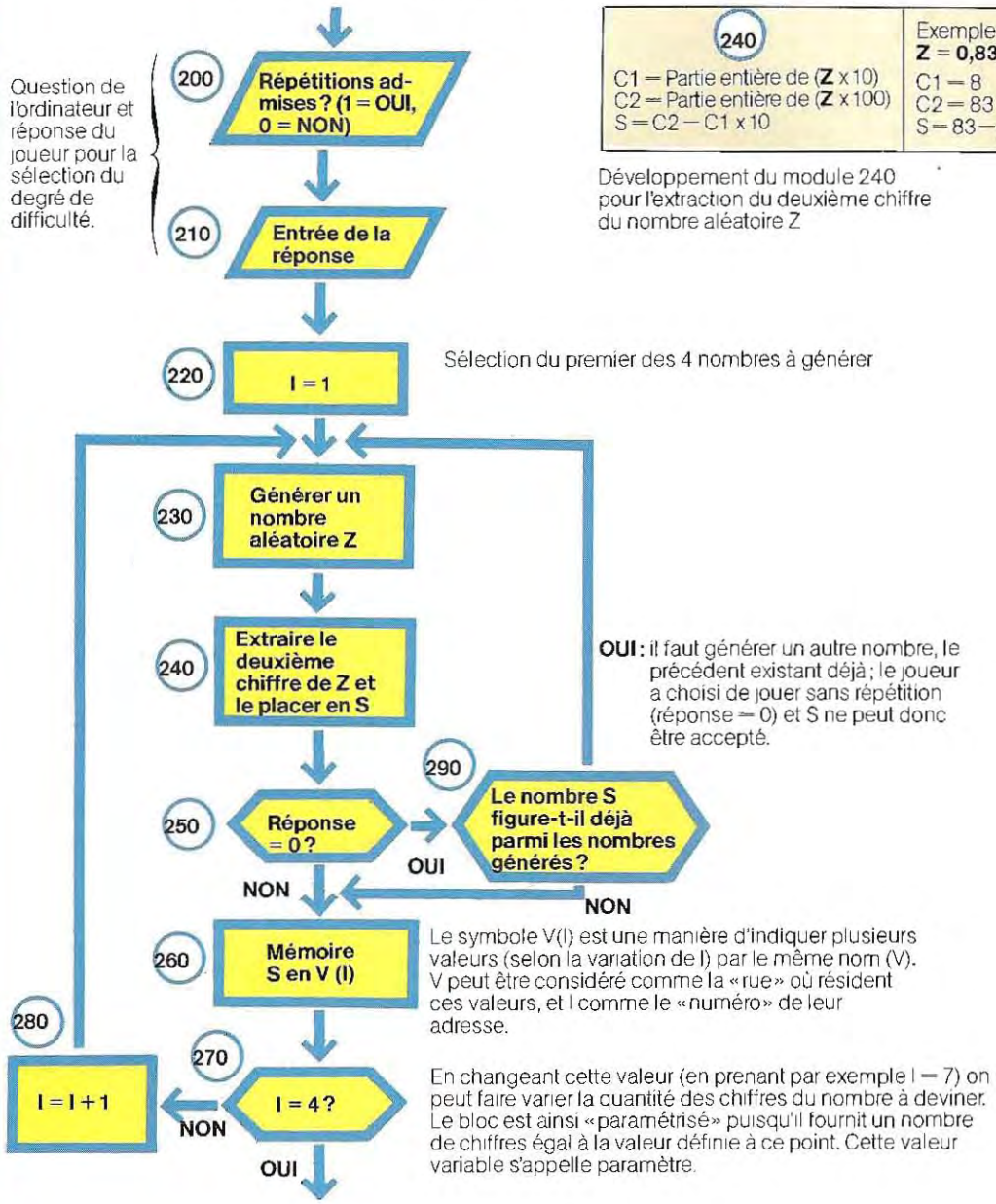
ORGANIGRAMME DE DETAIL DU BLOC 200



240

C1 = Partie entière de (**Z** x 10)
C2 = Partie entière de (**Z** x 100)
S = C2 − C1 x 10

Exemple :
Z = 0,837915
C1 = 8
C2 = 83
S = 83 − 80 = 3

Développement du module 240 pour l'extraction du deuxième chiffre du nombre aléatoire Z

La fonction principale de ce module est de générer des nombres de manière aléatoire, c'est-à-dire des nombres n'ayant aucune relation entre eux. Précisons ici que les ordinateurs sont dotés d'une instruction particulière permettant de générer un nombre aléatoire compris entre 0 et 1. Nous reviendrons ultérieurement sur la nature réelle du nombre ainsi obtenu et sa différence par rapport à un véritable nombre aléatoire.

Dans cette application, le résultat final consiste à obtenir un nombre de quatre chiffres,

c'est-à-dire le nombre à deviner. Ce résultat pourrait être obtenu simplement en isolant les quatre premiers chiffres du nombre aléatoire, mais nous avons choisi la méthode la plus longue qui consiste à générer quatre nombres, puis à isoler dans chacun d'eux l'un des chiffres à deviner.

Si le programme obtenu est plus long il a toutefois l'avantage de permettre de vérifier plus facilement s'il existe dans le nombre à deviner une duplication de chiffres.

Supposons que la machine soit capable de fournir un nombre aléatoire de six chiffres compris entre 0 et 1. Un seul de ces chiffres nous intéressant, il nous faudra multiplier le nombre entier par 10 et en extraire la partie entière.

Prenons par exemple, le nombre aléatoire 0,837915. En le multipliant par 10, on obtient 8,37915, et en en extrayant la partie entière, on obtient 8. De même, si nous voulons extraire le deuxième chiffre (3) au lieu du premier, il nous faut :

1 / extraire le premier chiffre comme dans l'exemple précédent (ce qui donne 8) ;

2 / multiplier le nombre aléatoire par 100 ($0{,}837915 \times 100 = 83{,}7915$) et extraire la partie entière (ce qui nous donne 83) ;

3 / multiplier par 10 le nombre extrait à la première phase ($8 \times 10 = 80$) ;

4 / calculer la différence entre le nombre extrait à la deuxième phase et le nombre calculé à la troisième phase ($83 - 80 = 3$) ; le résultat est le deuxième chiffre du nombre aléatoire initial.

Le chiffre ainsi extrait du nombre aléatoire est l'un des quatre qui composent le nombre à deviner ; en répétant quatre fois la procédure et en stockant les quatre résultats en mémoire, nous obtiendrons le nombre complet. Il faudra à ce moment décider du degré de difficulté que nous voulons donner au jeu : il s'agit dans ce cas d'admettre ou non les répétitions des valeurs entre les quatre chiffres à deviner. La question est posée au début du jeu et la décision incombe au joueur.

Le schéma de la page 199 représente l'organigramme du module 200, développé pour composer le nombre à deviner en extrayant le deuxième chiffre d'une série de nombres aléatoires, avec possibilité d'exclure les répétitions à l'intérieur du nombre à deviner.

Pour compléter ce travail, il suffit à présent de traduire les organigrammes en un langage symbolique. Nous verrons ultérieurement comment établir la liste des instructions de ce programme.

Applications scientifiques

Les organigrammes sont également utilisés, quoique moins souvent, pour la résolution de problèmes scientifiques. Ces applications comportent normalement des programmes de structure simple, c'est-à-dire composés d'un nombre de pas relativement réduit. En revanche les formules à adopter peuvent être compliquées.

La structuration du programme perd de sa minutie et c'est la finesse de l'analyse des procédures mathématiques et des séquences de calcul qui devient déterminante. Ce n'est plus un organigramme qui en résulte mais la méthode mathématique de calcul (algorithme) à adopter pour résoudre le problème donné. Dans les programmes de gestion, au contraire, l'algorithme de calcul est toujours déterminé à l'avance.

La principale difficulté des problèmes scientifiques est justement de déterminer la formule qu'il faut appliquer, et de quelle manière. Le déroulement des calculs peut aboutir à des erreurs selon l'ordre dans lequel ils ont été effectués. En effet, l'ordinateur doit arrondir le résultat de chaque opération car les circuits de calcul ne peuvent traiter des nombres dépassant une certaine longueur : la partie excédentaire est ignorée, d'où la possibilité d'erreurs.

Supposons que notre ordinateur puisse traiter des nombres comprenant un maximum de cinq chiffres et que l'on veuille calculer $(100 \times 0{,}0002) : 100$. Le résultat devrait être 0,0002, puisqu'en multipliant puis en divisant par 100 la même quantité, elle reste invariable.

Développons ce calcul, en effectuant d'abord la multiplication :

$$\mathbf{100 \times 0{,}0002 = 0{,}02.}$$

Le résultat peut encore être traité par la machine, puisqu'il comporte un nombre de chiffres inférieur à cinq.

Effectuons la division par 100 :

$$\mathbf{0{,}02 : 100 = 0{,}0002.}$$

On obtient encore un nombre à cinq chiffres qui peut être traité, et le résultat est exact.

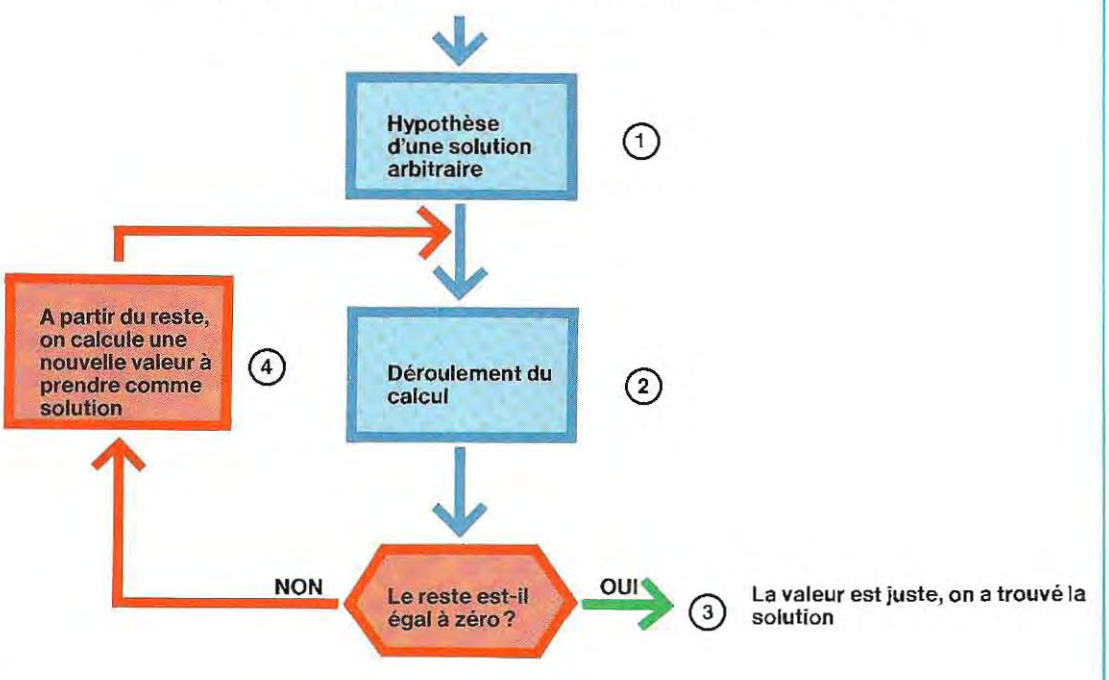
Mais la situation se modifie si nous commençons le calcul par la division :

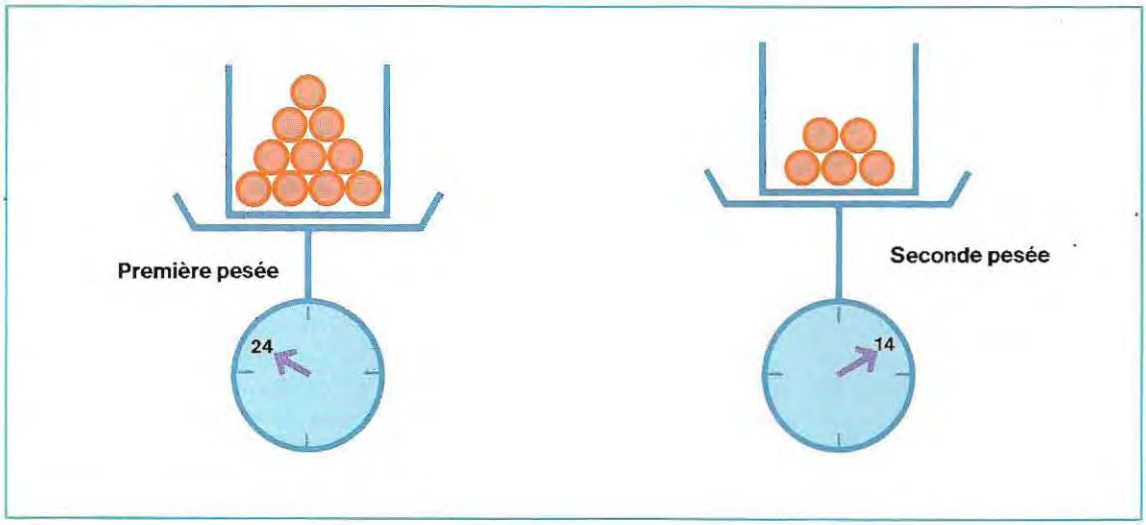
0,0002 : 100 = 0,000002.

