PRINCIPAL DE TEST

```
10 ' ** PROGRAMME PRINCIPAL **
20 '    FICHIER = ENCYCL
30 OPTION BASE 1
40 '
50 ' **** DEFINITIONS ****
60 ' TF(*)  = codes des touches de fonction (6000)
70 ' DX$(*) = buffer d'E/S du fichier disque N° * (1200)
80 ' TC(*)  = type du champ * (7500)
90 ' DO(*)  = dernier octet du champ * (7500)
100 ' PO(*)  = premier octet du champ * (7500)
110 ' LI$(*) = libelle du champ * (7500)
120 ' NC(*)  = nombre de champs du masque * (7500)
130 ' BF$(*) = buffers de saisie (6000)
140 ' BC$(*) = chaine de blancs pour completer les libelles (5000)
150 ' LC(*)  = longueur du champ * (5000)
160 ' BA$(*) = buffer de pointilles ou de donnee a presenter (5000)
170 '
180 '
190 '
200 '
210 DEFINT A-Z
230 DIM TF(20)          ' Tableau de 20 codes de fonctions
240 FOR I=1 TO 20 : READ TF(I) : NEXT I
250 DATA 10!,8!,9!,11!,16!,17!,13!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,
255 ' Seules 7 touches sont utilisees. Pour en rajouter, il faut modifier le DATA
260 DIM DX$(4)
270 DIM TC(14),DO(14),PO(14),LC(14),NC(14)
280 DIM LI$(14),BF$(14),BC$(14),BA$(14),
300 '
310 '
320 '
330 ' * FONCTIONS *
340 '
350 BL$="                                                  "
360 BP$=".................................................."
370 BI$=CHR$(27)+"+"+CHR$
390 '
400 GOSUB 9000       ' Programmation des touches de fonction
410 PRINT BI$        ' Efface l'ecran et emet un bip
420 '
500 ' ** SELECTION DU MASQUE DE SAISIE **
510 INPUT "Taper le numero du masque desire ";NM
520 IF NM=0 THEN GOTO 900          ' Sortie
530 IF NM<1 OR NM>2 THEN PRINT "ERREUR : ce masque n'existe pas"
540 IF NM<1 OR NM>2 THEN S$=INPUT$(1) : GOTO 510
550 '
560 ' ** PREPARATION DU MASQUE **
570 GOSUB 7500          ' Chargement des DATA
575 '
580 ' ** OUVERTURE DU FICHIER **
590 OP=1 : GOSUB 1200
600 ' Lecture de l'enregistrement 1
610 NE=0 : OP=2
620 GOSUB 1200
630 DE=VAL(BF$(1))       ' Dernier enregistrement ecrit
640 NE=DE+1              ' Premier emplacement libre
650 PRINT "Le premier emplacement libre est : ";NE
660 PRINT "Appuyer sur une touche pour continuer" : S$=INPUT$(1)
665 '
670 ' ** VISUALISATION DU MASQUE DE SAISIE **
680 GOSUB 5000
685 '
690 ' ** SAISIE DES DONNEES **
700 GOSUB 6000
```

```
710 ' ** BRANCHEMENT CONDITIONNEL (VALEUR DE F1)
720 K=F1-1          'On soustrait 1 : valeurs possibles 1 ou 2 au lieu de 2 et 3
725 '
730 ON K GOTO 740,870
735 '
740 '*F1=2 : SAISIE*
750 OP=3            'operation=ecriture
760 GOSUB           'Ecriture des donnees dans l'enregistrement NE
770 '
780 'Remise a jour du premier enregistrement
790 A$=STR$(NE)             'Conversion en ASCII du nombre d'enregistrements
800 NE=0                    'Le premier enregistrement est designe par 0
810 '                        On utilise le buffer BF$(1) pour transferer NE
820 N=DO(1)-PO(1)+1         'Nombre de caracteres de BF$(1)
830 BF$(1)=SPACE$(N)        'Mise a blanc de BF$(1)
840 RSET BF$(1)=A$          'Cadrage a droite du contenu de BF$(1)
850 OP=3 : GOSUB 1200       'Ecriture
860 GOTO 640                'Entree suivante
865 '
870 ' ** ANNULATION DES DONNEES **
880 PRINT BI$
885 GOTO 640
890 '
900 ' ** SORTIE **
910 PRINT BI$
920 PRINT "*** AU REVOIR ***"
930 END
```

```
9000 ' ** PROGRAMMATION DES TOUCHES DE FONCTION **
9010 '
9020 '    FORME DU CURSEUR
9030 PRINT CHR$(27)+"c4AA"
9040 '
9050 '
9060 'Les touches de deplacement du curseur (codes 10, 8, 12 et 11)
9065 'sont gerees dans la routine 6000, ainsi que celle de justification (code 13)
9066 'On code ici 8 touches (b a i) en leur attribuant les valeurs 16 a 23.
9067 'Les deux premieres sont utilisees dans le programme
9068 'et les suivantes laissees a la disposition de l'utilisateur
9069 'qui devra bien sur les ajouter au DATA et definir leurs fonctions.
9070 A$=CHR$(27)+CHR$(46)    'Premiere etape de la programmation des
9080 '                         touches de fonction
9090 B=98                    'Code de la premiere touche de fonction (98 = code ASCII de b)
9100 I1=49                  '49 et 48 ASCII=10 HEXA car 49=1 et 48=0
9110 I2=48                  'et 10 HEXA equivaut a 16 decimal
9120 FOR I=1 TO 8           'Boucle de programmation des touches b a i
9122 'La valeur decimale 16 est associee a la premiere touche de fonction (n° 98)
9130 C$=CHR$(I1)+CHR$(I2)   'Code a associer a la touche I (1 a 8)
9140 D$=CHR$(B)             'Selection de la touche
9150 E$=A$+D$+C$            'Chaine contenant les codes de commande
9160 PRINT E$               'La touche est programmee
9170 B=B+1                  'Touche suivante
9180 I2=I2+1                'Code suivant (incremente de 1)
9190 NEXT I
9200 RETURN
```

La compilation

Il est beaucoup plus commode de travailler en mode interprété pendant les phases de conception et de mise au point des programmes Basic. Par contre, l'exécution d'un programme sous le contrôle de l'Interpréteur est très longue pour trois raisons fondamentales :

- à chaque instruction d'appel (GOTO ou GOSUB), l'Interpréteur doit lire entièrement le programme jusqu'à la ligne où il doit poursuivre l'exécution ;
- quand un nom de variable est mentionné, l'Intepréteur doit chercher dans la table des variables l'emplacement de mémoire où se trouve sa valeur ;
- l'Interpréteur doit traduire chaque instruction en langage machine avant de l'exécuter. En particulier, dans une boucle, il décodera les instructions Basic à chaque itération. Or, si la traduction d'une instruction s'effectue très rapidement, on finit par atteindre des temps d'exécution assez longs quand on exécute de nombreux passages.

Tous ces inconvénients sont liés à la manière dont fonctionne l'Interpréteur.
On les évitera en **compilant** les programmes après leur mise au point. Le Compilateur se charge de traduire les instructions en binaire et de convertir les noms de variables en adresses absolues. L'exécution d'un programme compilé est donc bien plus rapide (4 à 10 fois). De plus, le Compilateur effectue un diagnostic global et relève éventuellement des erreurs qui ont pu échapper à l'Interpréteur.
C'est notamment le cas pour les instructions de branchement conditionnel. Si la ligne à laquelle renvoie l'instruction de saut n'existe pas, l'Interpréteur ne s'en apercevra (et, donc, ne le signalera) qu'au moment d'effectuer le saut, c'est-à-dire seulement si la condition est vérifiée.
L'erreur pourra donc passer inaperçue pendant la mise au point, et ne se révéler qu'après la mise en service du programme, ce qui risquera de fausser les données en cours de traitement, ou même d'entraîner leur perte.
En revanche, la compilation inclut un contrôle systématique des adresses, qui permet de déceler les erreurs éventuelles de ce type.