Le nombre obtenu dépasse les cinq chiffres, et l'ordinateur ignore automatiquement tous les chiffres au-delà du cinquième.
Le résultat de la division devient 0,0000, et la totalité du calcul sera fausse.
Evidemment les machines ne sont pas aussi limitées ; leur précision courante est très supérieure. Toutefois, il ne faut pas négliger l'éventualité d'erreurs d'arrondi, en particulier, lorsque certains calculs sont répétés un grand nombre de fois : l'accumulation graduelle des erreurs peut conduire à des inexactitudes de résultat, même sur les gros calculateurs, qui sont pourtant des machines extrêmement précises.
C'est la situation qui se présente lors de la résolution des problèmes par la méthode itérative, méthode permettant de trouver la solution d'un problème en donnant initialement à l'inconnue une valeur aléatoire que l'on modifie en fonction du résultat du calcul. Ce principe est illustré ci-dessous.
La première étape consiste à choisir une valeur au hasard (par exemple zéro), la deuxième étape consiste à utiliser cette valeur dans la formule de résolution et à vérifier (troisième étape) si le déroulement du calcul donne bien un zéro pour résultat. Dans ce cas, le calcul est exact et l'hypothèse choisie pour l'inconnue est juste. Dans le cas contraire, il existe un reste (différence entre la valeur prise pour hypothèse et la valeur calculée), qui est utilisé pour formuler une nouvelle valeur d'essai, entrée à son tour dans le module 2 pour une nouvelle vérification. Le processus se poursuit ainsi par approximations successives jusqu'à obtention d'un reste égal à zéro (ou très proche de zéro).
A titre d'exemple, appliquons la méthode itérative à un problème de pesée.
Un récipient, dont on ignore le poids, contient dix boules, dont on ne connaît pas non plus le poids. En plaçant le récipient sur le plateau d'une balance, on constate le poids global (récipient + dix boules) = 24 kg. En retirant cinq boules, on obtient un nouveau poids de 14 kg (récipient + cinq boules). Déterminer quel est le poids du récipient et le poids unitaire des boules.
Appelons **x** le poids du récipient et **y** le poids d'une boule.
La première pesée nous donne :

Récipient + 10 boules = 24 kg
soit : $x + 10y = 24$

SCHEMA DE PRINCIPE DE LA METHODE ITERATIVE





La seconde pesée nous donne :

Récipient + 5 boules = 14 kg
soit : $\mathbf{x} + 5\mathbf{y} = 14$

On trouvera la solution du problème en résolvant le système algébrique suivant, constitué par les deux équations :

$$\mathbf{x} + 10\mathbf{y} = 24$$
$$\mathbf{x} + 5\mathbf{y} = 14$$

Dans ce cas, la résolution du problème relève de méthodes algébriques et l'on obtient (en faisant par exemple la différence entre les deux équations) : $\mathbf{x} = 4$ et $\mathbf{y} = 2$. Vérifions

En cas d'erreur dans un programme d'itération, il arrive que l'on soit littéralement submergé par un océan de papier.

M. Wolf/Black Star-Grazia Neri

cette solution en lui appliquant la méthode itérative.

1 / Attribuons à l'une des inconnues une valeur arbitraire, par exemple $\mathbf{x} = 0$.

2 / Cette valeur est introduite par substitution dans la première équation qui devient :

$$0 + 10\mathbf{y} = 24$$

d'où :

$$10\mathbf{y} = 24$$

$$\mathbf{y} = \frac{24}{10} = 2{,}4$$

3 / La valeur de **y** ainsi calculée ($\mathbf{y} = 2{,}4$) est introduite par substitution dans la seconde équation, d'où l'on peut extraire une nouvelle valeur de **x**. L'équation $\mathbf{x} + 5\mathbf{y} = 14$, avec $\mathbf{y} = 2{,}4$ devient :

$$\mathbf{x} + (5 \times 2{,}4) = 14$$

soit :

$$\mathbf{x} + 12 = 14$$
$$\mathbf{x} = 14 - 12 = 2$$

4 / Cette nouvelle valeur de **x** est introduite dans la première équation de la même manière qu'à l'étape 2 :

$$2 + 10\mathbf{y} = 24$$
$$10\mathbf{y} = 24 - 2$$
$$10\mathbf{y} = 22$$
$$\mathbf{y} = \frac{22}{10} = 2{,}2$$

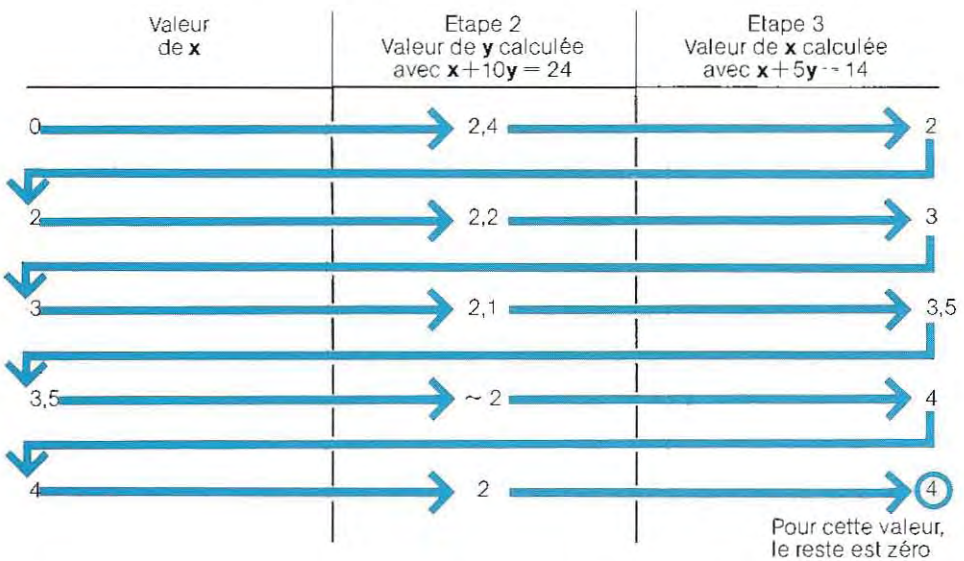
Une série de passages correspondant aux points 2, 3 et 4 fournissent des valeurs qui figurent sur le schéma ci-dessous.

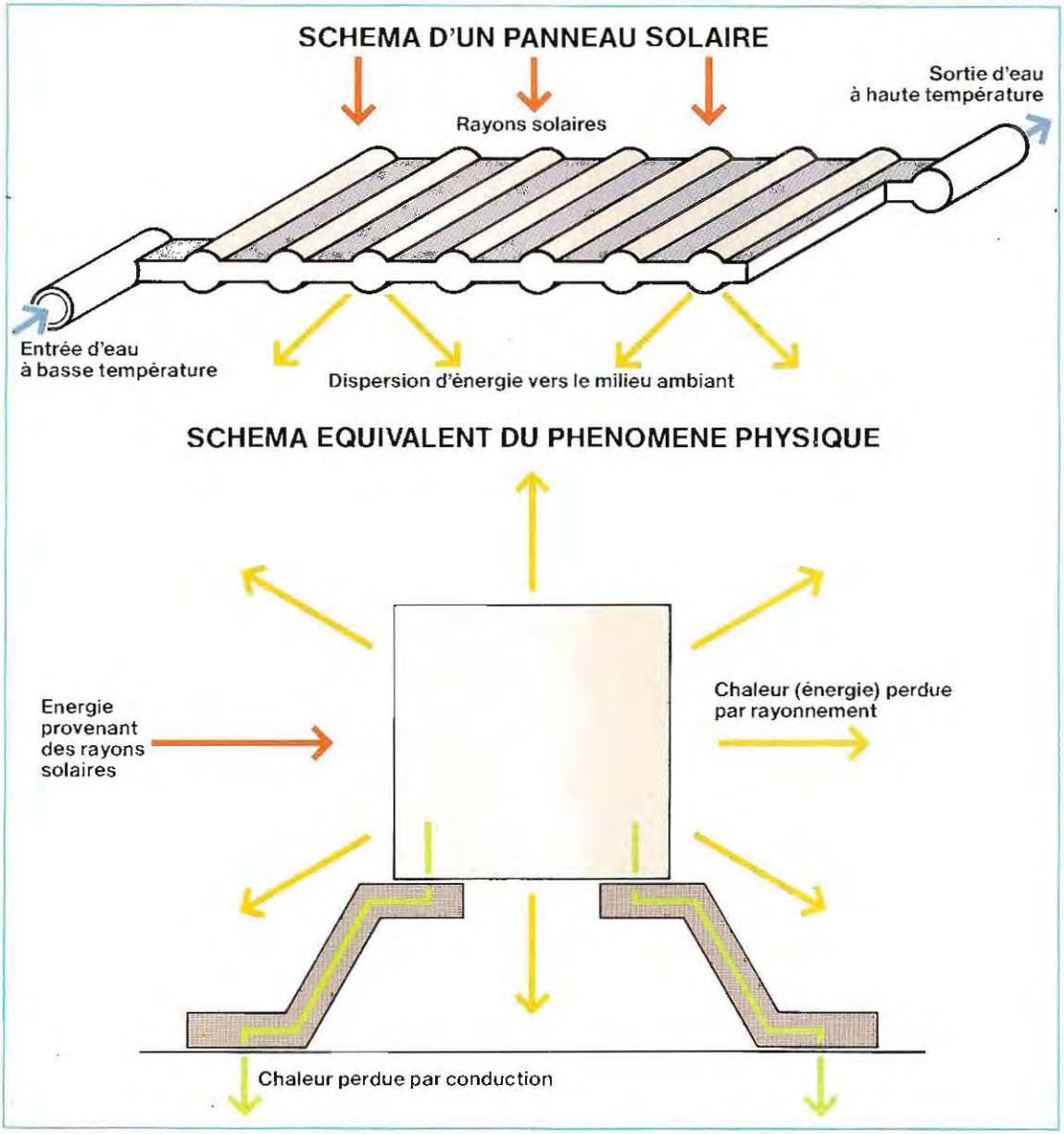
Les deux équations $\mathbf{x} + 10\mathbf{y} = 24$ et $\mathbf{x} + 5\mathbf{y} = 14$ donnent toutes deux zéro si l'on remplace **x** par 4 et **y** par 2. Ces valeurs représentent donc la solution cherchée. Si l'on intervertit les termes du problème, en faisant la première pesée avec cinq boules et la seconde avec dix, la méthode, au lieu de donner, par convergence, des solutions de plus en plus proches, donnera des valeurs négatives divergentes, s'écartant de plus en plus de la solution. Lorsqu'on veut adopter cette dernière méthode, il faut analyser avec beaucoup d'attention la mise en place du calcul, afin d'éviter d'aboutir à la divergence des solutions d'essai.

En fait, le système illustré ici possédant une solution algébrique, rien ne justifie l'utilisation de la méthode itérative pour en chercher la solution. Dans les applications scientifiques, en revanche, on rencontre souvent des problèmes impossibles à résoudre autrement que de cette manière. Le sujet étant vaste, de nombreuses méthodes théoriques, parfois complexes, ont été mises au point pour assurer la convergence. Le paragraphe suivant illustre l'application de la méthode itérative à la résolution de problèmes thermiques.

Applications de la méthode itérative

L'exemple précédent (solution d'un système de deux équations à deux inconnues) n'a été choisi qu'à titre explicatif. En réalité, quelle





que soit leur complexité, les systèmes du premier degré (dans lesquels les inconnues ne sont pas élevées à une puissance), comportant un nombre d'équations égal au nombre d'inconnues, peuvent être résolus par des méthodes algébriques plus rapides et plus sûres que la méthode itérative.

Cette méthode itérative sera réservée aux équations qui ne possèdent pas de solution algébrique. Voici une équation caractéristique de cette espèce :

$$A \times T^4 + B \times T = C$$

L'inconnue est T, tandis que A, B et C sont des valeurs numériques constantes. Avant d'examiner une méthode de résolution de ces équations, voyons à quel domaine elle peut être appliquée.

La civilisation industrielle doit faire face au problème fondamental des sources d'énergie. Les ressources traditionnelles, et surtout le pétrole, semblent en voie d'épuisement, leur coût ne cesse d'augmenter, et elles provoquent également des pollutions. Il est donc urgent d'envisager l'exploitation d'autres sources d'énergie. L'une d'entre elles, renouvelable par excellence, et non polluante,

est l'énergie solaire. Nous savons qu'un récipient plein d'eau exposé aux rayons solaires se réchauffe par accumulation de l'énergie du soleil. Une fois l'eau réchauffée, nous pouvons exploiter cette énergie à des fins différentes, et, en particulier, pour les besoins domestiques. C'est déjà une réalité dans plusieurs pays, où sont commercialisés des panneaux solaires destinés à capter l'énergie solaire nécessaire au chauffage d'une maison (chauffage central, eau chaude). Pour que l'installation soit réellement efficace, il faut qu'elle soit bien étudiée et c'est à cette étude qu'on applique la formule exposée. Il est possible de schématiser le phénomène physique pour une première approximation, comme l'indique la figure de la page 204. En haut, on voit le schéma fonctionnel d'un panneau solaire ; en bas, une représentation schématique qui illustre un phénomène physique réel. Supposons que l'eau ne circule pas. Si nous voulons maintenir à une température donnée (T) l'eau contenue (immobile) dans le récipient, il faudra fournir autant d'énergie que le récipient en perd. Si nous en fournissons moins, l'eau refroidira, si nous en fournissons plus, elle s'échauffera plus que nécessaire pour obtenir la température requise. Dans les conditions « stationnaires » d'une eau immobile à température constante, le bilan énergétique est :

énergie fournie = énergie perdue.

L'énergie fournie est la quantité de chaleur apportée par les rayons solaires et dont la valeur est représentée par C. L'énergie perdue est celle que le récipient cède au milieu environnant (déperdition).

Les principales déperditions thermiques se font par conduction et par rayonnement. Les pertes par conduction sont dues à la quantité

Les panneaux solaires exploitent la chaleur du rayonnement solaire pour réchauffer de l'eau. Ci-dessous : une installation de chauffage solaire.

Archives Curcio/CTIP Solar

de chaleur qui migre d'un corps plus chaud vers un corps plus froid mis au contact de celui-ci (par exemple, entre le récipient et les poutrelles qui le soutiennent), et elles sont proportionnelles à la différence de température entre les deux corps.

Déperdition par conduction =
= B × (température du premier corps −
− température du second corps).

Pour plus de simplicité (ce que nous exposons ici n'a qu'un caractère indicatif), nous négligerons la température du second corps par rapport à celle du premier :

Déperdition par conduction = B × T

B étant un facteur lié au type de matériau et à la forme géométrique du système.