Le schéma de la page 718 retrace la succession des opérations d'écriture, de compilation et d'exécution d'un programme Basic.
Nous allons maintenant décrire en détail ces différentes étapes.

La préparation

Le Compilateur ne peut opérer que sur des programmes sauvegardés en ASCII. Il faut donc, avant tout, s'assurer que le programme source a bien été mémorisé sous cette forme : il doit être sauvegardé sur disquette (depuis la mémoire centrale) avec l'option "A". Ainsi, pour un programme baptisé ESSAI, on procédera de la façon suivante :

```
LOAD "A:ESSAI"
SAVE "A:ESSAI",A
```

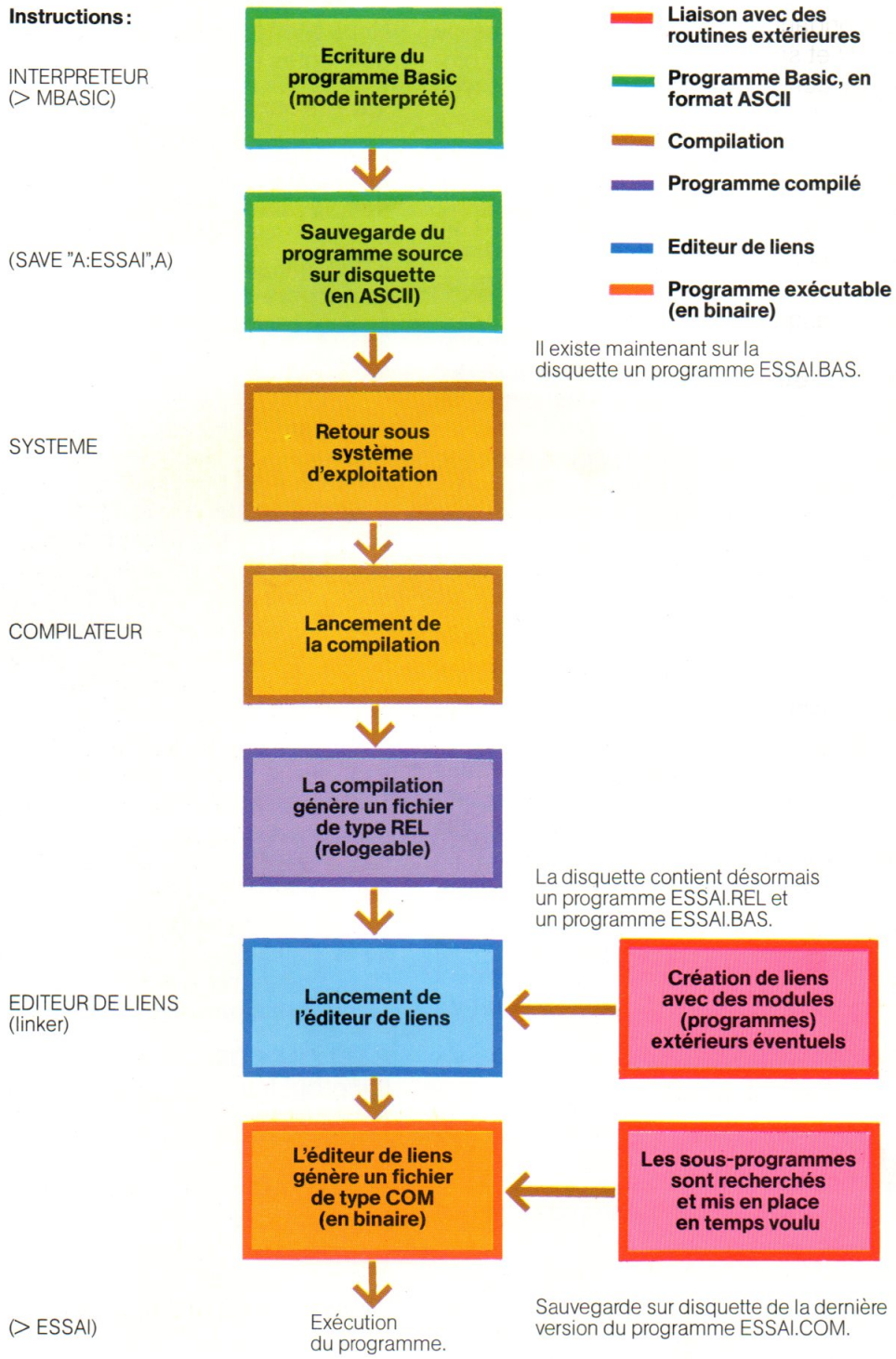
En outre, il est préférable – toujours pour raccourcir le temps d'exécution – de vérifier que les indices de toutes les boucles sont définis comme entiers, grâce à l'instruction DEFINT ou en spécifiant l'attribut % (écrire DEFINT I revient à faire suivre I du symbole %).
Enfin, il est nécessaire que tous les programmes (utilitaires du système ou non) qui seront appelés pendant la compilation, puis pendant l'« édition de liens »* (linkage) se trouvent sur la disquette A. C'est, en effet, sur l'unité A que travaille le Compilateur ; s'il est possible d'effectuer certaines opérations sur l'autre unité, il vaut mieux que tout se passe sur la même disquette. De toute façon, rien n'empêche de transférer le programme après sa compilation.
Les programmes indispensables à la compilation sont énumérés ci-dessous.