Le second type de perte, par rayonnement, est dû à la migration de chaleur d'un corps vers le milieu environnant (plus froid) sous la forme d'ondes électromagnétiques (radiations). Dans ce type d'échange de chaleur, l'énergie transmise est proportionnelle aux différences entre les puissances quatre des températures.

Déperdition par rayonnement =
= A × (température du premier corps)^4 −
− (température du deuxième corps)^4.

Dans ce cas aussi, en négligeant la température du second corps, nous obtenons :

Déperdition par rayonnement = A × T^4

Le facteur A dépend des caractéristiques physiques et géométriques du corps.

L'énergie totale perdue est la somme de ces deux pertes (conduction + rayonnement), soit :

Energie totale perdue = A × T^4 + B × T

Pour maintenir constante la température (T), il faut fournir au récipient une quantité égale d'énergie ; en d'autres termes, l'énergie apportée par les rayons solaires (C) doit être égale à l'énergie totale perdue :

C = A × T^4 + B × T

L'équation obtenue est de forme simplifiée, car elle ne tient pas compte des déperditions par échange de chaleur avec l'air qui entoure le récipient (échange de type convectif). En outre, dans bien des cas, si la valeur de T n'est pas grande, le terme A × T^4 peut être négligé. La formule que l'on en tire, connaissant les valeurs de A, B, C (constantes et dépendant des caractéristiques du récipient), fournit la température (T) d'équilibre à laquelle l'eau se maintient sous l'effet combiné de l'énergie d'entrée et de l'énergie de sortie. Dans la formule, la température (T) n'est pas exprimée en degrés Celsius, mais en degrés Kelvin (le 0° Kelvin se situe à −273,2° Celsius ; une température en degrés Kelvin se calcule donc en ajoutant 273,2 au nombre de degrés Celsius. Comme on le voit, l'utilisation des degrés Kelvin donne des nombres très élevés et, pour faciliter les calculs, on divise habituellement la valeur par 1000.

TEST 5

1 / Le symbole  représente :
a : une décision ; b : le développement d'un calcul ; c : la fin du programme.

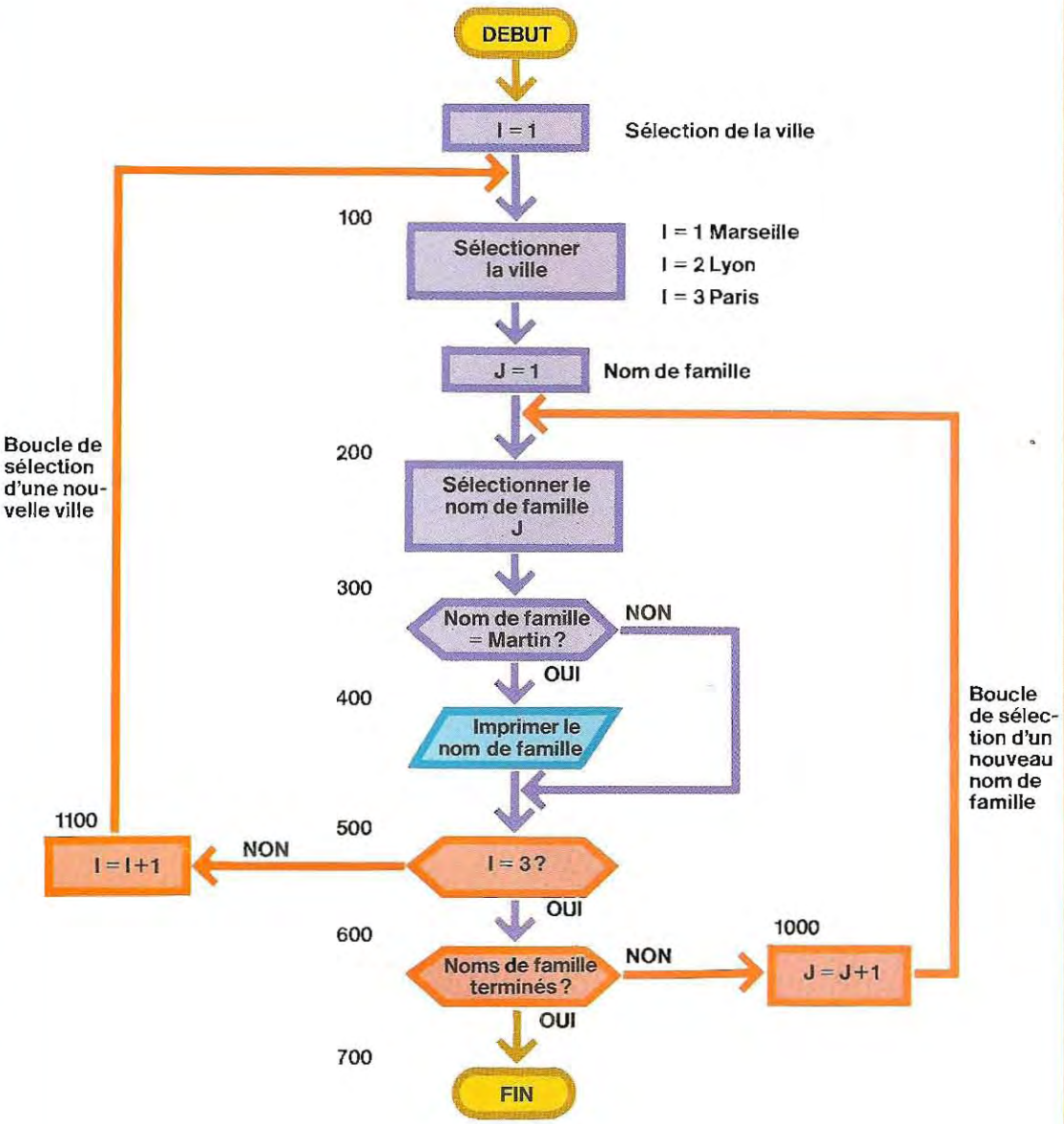
2 / Dessiner les symboles représentant les actions suivantes :
a : traitement général ; b : calcul ; c : impression.

3 / Quelle est l'utilisation du symbole : 

4 / Qu'est-ce qu'un drapeau (flag) :
a : un indicateur ; b : un compteur ; c : un registre de l'unité centrale.

5 / Dessiner l'organigramme d'un programme pour le calcul des pourcentages avec vérification à l'entrée que le pourcentage ne soit pas supérieur à 100.

6 / L'organigramme ci-dessous contient une erreur. Déterminer laquelle, et tracer la forme exacte du schéma. Le problème est le suivant : nous disposons de trois groupes d'adresses, le premier concerne la ville de Marseille, le deuxième la ville de Lyon et le troisième la ville de Paris. Chacun des groupes contient dix adresses. Nous voulons sélectionner toutes les adresses des personnes dénommées Martin (il faut deux boucles, une pour sélectionner la ville, et l'autre pour le nom de famille).



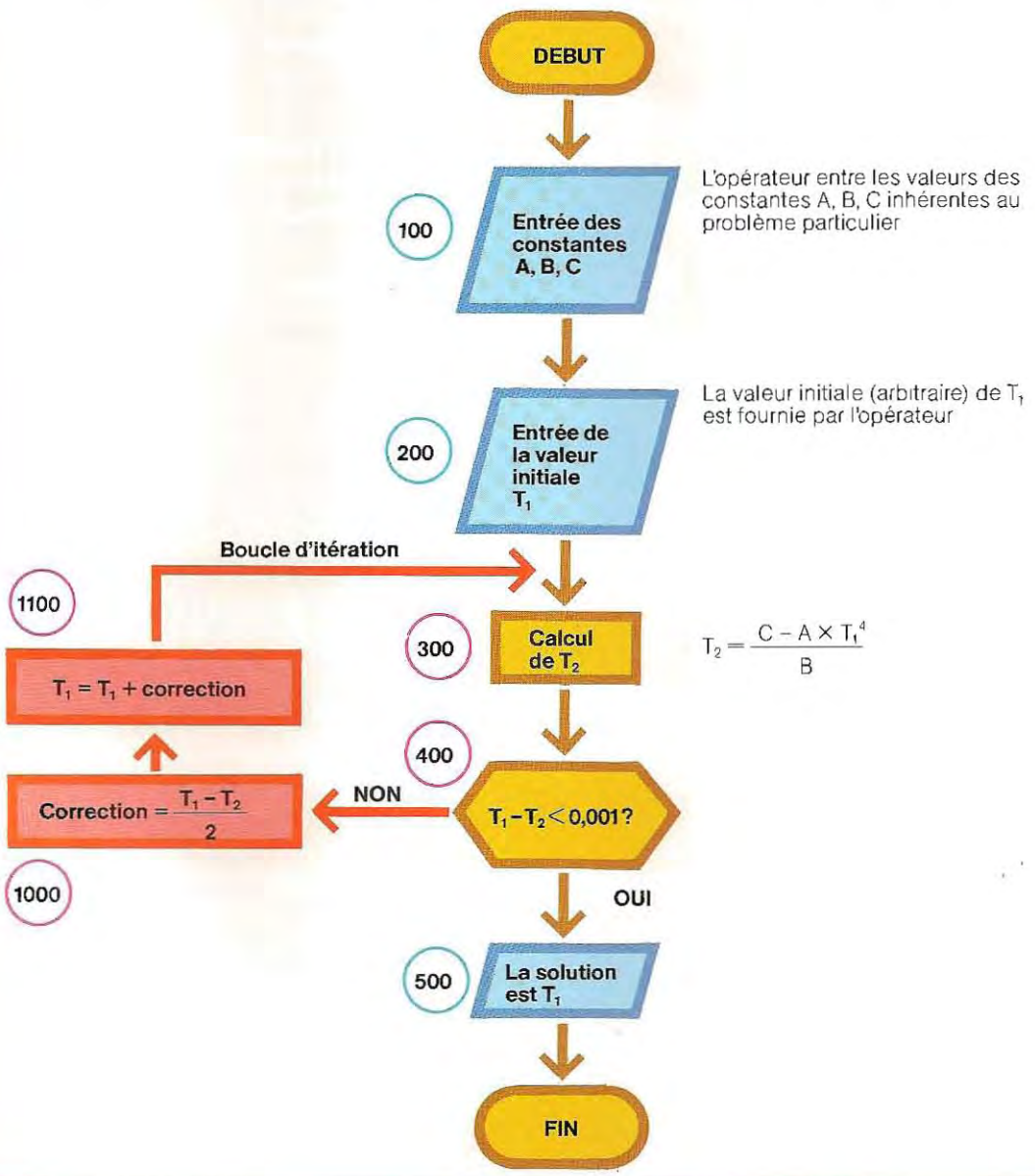
Les solutions du test se trouvent pages 214 et 215.

Tous les calculs se font alors sur de petits nombres. Par exemple, une température de 373,2° K élevée à la puissance quatre (T^4) donne 19 398 428 000, alors qu'en divisant la température par 1000 (c'est-à-dire 0,3732) on obtiendra 0,0193984, que l'on peut arrondir à 0,0194. A la fin du calcul, il faudra évidemment multiplier le résultat par 1000. Cette méthode est également pratique pour les valeurs particulières des constantes utilisées dans les calculs thermiques.

Inscrivons à nouveau la formule du bilan thermique:

C chaleur fournie par le soleil	=	$A \times T^4$ déperdition par rayonnement	+	$B \times T$ déperdition par rayonnement

ORGANIGRAMME POUR LA RESOLUTION DES EQUATIONS DE TYPE $C = A \times T^4 + B \times T$ PAR LA METHODE ITERATIVE



L'équation peut être réécrite de la manière suivante :

$$T = \frac{C - A \times T^4}{B}$$

La solution T (température à laquelle monte le récipient) s'obtient par la procédure itérative suivante :

1 / On introduit dans le terme $A \times T^4$ une valeur arbitraire de T (par exemple T_1) pour effectuer les calculs indiqués dans le second membre de l'équation.
Le calcul terminé, le premier membre de l'équation fournit une nouvelle valeur de T que nous appellerons T_2 :

$$T_2 = \frac{C - A \times T_1^4}{B}$$

2 / Si la différence entre les valeurs T_1 et T_2 est faible (par exemple 0,001) c'est que l'on a trouvé la solution.

3 / Dans le cas contraire, on calcule pour T une valeur de correction égale à la moyenne des deux valeurs T_1 et T_2 :

$$\text{correction} = \frac{T_1 - T_2}{2}$$

4 / Cette correction est appliquée à la valeur initiale (arbitraire) T_1 qui devient T_1 + correction.

5 / On repart de l'étape 1 avec cette nouvelle valeur de T.

L'organigramme de la page 208 illustre cette méthode.