BASCOM.COM. C'est un programme système qui traduit le Basic en Assembleur. Il fournit une version compilée du programme source.

BRUN.COM. Il contient la plupart des routines utilitaires appelées par le programme d'application. Utilisé pendant l'exécution de celui-ci, il doit être présent dès la première étape afin que les liens nécessaires puissent être mis en place.

* Nous expliquerons un peu plus loin en quoi consiste l'édition de liens entre des programmes ou des parties de programmes. Considérons pour l'instant que ce terme désigne une série d'opérations qui relient des modules compilés séparément.

AVANT, PENDANT ET APRES LA COMPILATION



La disquette contient donc au moins trois versions du programme :
ESSAI.BAS = Programme source en Basic ;
ESSAI.REL = Programme compilé ;
ESSAI.COM = Programme exécutable en binaire.

Sammie et l'automobile

La conception d'une automobile doit réaliser le compromis le plus satisfaisant entre des contraintes d'ordre technico-économiques souvent antagonistes. L'équilibre est très difficile à trouver, car il faut respecter des normes de plus en plus nombreuses, répondre aux exigences d'un marché en constante évolution, tout en assurant aux véhicules d'excellentes performances.

La complexité du travail de conception ne se pose plus en termes de technologie, mais plutôt d'une multiplicité d'éléments à concilier pour offrir au consommateur un éventail de services de plus en plus diversifiés.

En effet, le véhicule familial ne représente plus aujourd'hui un simple moyen de transport utilitaire. Il doit répondre à une foule de besoins plus ou moins subjectifs.

Une bonne voiture aura une charge utile élevée et un habitacle spacieux. De ligne séduisante, elle consommera peu et sera facile à conduire pour les automobilistes peu expérimentés. Sa production, son entretien et sa réparation seront simples et peu coûteux.

Sur le plan commercial, ses ventes devront assurer rapidement les bénéfices nécessaires au financement de nouveaux modèles.

Beaucoup de temps s'écoule entre la conception d'un nouveau modèle et sa commercialisation, et il n'est pas rare que les ingénieurs et les responsables du marketing doivent imaginer les caractéristiques techniques de produits qui ne seront pas mis sur le marché avant une dizaine d'années. Ces prévisions sont suivies d'un ensemble d'études complexes et interdépendantes, depuis la réalisation de maquettes, de modèles, de prototypes, de moules et de machines-outils, jusqu'à la mise en place des équipements de production. Ce n'est qu'au terme de ce long processus que la production en série peut commencer.

On comprend donc l'intérêt des ***systèmes de CAO*** *(Conception Assistée par Ordinateur), qui réduisent considérablement les délais de conception et facilitent le recours aux techniques d'analyse structurelle, dynamique et vibratoire, qui permettent d'améliorer le produit et, par conséquent, de donner plus grande satisfaction à l'utilisateur.*

Avec les systèmes de CAO, on peut notamment rejeter tout de suite de mauvaises solutions, dont les défauts n'apparaissaient autrefois qu'après la construction des prototypes, alors qu'on manquait de temps ou d'argent pour résoudre les ultimes problèmes découverts. Les dernières étapes de la conception et les systèmes d'expérimentation non informatisés absorbaient d'énormes investissements.

De nombreux logiciels disponibles aujourd'hui rendent possible la conception de ***maquettes virtuelles tridimensionnelles*** *(3D). Ils permettent de définir simplement des formes géométriques à l'aide d'un langage évolué.*

On peut combiner à volonté ces éléments géométriques dans l'espace, pour représenter les objets les plus complexes, qu'il s'agisse d'un bâtiment, d'une pièce mécanique de moteur, d'une chaîne de production, du fuselage d'un avion, de la carcasse d'une automobile ou de n'importe quelle autre structure. Citons pour mémoire les systèmes français EUCLID (Matra-Datavision), CATIA (Dassault-Systèmes), UNISURF (Renault-Automatismes) et SISTRID (SNIAS).

Le logiciel de CAO américain SAMMIE offre la particularité intéressante de pouvoir représenter un personnage humain, et de simuler ses déplacements, facilitant ainsi toutes les études d'ergonomie en vogue actuellement.

Nous avons choisi l'exemple concret de la conception d'un nouveau modèle de voiture pour mieux illustrer cette possibilité.

Le dessinateur commence par résumer les études préliminaires dans un croquis où figurent les mesures les plus significatives (longueur totale, largeur, hauteur) et les formes proposées pour chaque surface.

A partir de ces quelques données disponibles immédiatement, on peut matérialiser l'idée initiale par un modèle à trois dimensions. On se contente, dans un premier temps, de représenter les surfaces courbes de la carosserie par un assemblage de carreaux plans. Cette opération, relativement rapide, demande environ 30 % de la durée totale de la phase de conception. Ainsi, même s'il ne dispose encore que de données extrêmement sommaires, le concepteur est déjà en mesure de se faire une idée relativement précises du nouveau véhicule.

On fait ensuite « tourner » la maquette sur elle-même, afin de pouvoir l'examiner en perspective sous tous ses angles ; cette étude, avec les techniques traditionnelles, était un luxe exigeant des heures de travail sur la planche à dessin.

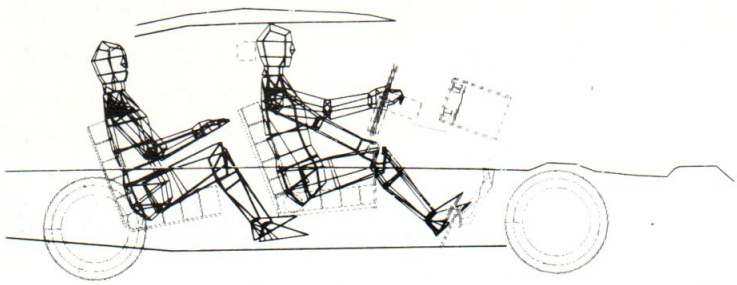
Après avoir défini la carosserie du véhicule, on commence à s'occuper de l'habitacle. Le concepteur doit alors tenter de réutiliser le plus grand nombre possible d'éléments ayant déjà servi sur les précédents modèles. On pourra, dans certains cas, reprendre les sièges d'une voiture antérieure et les adapter aux dimensions de la nouvelle carroserie. Il suffira, pour cela, de rechercher dans la ***base de données*** *de SAMMIE les maquettes créées auparavant. On réduit non seulement la durée mais le coût des études.*

Il n'est évidemment pas possible d'appliquer cette méthode à tous les éléments de l'habitacle. La disposition ergonomique des appareils du tableau de bord soulève des difficultés particulières, car le conducteur doit pouvoir les consulter, à tout moment, d'un seul coup d'œil et sans effort.