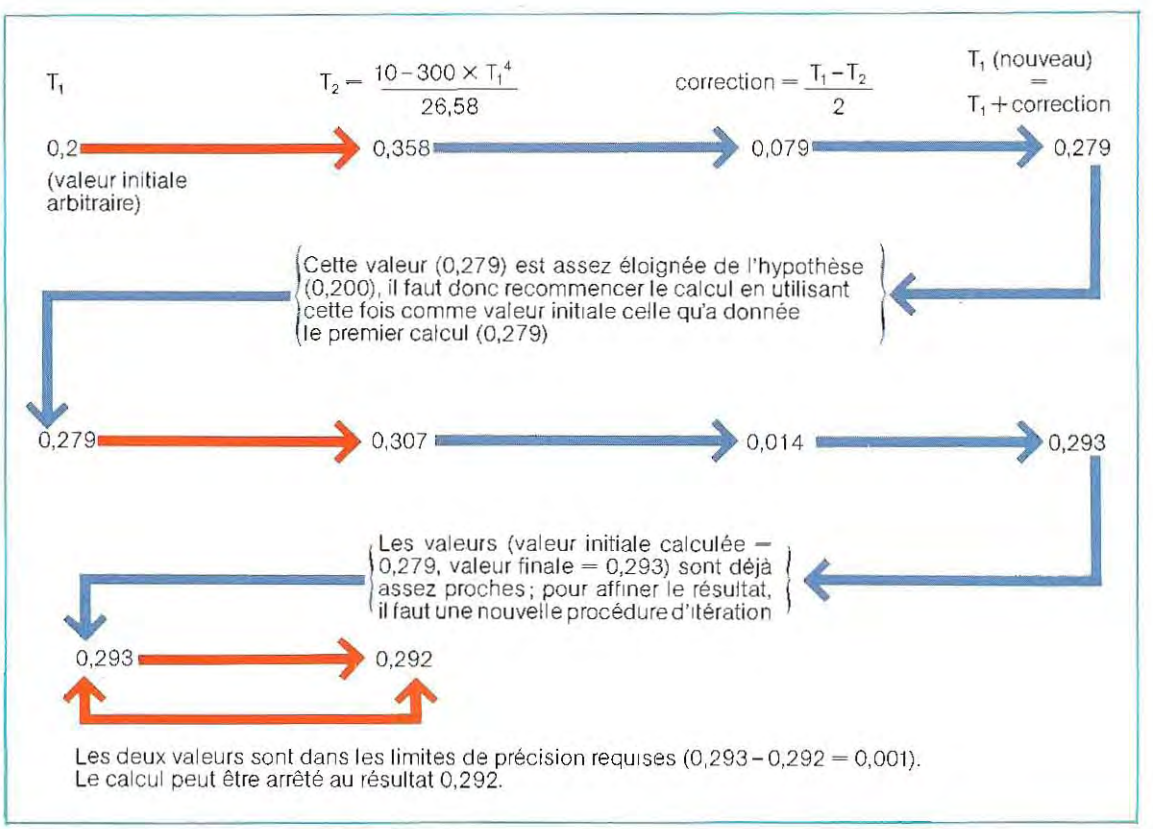
Le tableau ci-dessous illustre le développement d'un calcul avec adoption des valeurs A = 300 ; B = 26,58 ; C = 10 ; ce qui donne pour l'équation :

$$300 \times T^4 + 26{,}58 \times T = 10$$

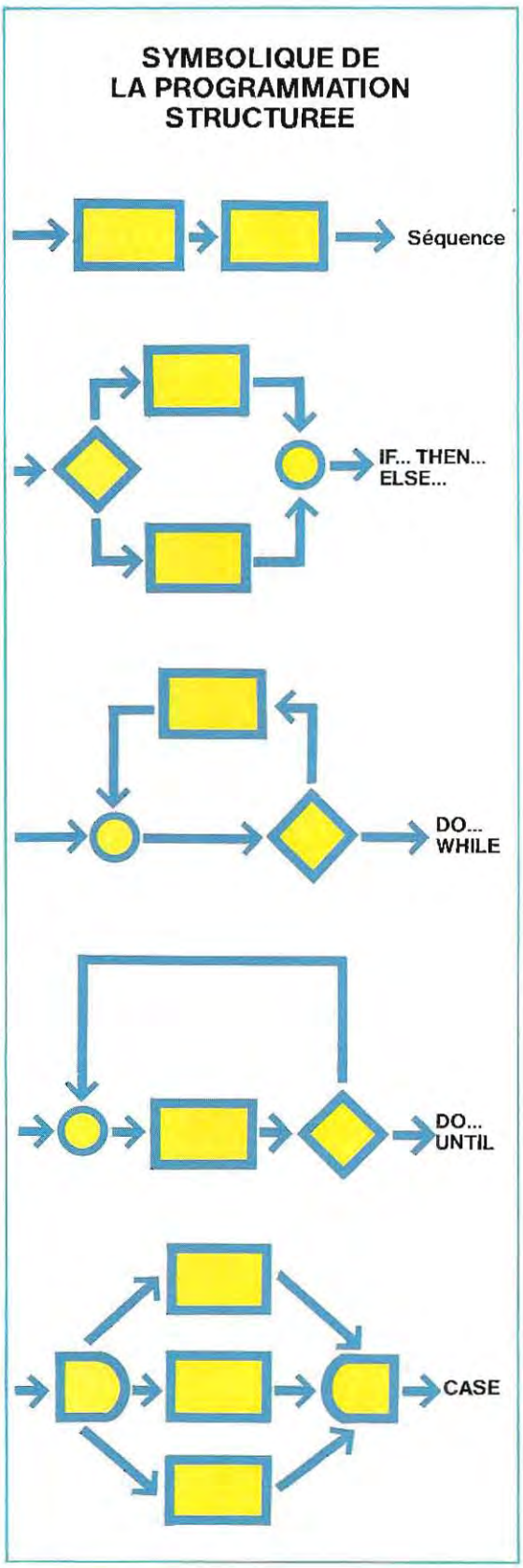
Programmation structurée

Les méthodes appliquées à l'analyse d'un problème, à sa décomposition en un certain nombre de problèmes plus simples, et finalement à l'établissement des organigrammes correspondants, permettent de définir la structure logique qui est valable dans la majorité des cas. Les problèmes plus complexes, toutefois, ne seront pas résolus de cette manière.

On appliquera avec profit la méthode HIPO et la préparation des organigrammes à la pro-



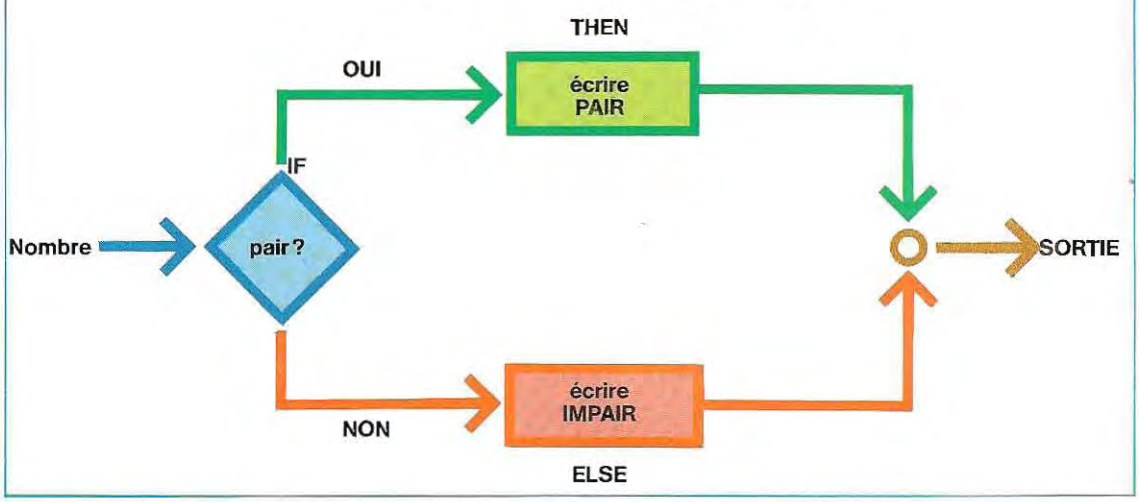
grammation des micro-ordinateurs et des mini-ordinateurs.
Pour les systèmes plus complexes, on aura recours à des techniques plus rigoureuses qui, bien que laissant moins de place à l'interprétation individuelle, sont régies par des règles suffisamment précises pour permettre un travail d'équipe.
La préparation d'un programme pour un micro ou un mini-ordinateur est généralement l'affaire d'une seule personne; le travail est alors assez souple. Mais dès que l'ordinateur devient plus performant et que la complexité des problèmes augmente, le nombre de personnes qui participent à la tâche augmente aussi, en même temps que la nécessité d'une documentation claire et de méthodes normalisées.
Il faut en outre adopter des techniques réduisant les risques d'erreur tout en permettant de gagner du temps.
La plus moderne et la plus sophistiquée de ces méthodes est la **programmation structurée**.
Il s'agit d'une méthode complexe, tout à fait précieuse pour les applications destinées aux gros ordinateurs ou aux mini-ordinateurs de haut de gamme.
Nous aurons l'occasion de revenir sur cette technique de programmation, fondée sur cinq fonctions principales, mais nous nous limiterons ici à en décrire certains des aspects principaux.
Puisque chacun des symboles d'un organigramme signale par lui-même une action élémentaire, un organigramme complet en contient généralement un nombre assez grand.
Au contraire, la programmation structurée ne comporte pas de symboles élémentaires, mais des symboles composés, qui expriment des actions complexes. Par exemple, un organigramme décrivant une boucle à rupture de code le fait à l'aide d'un seul symbole complexe.
Naturellement, cette approche n'a de sens que si l'on dispose, lors de la phase d'écriture du programme, d'un langage approprié. Il serait inutile de préparer un organigramme synthétique s'il fallait ensuite le développer en détails pour le traduire en instructions.
Les langages les plus courants (Basic, Fortran, Cobol) pour les micro-ordinateurs sont



en général peu adaptés à la programmation structurée. Mais il existe d'autres langages, tels le **PL/1**, le **PL/C**, le **PASCAL**, l'**ALGOL**. Relativement récents, ils permettent une notable économie de temps. Il n'est d'ailleurs pas impossible que leur emploi se généralise bientôt (PL/1 et PASCAL sont déjà utilisés pour certains micro-ordinateurs).
Les principales fonctions de la programmation structurée sont illustrées par le schéma de la page 210 et décrites ci-dessous.

Séquence
Le symbole graphique exprime clairement la fonction : une série d'opérations (de quelque type que ce soit) à exécuter à la suite l'une de l'autre.

ORGANIGRAMME D'UNE DECISION REECRITE AVEC LES SYMBOLES DE LA PROGRAMMATION STRUCTUREE



IF... THEN... ELSE...
C'est un symbole composé. Il commence par une décision (test de condition) ; le résultat débouche sur deux voies possibles, l'une à adopter si la condition est vraie, l'autre si la condition est fausse. La sortie est commune.
Le schéma ci-dessus en illustre une application. A l'entrée, le bloc reçoit un nombre et doit inscrire le message « pair », si le nombre est pair ; dans le cas contraire, le message doit être « impair ».
En interprétant à la lettre les phrases que contient ce bloc, on obtient :

IF « le nombre est pair » ; THEN « écrire pair » ;
ELSE « écrire impair ».

Il s'agit donc d'une forme très comparable au symbole de décision habituel.

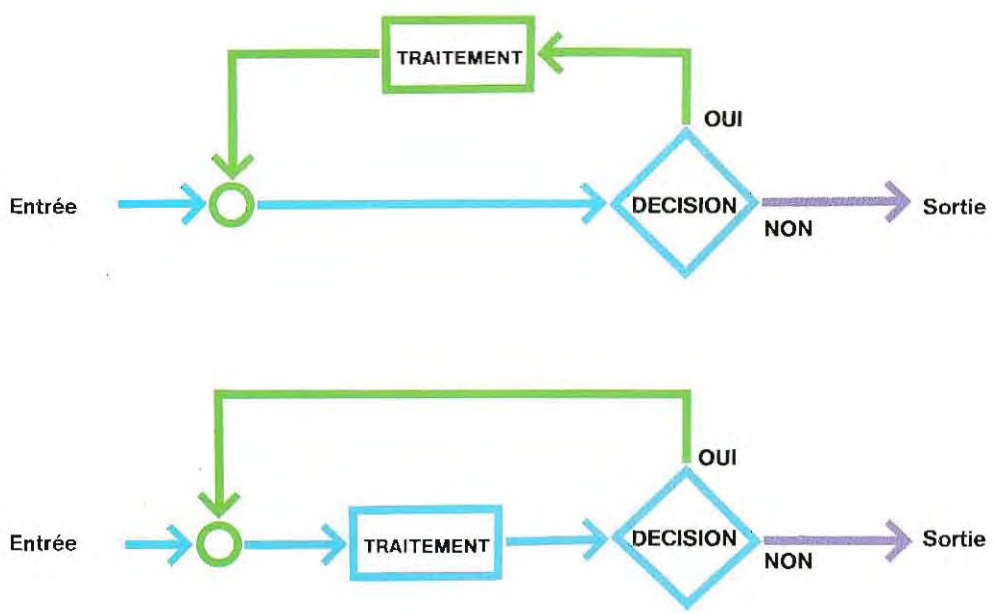
DO... WHILE et DO... UNTIL

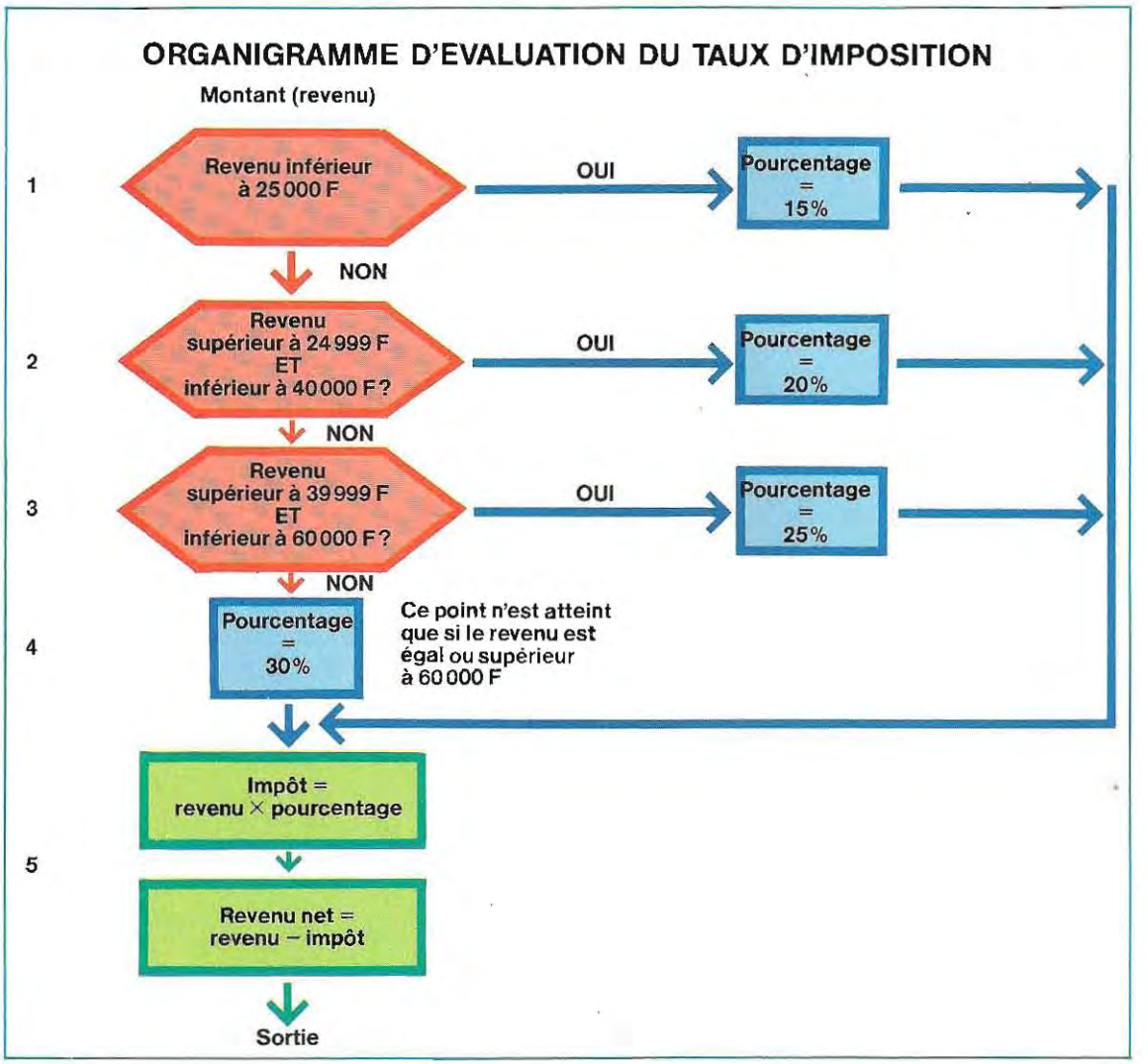
Ces deux blocs sont semblables et contiennent un test (symbole de décision) et un traitement. Dans le premier, la décision est exécutée après l'entrée ; si la réponse est positive, on exécute le traitement et on rentre dans la décision. La sortie est obtenue si la décision donne un résultat négatif. Dans le second bloc, le traitement précède la décision. Il s'agit de deux formes compactes de boucle.

CASE

Ce bloc n'a pas de lien immédiat avec la logique symbolique des organigrammes. Il signifie une comparaison multiple ; pour obtenir le même résultat avec des symboles de décision, il faudrait en utiliser autant que de voies possibles. Voici un exemple d'application. Les impôts sur le revenu dépendent du montant de ce revenu. Supposons qu'il n'y ait que quatre échelons :

BLOCS DO... WHILE / DO... UNTIL

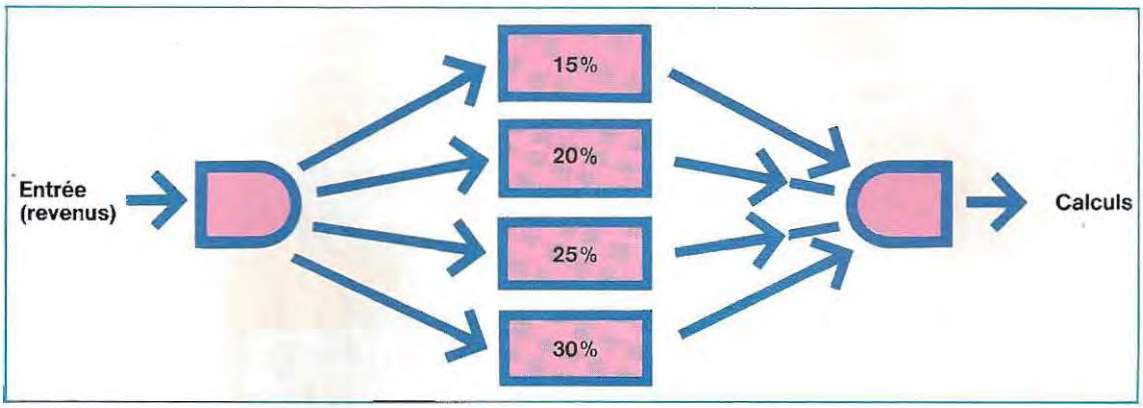




Revenu :

inférieur à 25 000 F	impôt = 15 %
de 25 000 à 39 999 F	impôt = 20 %
de 40 000 à 59 999 F	impôt = 25 %
égal ou supérieur à 60 000 F	impôt = 30 %

Le problème consiste à écrire un programme capable de déterminer le taux à appliquer et de calculer le montant de l'impôt et du revenu net.
Le schéma ci-dessus représente l'organigramme du problème avec les symboles normaux.
La décision 1 expose que le revenu n'est pas supérieur à 24 999 (inférieur à 25 000) ; dans ce cas le pourcentage est de 15 %. Ayant défini le pourcentage, le programme saute directement au calcul (étape 5).
Dans le cas où la réponse du bloc 1 est négative, on passe à la décision 2. Si le revenu est situé dans la deuxième fourchette (de 25 000 à 39 999 F), le programme sélectionne le taux de 20 % et passe au calcul.
On notera l'utilisation de l'opérateur ET pour exprimer deux conditions qui doivent se vérifier simultanément (revenu supérieur à 24 999 et inférieur à 40 000 F, c'est-à-dire de 25 000 à 39 999 F. L'étape 4 ne nécessite aucune vérification, car si l'étape 3 indique que le revenu n'est pas inférieur à 60 000 F, il faut automatiquement appliquer le pourcentage maximum.
La forme CASE permet d'exprimer cette logique d'une manière plus synthétique puisque le CASE peut être considéré comme les trois symboles de décision réunis. On en trouvera le schéma de principe à la page 214.
La programmation en Basic ou en Fortran ne permet pas de disposer d'instructions autorisant l'utilisation de logiques aussi compactes.



Solutions du test 5

1 / a : le symbole représente une décision. On indique le développement d'un calcul par un symbole de traitement général (rectangle) à l'intérieur duquel on inscrit le calcul à effectuer. La fin du programme est un ovale portant la mention FIN ou END.

---

2 / a et b:  le symbole du traitement général est utilisé aussi pour les calculs.

c :  ce symbole désigne l'impression d'un document.

---

3 / Ce symbole (renvoi) sert à indiquer les points de l'organigramme auxquels se réfèrent des parties éloignées (du schéma) et qu'il ne serait pas facile de relier entre elles par une ligne continue.

---

4 / a : le drapeau (flag) est un indicateur dont l'état signale si un événement déterminé a eu lieu ou non.

---

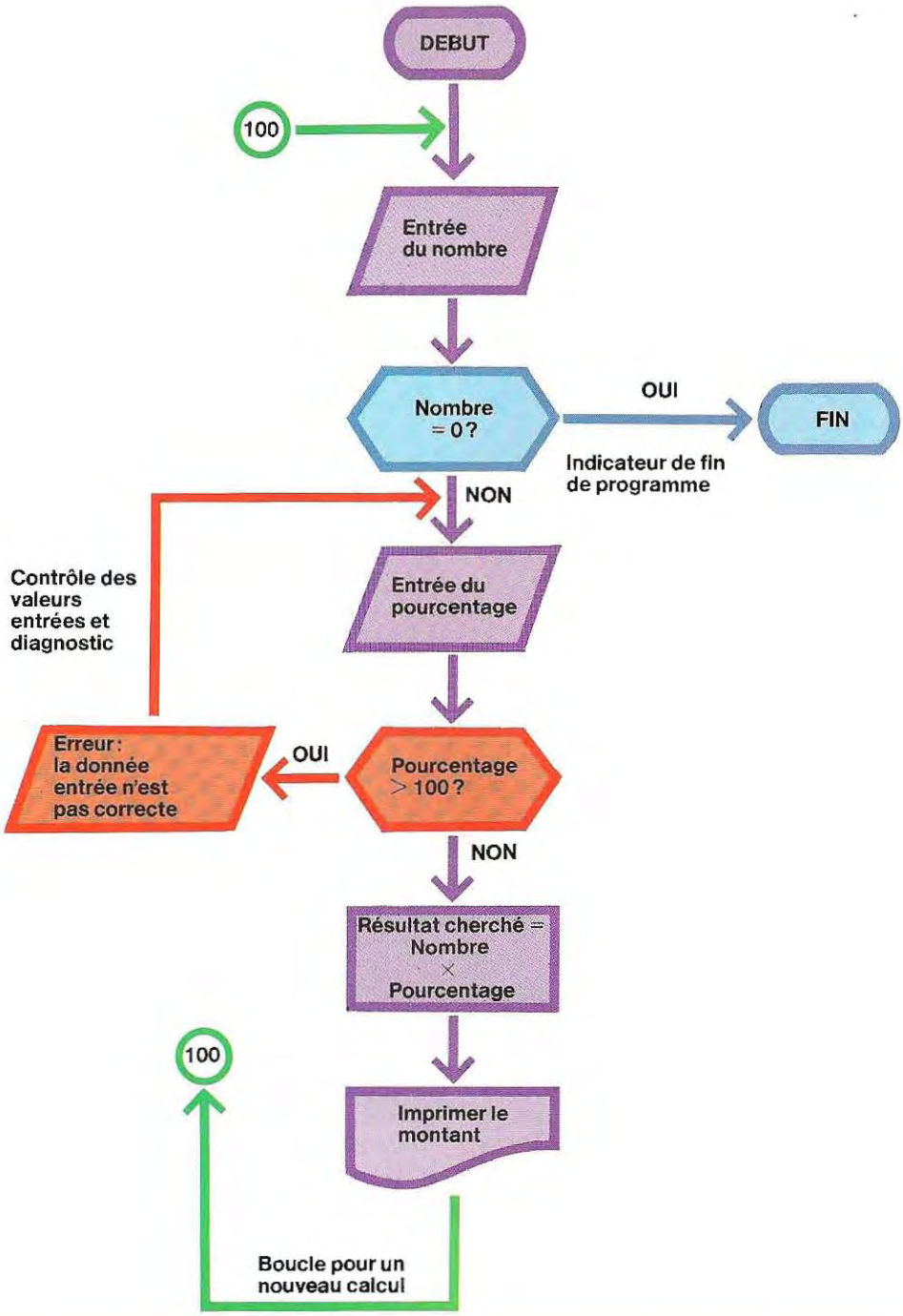
5 / L'analyse du problème peut être synthétisée comme suit :
- Entrée d'un nombre donné dont il faut calculer le pourcentage.
- Vérification que ce nombre ne soit pas égal à zéro (la valeur zéro peut être adoptée comme aiguillage, ce qui aboutirait à interrompre le programme).
- Entrée de la valeur du pourcentage et de sa vérification.
- Calcul : résultat cherché = nombre × pourcentage.
- Impression.

Cet organigramme est illustré page 215.

---

6 / L'erreur est double. Elle concerne d'abord la présence de deux boucles (I et J) qui se recoupent. La boucle I (qui commence au bloc 100) doit être fermée après la boucle J (qui commence à 200), pour éviter toute intersection. Pour bien déceler l'erreur, il suffit de tracer la boucle passant par le bloc 1 100 (à gauche sur le schéma p. 207) du même côté que la boucle 1 000 (à droite) : on voit clairement que les deux lignes de retour se coupent, ce qui n'est pas autorisé. La correction se fait en

échangeant la position des blocs 500 et 600 de fermeture des boucles.
L'erreur réside ensuite dans la forme du symbole utilisé pour désigner l'impression du nom : on aurait dû tracer le symbole I, spécifique de l'impression (voir p. 168).



Des machines qui parlent

Les considérables progrès techniques appliqués à l'informatique permettent, depuis quelques années déjà, des essais d'échanges verbaux entre l'homme et la machine.

Le problème est double : il faut que la machine soit capable d'entendre la voix humaine et de la comprendre, mais aussi qu'elle puisse parler de façon cohérente et formuler des phrases constituant une réponse à la question posée.

Les dispositifs électroniques dotés de parole sont fondés sur deux principes fondamentaux.

Le premier, la « synthèse verbale concaténée », utilise un ordinateur pour emmagasiner une sorte de dictionnaire des syllabes d'une langue donnée, présentées dans toutes les combinaisons possibles (voyelles-consonnes et consonnes-voyelles). Pour former un mot, l'ordinateur extrait de sa mémoire les syllabes requises et les relie ensemble dans le bon ordre (par exemple pour dire « or-di-na-teur »).

La méthode du parler concaténé est très rigide et produit un discours de mauvaise qualité. Elle a toutefois l'avantage d'être rapide et relativement économique.

L'autre méthode, dite de la « synthèse par règles », n'emmagasine pas dans la mémoire des sons ou des combinaisons de sons, mais plutôt un guide électronique des règles tirées d'une analyse détaillée du langage humain, et de la façon de générer ces sons. Le parler obtenu est de meilleure qualité ; toutefois, comme il faut mettre au point de très nombreuses règles, il faut aussi beaucoup de temps à la machine pour synthétiser le langage.

Un certain nombre de machines parlantes appliquant la « synthèse verbale concaténée » sont déjà utilisées, surtout aux Etats-Unis. Au lieu de les afficher, les calculatrices parlantes peuvent prononcer les chiffres correspondant aux touches du clavier que l'on enfonce, et donc lire le résultat à haute voix. Il existe également de petits dispositifs parlants, un peu plus grands qu'une calculatrice, et qui servent à apprendre à parler aux enfants. Une autre petite machine qui sait jouer aux échecs illustre verbalement chacun de ses mouvements.