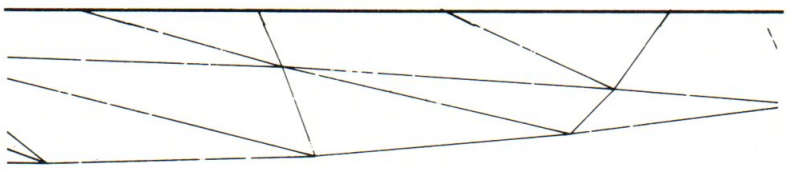
Il faut donc les placer à la fois le plus haut et le plus loin possible du conducteur. Cette nécessité est, bien sûr, en contradiction avec le besoin de visibilité et les contraintes de la géométrie extérieure du véhicule.

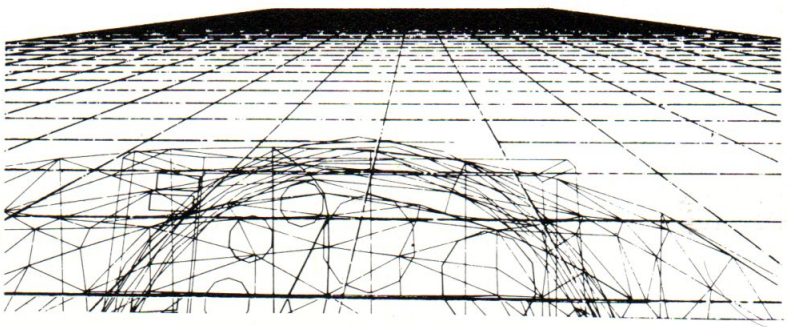
On peut résoudre ce genre de problèmes en recourant aux méthodes classiques mais la CAO offre une toute autre efficacité.

SAMMIE autorise une véritable ***« modélisation sous contraintes »,*** *qui prend en compte des données comme l'occupation de l'espace, les intersections et tangences autorisées, etc.*

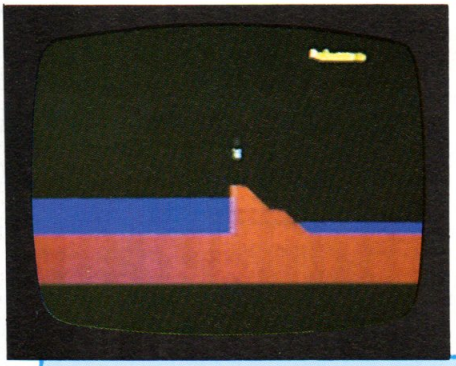


Les logiciels de CAO permettent aujourd'hui la conception rationnelle et rapide des structures les plus complexes.





Ces images ont été réalisées par une table traçante contrôlée par le logiciel SAMMIE. Elles illustrent deux étapes de la conception de l'habitacle d'un véhicule de tourisme. L'ingénieur peut s'assurer que les sièges, le volant et le tableau de bord se placent correctement les uns par rapport aux autres (en haut) et que la visibilité est satisfaisante (ci-contre). On aperçoit le tableau de bord au premier plan.



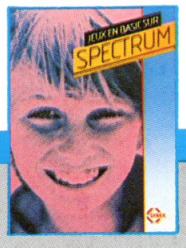
BRISEURS DE BARRAGES

Un jeu pour SPECTRUM

Durant votre vol de retour, après une mission en territoire ennemi, vous avez des problèmes de moteur et vous perdez de l'altitude. Le seul endroit sûr pour vous poser est une rivière. Malheureusement il y a un barrage sur votre chemin et vous devez le faire sauter. Utilisez la touche <O> pour larguer vos bombes.

```
  10 REM Briseurs de barrages
  20 GO SUB 9000: LET rec=0
  30 GO SUB 8000
  40 GO SUB 7000
  50 FOR v=1 TO 15
  55 PRINT AT v-1,0; PAPER 8;"
   "
  60 FOR h=0 TO 31
  70 PRINT AT v,h; INK 6; PAPER
8;"▄▄"
  80 IF SCREEN$ (v,h+6)<>" " THE
N GO TO 1000
  90 IF INKEY$="0" THEN GO SUB 2
00
  95 LET sc=sc+1
 100 BEEP (.008 AND ch=1)+(.01 A
ND ch=0),0
 105 IF ch=0 THEN GO TO 120
 110 IF (SCREEN$ (13,16)=" " AND
 SCREEN$ (13,17)=" " AND SCREEN$
 (13,18)=" ") THEN LET ch=0: PRI
NT AT 13,0;                 ";AT
 14,20; INK 2; PAPER 1;"▙
  ";AT 13,16; PAPER 0;"  "
 180 NEXT h
 190 NEXT v
 195 GO TO 1500
 200 PRINT AT v+1,h; INK 5; PAPE
R 8;"▀"
 210 PRINT AT v,h; PAPER 8; INK
6;"▄▄"
 220 LET h=h+1: IF h>30 THEN LET
 v=v+1: LET h=0
 240 PRINT AT v+1,h; INK 5; PAPE
R 8;"▲"
 250 PRINT AT v,h; PAPER 8; INK
6;"▄▄"
 260 LET h=h+1: IF h>31 THEN LET
 v=v+1: LET h=0
 275 PRINT AT v+2,h; PAPER 8;" "
 280 LET b1=h
 290 FOR b=v+1 TO 14
 295 BEEP .01,b
 300 PRINT AT b,b1; PAPER 8; INK
 5;" ";AT b+1,b1;"▼"
 310 PRINT AT v,h; INK 6; PAPER
8;"▄▄"
 320 LET h=h+1: IF h>30 THEN LET
 v=v+1: LET h=0
 330 IF SCREEN$ (b+2,b1)<>" " TH
EN GO TO 500
 335 PRINT AT v-1,31;" "
 340 NEXT b
 350 PRINT AT b,b1; PAPER 8;" "
 360 LET h=h-1
 380 RETURN
 500 FOR b=b TO b+1
 510 PRINT AT b,b1; PAPER 8;" ";
AT b+1,b1; INK 5;"▼"
 520 NEXT b
 524 BEEP .005,-b
 525 LET h=h-1: PRINT AT b,b1; P
APER 8;" "
 530 RETURN
1000 FOR a=v TO 15
1010 PRINT AT a,h+1; PAPER 1;"
  ";AT a+1,h+1; INK 6;"▄▄"
1020 BEEP .5,-a: NEXT a
1030 GO TO 1510
1500 LET sc=sc+100: PRINT AT 0,1
2;"TERMINE"
1510 PRINT AT 5,10,"Score : ";sc
1520 IF sc>rec THEN LET rec=sc
1530 PRINT AT 10,6;"Record : ";r
ec
1540 INPUT "Tapez ENTER pour rej
ouer "; LINE a$: GO TO 30
7000 CLS
7010 PRINT AT 13,0; PAPER 1;"
                "
7020 PRINT AT 14,0; PAPER 1;"
                "
7030 PRINT AT 15,0; PAPER 1;"
                "
7040 PRINT AT 12,16; INK 2;"▙";
AT 13,16; PAPER 1;"▄▄"; PAPER 0;
"▙";AT 14,16; PAPER 1;"▄▄▄▄"; PA
PER 0;"▙";AT 15,16; PAPER 1;"▄▄▄
▄▄"
7050 FOR a=1 TO 4
7060 PRINT PAPER 2;"
                "
7070 NEXT a
7990 RETURN
8000 BORDER 0: PAPER 0: INK 9: C
LS
8010 LET sc=0
8020 LET ch=1
8990 RETURN
9000 FOR a=USR "a" TO USR "i"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 56,60,63,63,63,7,0,0
9040 DATA 0,0,128,255,255,255,25
5,0
9050 DATA 0,0,63,255,255,255,255
,0
9060 DATA 0,0,224,248,248,248,24
0,0
9070 DATA 0,0,220,126,126,220,0,
0
9080 DATA 8,8,28,254,30,14,0,0
9090 DATA 36,60,24,60,60,24,0,0
9100 DATA 128,192,224,240,248,25
2,254,255
9110 DATA 254,255,255,255,255,25
5,255,255
9990 REM a b c d e f g h i
9999 REM ▙ ▄ ▄ ▗ ▀ ▲ ▼ ▙ ▄
```