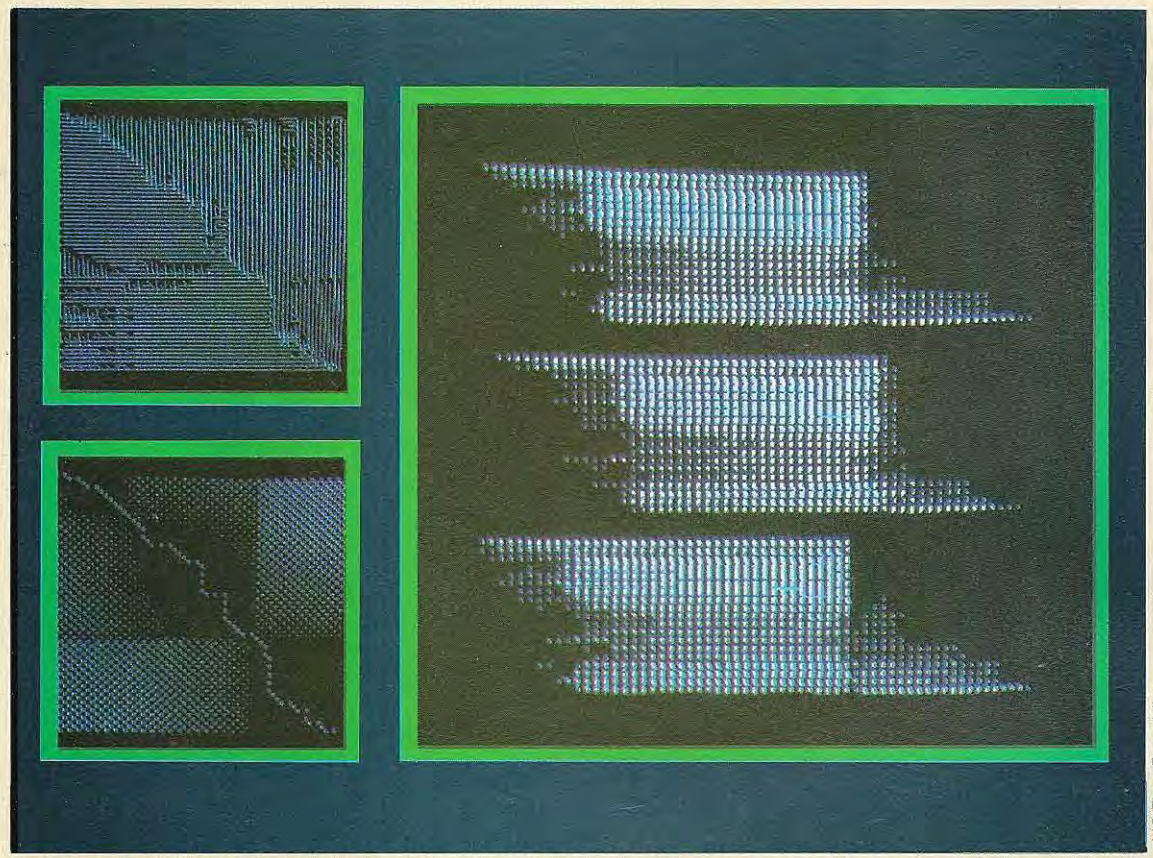
Une machine dotée de la parole est l'élément essentiel d'un système utilisé aux Etats-Unis par la Bell Telephone Company pour améliorer l'efficacité et réduire le coût de certains services normalement assurés par des opérateurs humains. Un petit clavier comparable à celui d'une machine à écrire est mis en place dans l'appareil téléphonique. Pour trouver le numéro d'un abonné, on se met en liaison avec le service des renseignements téléphoniques et l'on tape sur le clavier le nom et l'adresse de l'utilisateur recherché. L'ordinateur, à l'autre bout de la ligne, cherche dans sa mémoire, trouve le numéro requis et, avec la voix de la machine, répond par téléphone.

Le stade ultérieur prévoyant l'élimination du clavier et la dictée des questions à la machine, représente une difficulté supérieure.

Le spectrographe acoustique, inventé au début de la Seconde Guerre mondiale, est l'instrument fondamental des chercheurs qui tentent de mettre au point le système permettant à la machine de reconnaître le langage articulé. Il s'agit d'un dispositif électronique qui analyse les ondes acoustiques et en fournit le tableau graphique permanent. L'intérêt de ces tableaux est qu'ils peuvent être stockés (sous forme numérique) dans la mémoire d'un ordinateur.

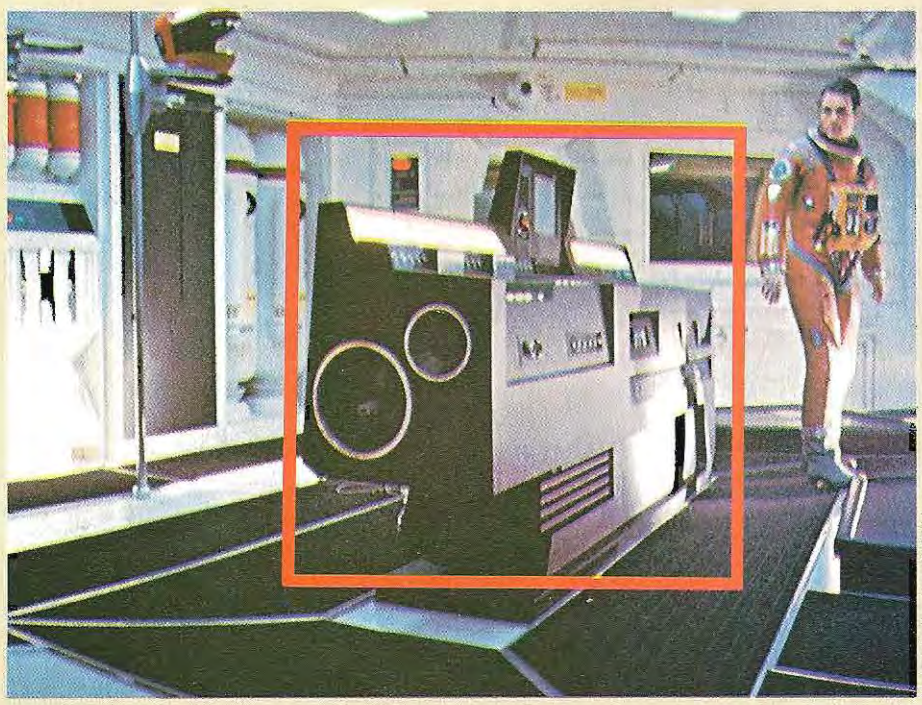
En théorie, il devrait être facile à ce dernier de reconnaître une parole donnée en comparant le spectrogramme de chacun des sons reçus avec les informations spectrographiques stockées dans sa mémoire. Le problème est qu'il n'existe pas deux personnes (même dotées d'un accent régional similaire) qui prononcent un même mot de manière exactement identique. Des paroles identiques prononcées par des sujets différents produisent des tableaux spectrographiques distincts, ce qui crée bien évidemment la difficulté, mais peut être fort utile lorsque l'ordinateur doit savoir identifier une voix (ouverture électronique des portières d'une voiture à la demande de son propriétaire).

Le moyen le plus efficace que l'on ait trouvé jusqu'ici pour surmonter cette difficulté consiste à « former » la machine, tout à fait comme on apprend à parler aux enfants. La formation commence avec un groupe d'environ cent personnes, ayant le même accent régional. Elles prononcent chacune à leur tour l'ensemble des mots que l'on veut faire reconnaître par la machine. Les différents spectrogrammes enregistrés sont ensuite introduits dans l'ordinateur qui analyse les tableaux graphiques d'un même mot et dresse un tableau intermédiaire, ou tableau type, afin de le mettre en mémoire pour les comparaisons futures. L'autre tâche de l'ordinateur consiste à évaluer la déviation par rapport au spectogramme type.



M€/Theo Bergström

A droite, de haut en bas : deux spectres du mot « sum » (somme), prononcé par une même personne, et la synthèse réalisée par l'ordinateur. A gauche : l'analyse des différences.



Kobal Collection MGM Inc.

Hal, l'ordinateur qui « joue » dans le film « 2001, l'Odyssée de l'espace », est l'exemple classique de la machine parlante de science-fiction.

Par la suite, quand d'autres personnes s'adresseront à la machine, cette dernière sera en mesure de comparer leur voix à celles dont les spectres ont déjà été enregistrés.
Les spectres stockés en mémoire jouent ainsi le rôle de portraits-robots, et ils peuvent se déformer, s'étirer, pour s'adapter aux différentes vitesses d'articulation des mots.
Lorsque l'un de ces spectres enregistrés dans la machine coïncide avec celui que l'ordinateur entend, l'ordinateur reconnaît le mot prononcé.

Ce type de procédé qui, à ce qu'on affirme, permet à la machine de reconnaître les mots émis même dans différentes constructions de phrase, peut être efficace dans 80,90, voire même 98 % des cas, si les mots lui sont adressés par la personne qui a préparé la machine, c'est-à-dire la personne qui a entré les mots dans l'ordinateur. Dès lors, il est possible de construire des machines capables de comprendre des ordres simples, et d'agir en conséquence. On peut facilement imaginer, par exemple, un distributeur de boissons auquel il suffirait de dire «thé», «lait», «sucre», pour obtenir effectivement une tasse de thé au lait sucré. Ce qui peut apparaître comme un prodige, n'est en fait qu'un modeste début, si l'on considère ce que serait l'étape suivante : un ordinateur capable de comprendre, non plus des mots isolés, mais un discours complet fait de phrases.

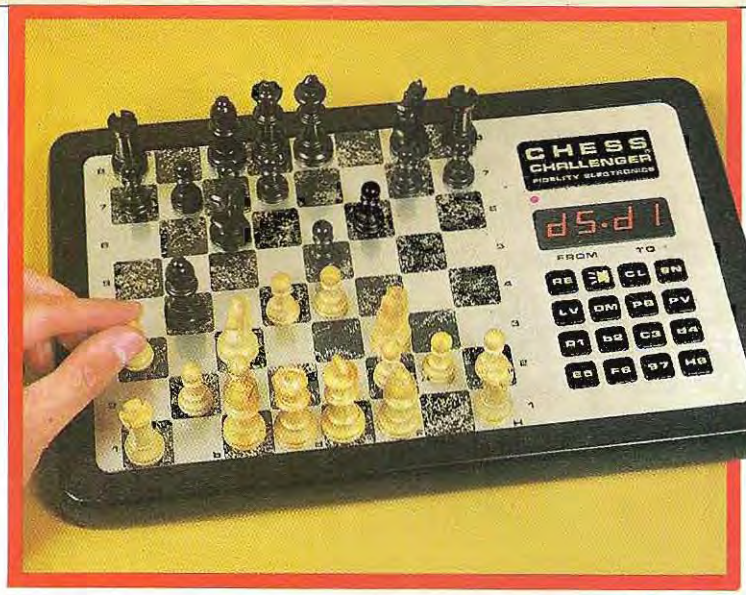
* * *

En 1970, le Département de la Défense américaine demanda à un groupe de chercheurs d'établir un rapport sur ce que l'on pouvait raisonnablement espérer d'un système électronique capable de reconnaître le langage parlé humain. Selon les conclusions du rapport, un système de ce type serait limité à un vocabulaire de 1 000 mots (le vocabulaire d'une personne ayant poursuivi des études jusqu'en classe de seconde peut comprendre entre 5 000 et 10 000 termes) et ne pourrait utiliser qu'une syntaxe grammaticale très simplifiée. Il faudrait en outre que son utilisation soit très spécifique.
L'entrée des mots dans l'ordinateur devrait aussi être réalisée de manière continue et égale, sans couper les mots en syllabes, de façon à ce que la machine ne prenne pas chaque syllabe et chaque son pour un mot distinct. La voix de la personne chargée de préparer ainsi la machine devrait donc être claire, le microphone utilisé de bonne qualité, et l'enregistrement devrait avoir lieu dans une pièce parfaitement silencieuse.

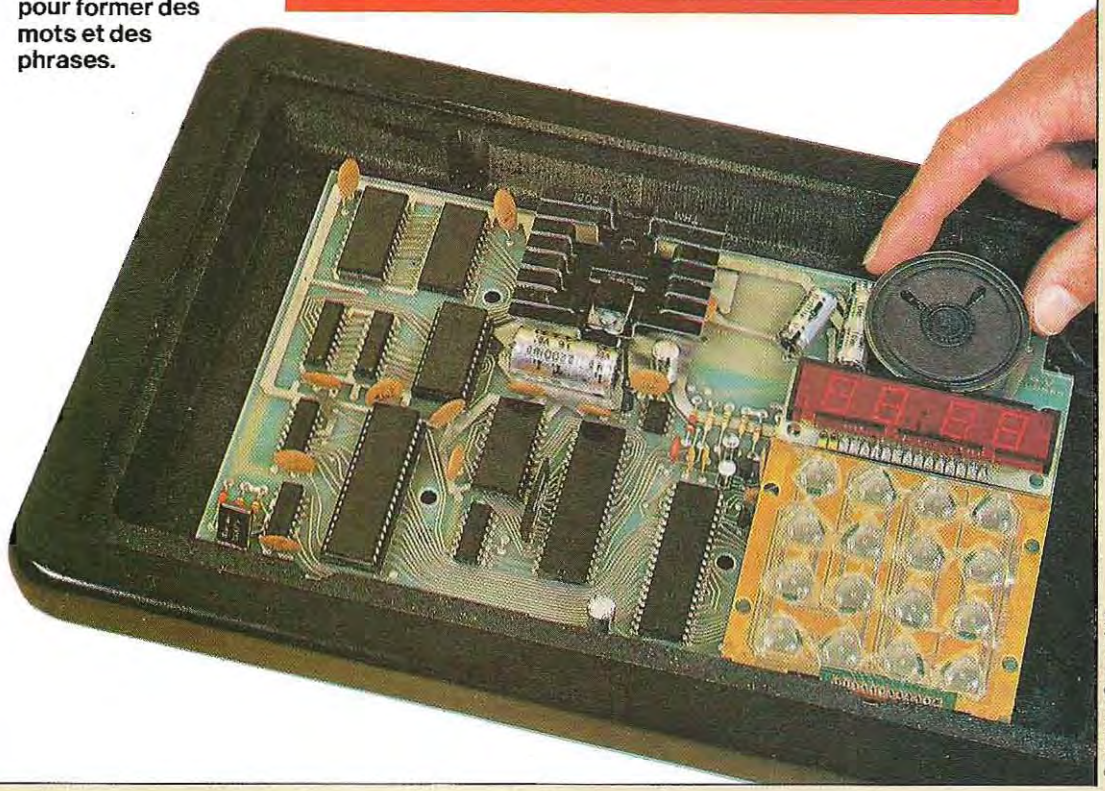
Vers la fin des années 1970, malgré certaines réalisations, les machines capables de reconnaître les mots étaient encore loin de remplir les conditions énoncées dans le rapport américain. Leur «vocabulaire» restait extrêmement pauvre. Deux des meilleurs systèmes mis au point étaient tout juste capables d'utiliser respectivement un répertoire de 39 mots (les 26 lettres de l'alphabet, les chiffres de 0 à 9 et trois ordres) et de 54 mots. On était loin des 1 000 termes annoncés comme une limite de la machine !
Pourtant c'est en 1980 qu'une équipe de chercheurs d'IBM annonça que le but était atteint. Passant deux heures devant un micro, l'opérateur devait prononcer neuf cents phrases constituant la base des connaissances de la machine en langage parlé. Le traitement par la machine d'une phrase prononcée en trente secondes par l'opérateur pouvait durer une centaine de minutes. Sans parler du risque d'erreurs...
En d'autres termes, l'entreprise n'était pas aisée et la commission américaine de 1970 n'avait sans doute pas prévu toutes les difficultés d'un tel projet.

* * *

Le langage humain est en effet extrêmement nuancé. Le discours le plus simple et le plus bref contient pourtant des centaines de petits fragments oraux, les phonèmes, que la machine doit d'abord identifier avec soin avant de les sélectionner. Elle doit aussi déterminer où finit un mot et où commence le suivant. Or, la plupart des gens parlent rapidement, sans nettement détacher les mots (ce qui est parfois justifié par les liaisons que l'on fait habituellement dans certaines langues).
En outre, et en admettant même qu'une machine soit assez perfectionnée pour isoler les mots prononcés et donc pour les reconnaître à coup sûr, il faudra aussi qu'elle soit capable d'accorder à deux mots identiques leur sens respectif.
Si les perfectionnements de la technique per-



Non seulement cette machine joue aux échecs à la perfection, mais elle parle. Elle «annonce» en effet les coups qu'elle est en train de jouer. La synthèse de la parole est obtenue par enchaînement de particules phonétiques enregistrées dans sa mémoire: les sons se trouvent assemblés électroniquement dans l'ordre voulu pour former des mots et des phrases.



Computer Games Ltd

mettent un jour de faire enregistrer à la machine assez d'éléments de base pour qu'elle puisse s'y retrouver dans toutes les subtilités d'une langue, ils devront également accélérer le traitement de ces données par l'ordinateur de telle manière qu'une vraie conversation puisse avoir lieu entre machine et opérateur, se déroulant selon le rythme habituel d'un dialogue entre deux personnes.

La mise au point de toutes ces améliorations techniques demandera sans doute un certain nombre d'années, et les recherches prendront peut-être un tour différent de celui qu'elles ont actuellement.

En tout cas, cette idée d'arriver à fabriquer des ordinateurs qui dialoguent avec l'opérateur est assez fascinante et l'avenir dira si elle relève de la pure science-fiction ou non.

Le stockage des données

Les informations traitées par l'ordinateur ne peuvent pas demeurer en permanence dans la mémoire centrale. Les données présentes dans cette mémoire n'y restent en effet que pour le temps de leur traitement par l'ordinateur et y sont remplacées par les autres informations qui doivent être traitées à leur tour. Il est donc nécessaire de préserver toutes les informations concernées par un traitement avant qu'une nouvelle utilisation de la machine ne vienne effacer le contenu de la mémoire. On le fera en les stockant sur les supports adéquats dont nous reparlerons. Un ordinateur peut saisir, traiter et fournir un très grand nombre d'informations, et cela à une vitesse extrêmement rapide. Aussi est-il nécessaire d'enregistrer ces informations à l'aide de supports de très grande capacité. Ces données devront être archivées et enregistrées à la sortie de l'ordinateur de façon à pouvoir être réutilisées en entrée lors de traitements ultérieurs.
Rappelons en outre que la moindre coupure de l'alimentation électrique de l'ordinateur entraîne la disparition des données contenues dans la mémoire centrale. Cela souligne encore la nécessité de stocker les données sur des supports totalement fiables, capables de conserver leur contenu pendant plusieurs années, et même sans alimentation électrique. Prenons un exemple : les salaires des employés d'une société sont calculés une fois par mois et les données de la paie doivent être disponibles chaque mois à une date déterminée (le 20 et le 21, par exemple). Il serait impensable d'encombrer la mémoire centrale de l'ordinateur, si grande soit-elle, avec ces millions de caractères pendant tout le reste du mois. De plus, il est bon qu'après le calcul du salaire de M. Leblanc, le résultat de ce calcul soit archivé ailleurs, libérant ainsi l'espace mémoire correspondant pour les données qui serviront à calculer la paie de M. Rousseau.
On voit bien là l'importance capitale du stockage des données sur des supports externes à l'ordinateur et que l'on appelle des mémoires auxiliaires ou mémoires de masse.

Les mémoires de masse

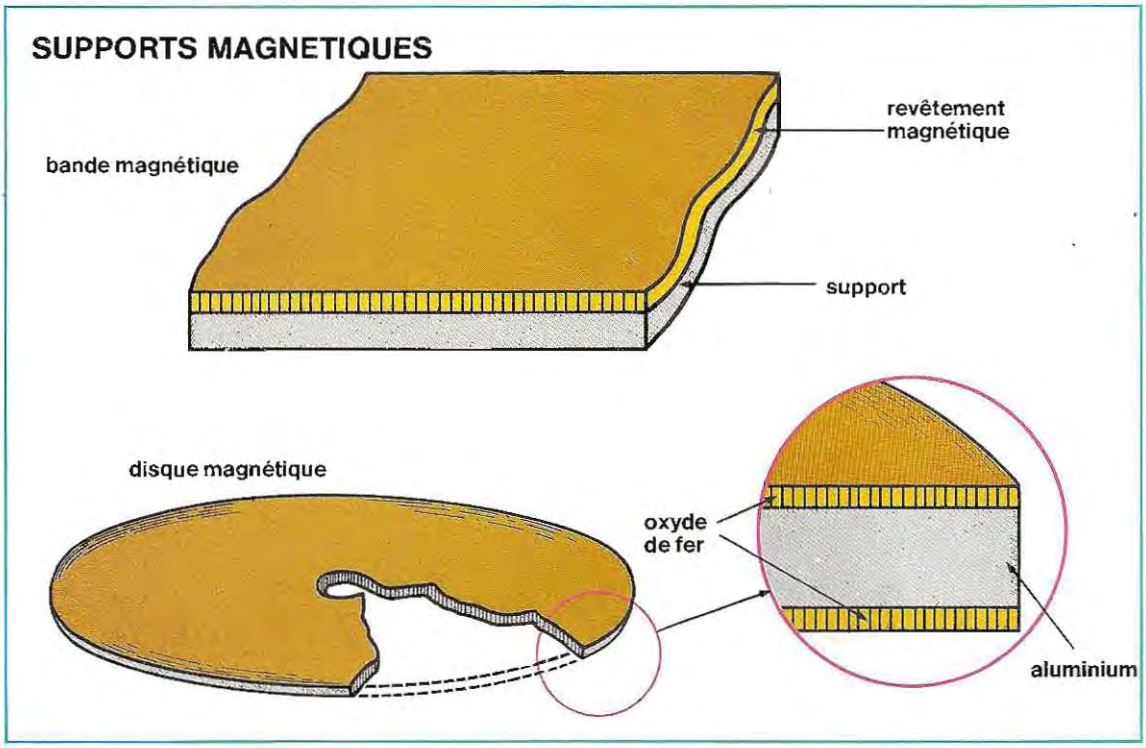
Laissons de côté le stockage des données sur cartes ou sur bandes perforées. Ce sont là des supports aujourd'hui dépassés ou, du moins, désormais réservés à des usages particuliers. Etudions plutôt les supports qui permettent l'enregistrement magnétique des données.. Ces supports ont en commun quelques caractéristiques générales. La lecture et l'écriture des données s'effectuent au moyen d'appareils spéciaux munis de dispositifs appelés têtes de lecture ou d'écriture. L'accès aux portions élémentaires du support magnétique se fait grâce à un mouvement mécanique de la mémoire de masse et de la tête située à proximité de la surface magnétisable (voir schéma page 223).
Les mémoires de masse se distinguent selon la manière dont elles permettent l'accès aux données qu'elles détiennent, c'est-à-dire selon la méthode qu'il faut utiliser pour repérer une information qu'on veut lire ou pour positionner une information qu'on veut écrire. On parlera ainsi d'accès **séquentiel** et d'accès **direct**.
L'accès est séquentiel lorsque, pour trouver une information, on doit nécessairement passer en revue tous les enregistrements existants jusqu'à la rencontre de la donnée recherchée.
Inversement, l'accès est direct lorsqu'il est possible de retrouver immédiatement l'information recherchée à l'aide de son adresse physique (numéro du disque, numéro de la piste et numéro du secteur).
Nous approfondirons ces notions dans la partie consacrée aux fichiers.

Les bandes magnétiques

On utilise beaucoup les bandes magnétiques pour l'enregistrement des très grandes masses de données, pour la copie des fichiers volumineux que l'on veut sauvegarder et pour faciliter le traitement de données organisées en fichiers séquentiels.
La bande est constituée d'un support plastique souple composé de nylon, de polyester, de mylar ou d'acétate de cellulose. Elle a, en général, une largeur d'un demi-pouce (12,7 mm) et sa longueur, variable, est couramment de 730 m. L'une de ses faces est recouverte d'un revêtement magnétique très mince. Elle ressemble donc beaucoup aux bandes que nous utilisons dans nos magnétophones.
Pour consulter ces bandes magnétiques

Italcable

Ces dérouleurs de bande utilisés dans un centre de calcul nécessitent une intervention manuelle.



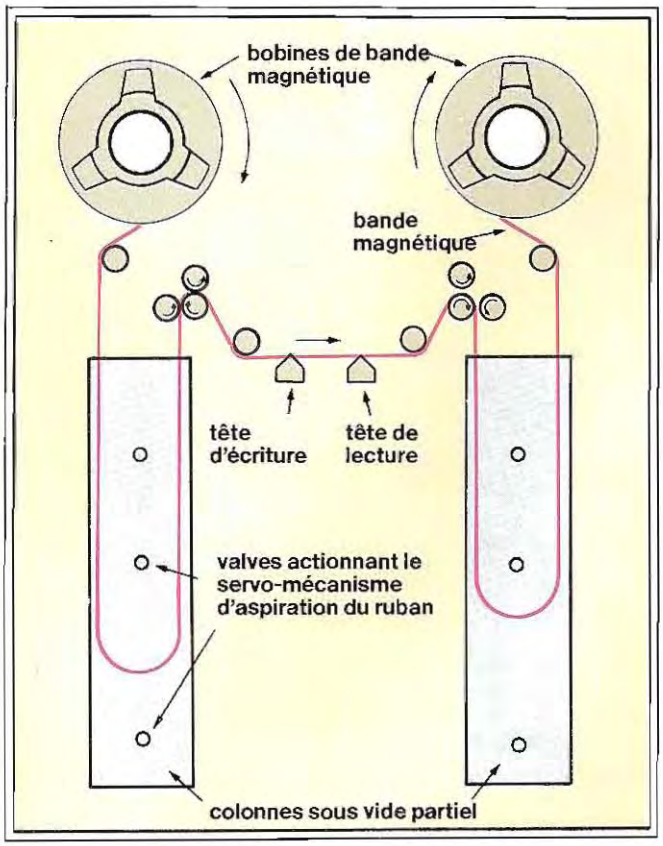
Sann

Exemple de mémoire de masse : dérouleurs à chargement automatique.

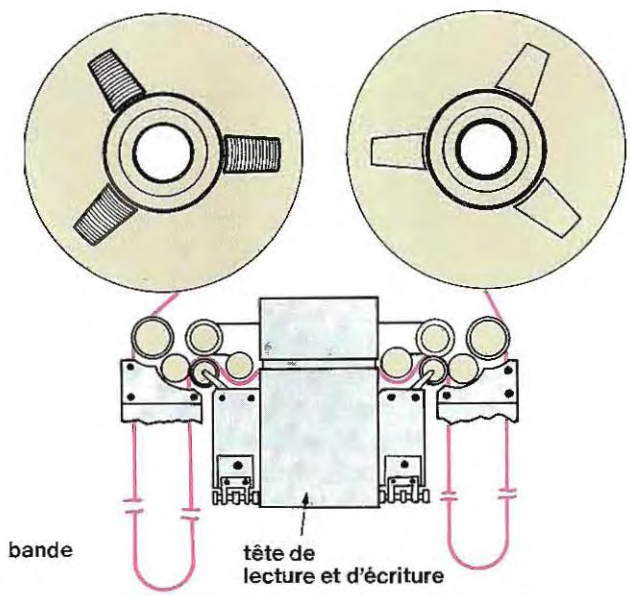
(lecture) ou y enregistrer des données (écriture) on a recours à des appareils spéciaux, les **unités de bande**, ou **dérouleurs**, considérés comme des unités périphériques de l'ordinateur. Le dispositif qui prend en charge la lecture et l'écriture ainsi que l'effacement de la bande est l'organe essentiel de l'unité de bande (voir schéma page 223). L'enregistrement se fait par la magnétisation de points qui peuvent prendre deux états associés aux valeurs 1 (point magnétisé) et 0 (point non magnétisé).

On peut considérer que la surface d'une bande magnétique est divisée en **pistes** (tracks en anglais) le long desquelles se succèdent les points magnétisables, de valeur 0 ou 1. Les bandes peuvent comporter sept ou neuf pistes, l'une d'elles étant utilisée comme piste de parité. Le bit (point) de parité prend la valeur 1 ou 0 selon que le nombre total de bits ayant la valeur 1 dans le caractère considéré est pair ou impair (parité transversale). Il existe le même type de contrôle* pour les bits d'un segment de piste, ou bloc, c'est la clé longitudinale (LRC, Longitudinal Redundancy Control).

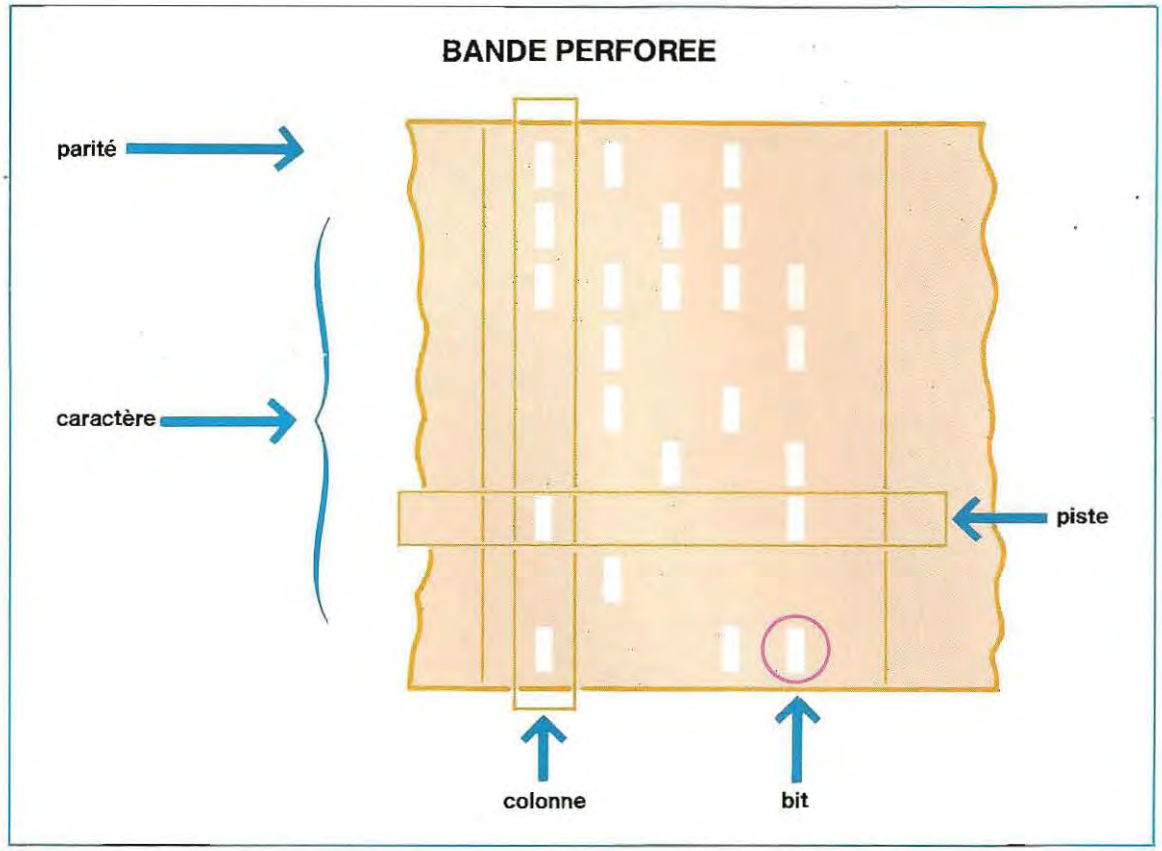
*Cette méthode est parfaitement comparable à la méthode de contrôle de transmission utilisée en code ASCII (bit de parité, voir de la page 117 à la page 119).



SCHEMA D'UNE UNITE DE BANDE



STRUCTURE D'UNE UNITE DE BANDE



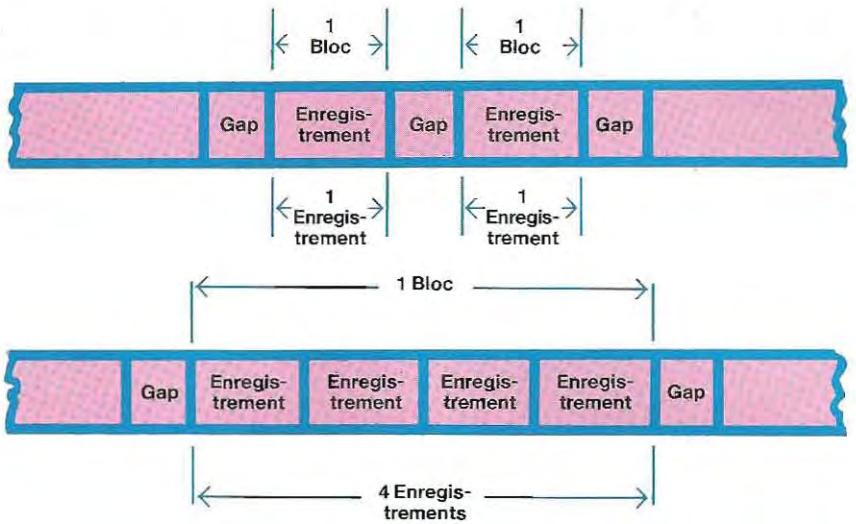
Le nombre de caractères que l'on peut enregistrer sur une bande magnétique dépend de la longueur de la bande (600, 1 200, 2 400 ou 3 600 pieds) et de la quantité de caractères que l'on peut enregistrer par unité de longueur (appelée **densité d'enregistrement**).

Les densités d'enregistrement les plus courantes sont 800, 1 600 et 6 250 caractères par pouce. Ces chiffres permettent d'évaluer la capacité théorique d'une bande. Ainsi, une bande de 2 400 pieds (730 mètres environ) enregistrée avec une densité de 800 caractères par pouce, a une capacité théorique de 24 000 000 de caractères.

En fait, l'enregistrement des caractères sur la bande n'est pas continu car les caractères sont regroupés en **blocs**, ou **enregistrements physiques**, séparés les uns des autres par une portion de bande non utilisée appelée **espace entre blocs**, ou **gap**. En général, la taille des blocs est fonction de l'organisation des données et de la capacité de la mémoire centrale de l'ordinateur. Quant à la longueur de l'espace entre blocs, elle dépend

du type d'unité de bande et est assez courte (par exemple, 0,6 ou 1,2 pouce).
La capacité réelle d'une bande magnétique dépend également de la longueur des blocs et de la fréquence des gaps. Ces espaces ont une influence importante sur la capacité et le

EXEMPLES D'ENREGISTREMENT DE DONNEES SUR BANDE MAGNETIQUE



débit réel des bandes magnétiques.
Comme nous l'avons vu plus haut, les unités de bande sont des périphériques reliés à l'ordinateur.
La transmission des caractères par câble (canal) à grande vitesse permet la commutation entre les deux appareils. La vitesse de transfert est très rapide : 40000, 160000, 320000, voire 1250000 octets par seconde.
Les bandes doivent, pour pouvoir être utilisées en toute sécurité, comporter une certaine longueur non enregistrable au début et à la fin (environ 3 et 4 mètres) qui leur permet, en début de bande, de garnir le mécanisme d'entraînement, et, en fin de bande, d'éviter un débobinage complet. Le début et la fin de la partie utilisable de la bande sont indiqués par des marques en aluminium de 25 mm environ, appelés **indicateur de début** et **indicateur de fin de bande**. Ce sont des plaquettes réfléchissantes, collées sur le côté non magnétique de la bande et qu'on appelle des **stickers.** Elles permettent aux cellules photoélectriques placées dans l'unité de bande de déterminer l'endroit où peut commencer **(point de chargement),**

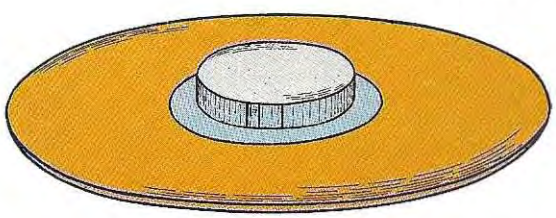
Rhône-Poulenc Italie

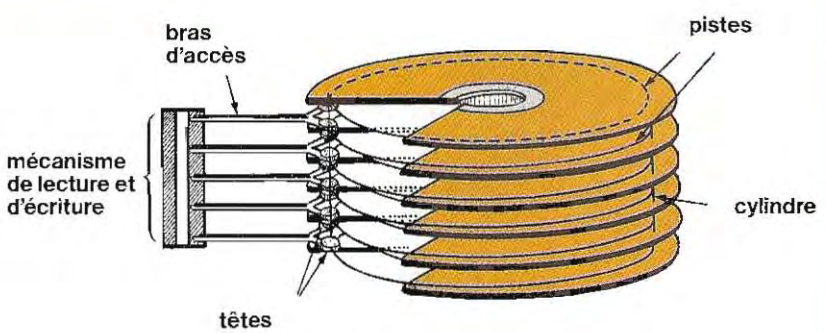
L'équilibrage sur deux plans confère une parfaite stabilité au disque pendant sa rotation dans le mécanisme d'entraînement (drive).

Rhône-Poulenc Italie

La certification numérique (contrôle de qualité à l'aide d'un oscilloscope) garantit la bonne adaptation du dispac à l'ordinateur.

DISPAC





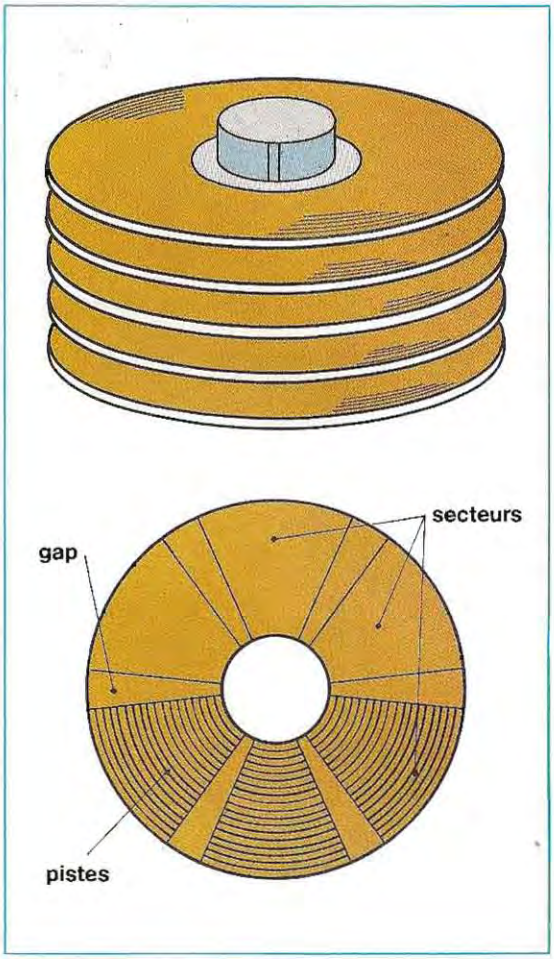
et l'endroit où doit prendre fin l'opération de lecture et d'écriture

Les disques magnétiques

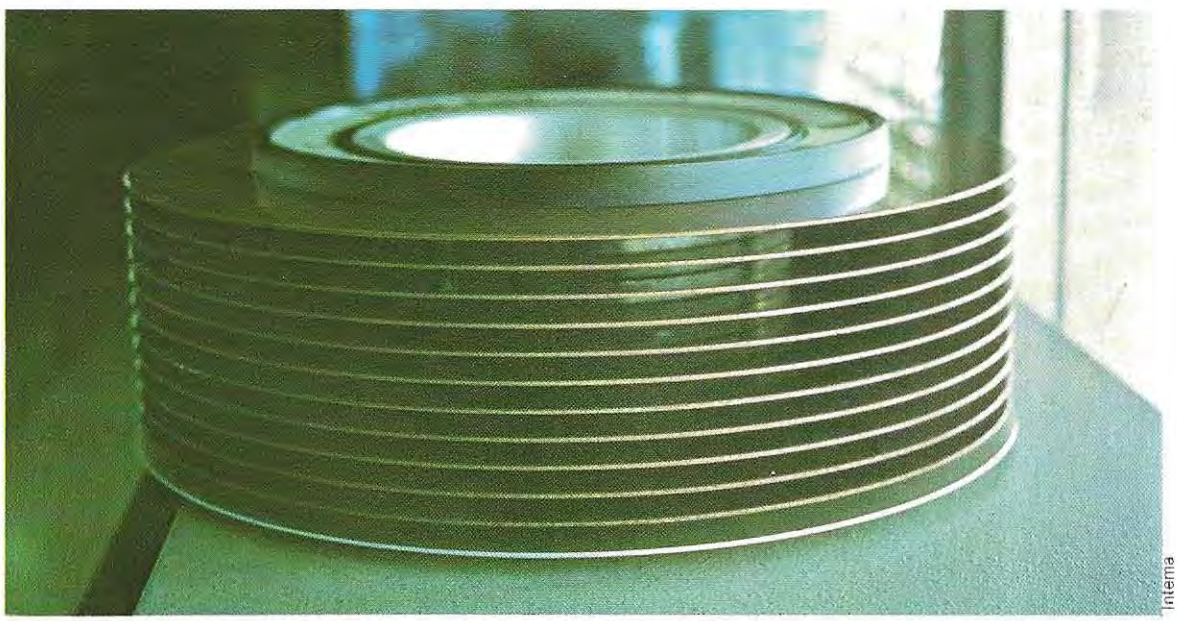
Pour conserver les grandes quantités de données, on utilise aussi les disques magnétiques, qui sont des mémoires de masse à accès direct. Ils se différencient des bandes magnétiques par la rapidité avec laquelle ils permettent d'accéder à l'information recherchée. Un disque magnétique est constitué d'un mince disque, généralement en aluminium, dont les deux faces sont enduites d'un revêtement magnétique (schéma p. 226). Plusieurs disques peuvent être empilés sur un axe central et tourner à la même vitesse. Un ensemble de ce type est un dispac, ou disk-pack (voir schéma page 227). Si le montage est définitif, on dit que l'unité est à disques fixes ou à disques non amovibles. Si les mécanismes de lecture et d'écriture sont rétractables, on peut changer le dispac. On parle dans ce cas d'unité à disques amovibles.

Les disques sont séparés par des espaces dans lesquels se positionnent les têtes de lecture et d'écriture. Par un mouvement horizontal d'avant en arrière elles se placent en face des pistes.