Tiré de «Jeux en BASIC sur SPECTRUM» de Peter Shaw, Editions SYBEX, Réf. 276, Ft 16 × 22, 96 p., 49 F



Cher lecteur,

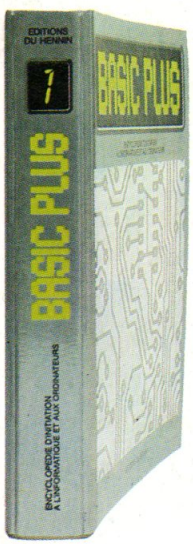
Depuis quelques mois vous trouvez chez votre marchand de journaux BASIC PLUS car vous êtes conscients comme nous, qu'il faut au plus vite apprivoiser ce nouveau langage que représente l'informatique.

Nous vous remercions de la confiance que vous accordez à notre publication et voulons vous donner quelques conseils pour que vous puissiez assembler au mieux, semaine après semaine, les fascicules de votre encyclopédie.

CHAQUE SEMAINE : 24 PAGES DE BASIC PLUS

- Comme vous l'avez constaté, BASIC PLUS n'est pas un magazine mais une véritable encyclopédie conçue avec les ambitions et les moyens des grands ouvrages de références.
- Sa diffusion chez les marchands de journaux vous permet, où que vous soyez, de vous procurer chaque semaine le nouveau fascicule.
- BASIC PLUS paraît tous les mercredis. En achetant vos fascicules de préférence chez le même marchand de journaux, vous serez mieux servis et nous permettrez d'assurer avec plus de précision la distribution.

POUR 12 FASCICULES : UNE ELEGANTE RELIURE MOBILE



Notre ouvrage encyclopédique se devait d'avoir une présentation luxueuse et pratique, afin de pouvoir être consulté souvent par toute la famille et contribuer au prestige de votre bibliothèque. Vous trouverez nos reliures mobiles chez votre marchand de journaux sous la référence : « reliure mobile BASIC PLUS ». Elles vous permettront, dès l'achat des fascicules, de les ranger et de compléter vos volumes. Chaque reliure contient douze fascicules. Nous vous conseillons d'avoir toujours une reliure d'avance afin d'être à même de pouvoir les protéger au fur et à mesure de leur parution.

IL VOUS MANQUE UN OU PLUSIEURS NUMEROS DE BASIC PLUS ?

Une encyclopédie doit être complète et nous savons qu'en période de vacances ou à d'autres occasions, il ne vous sera pas toujours possible de vous procurer le dernier numéro paru. Votre marchand de journaux, habitué à ce problème, se fera un plaisir de recevoir votre commande. Précisez-lui bien le titre : BASIC PLUS, la codification : M 6268, ainsi que le ou les numéros que vous désirez acheter.

Si votre marchand de journaux ne pouvait satisfaire votre demande, écrivez-nous. Précisez le ou les numéros que vous désirez recevoir et accompagnez votre commande d'un chèque bancaire ou postal majoré de 10 %, représentant la participation aux frais de port.

Adressez votre courrier à : BASIC PLUS
France : Service de vente au numéro : 20-22, rue de Clichy – 75009 Paris
Belgique : Avenue Frans Van Kalken, 9 B-1070 Bruxelles

BASIC PLUS : TOUTE UNE EQUIPE A VOTRE SERVICE

Que ce soit pour des problèmes d'achat au numéro, de souscription, ou pour toute autre raison, n'hésitez pas à nous écrire. Notre but est de satisfaire nos lecteurs et de trouver pour chaque problème, et avec vous, la solution.

Dès maintenant, nous vous remercions de votre fidélité.

Les Editions du Hennin

BASIC PLUS

ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS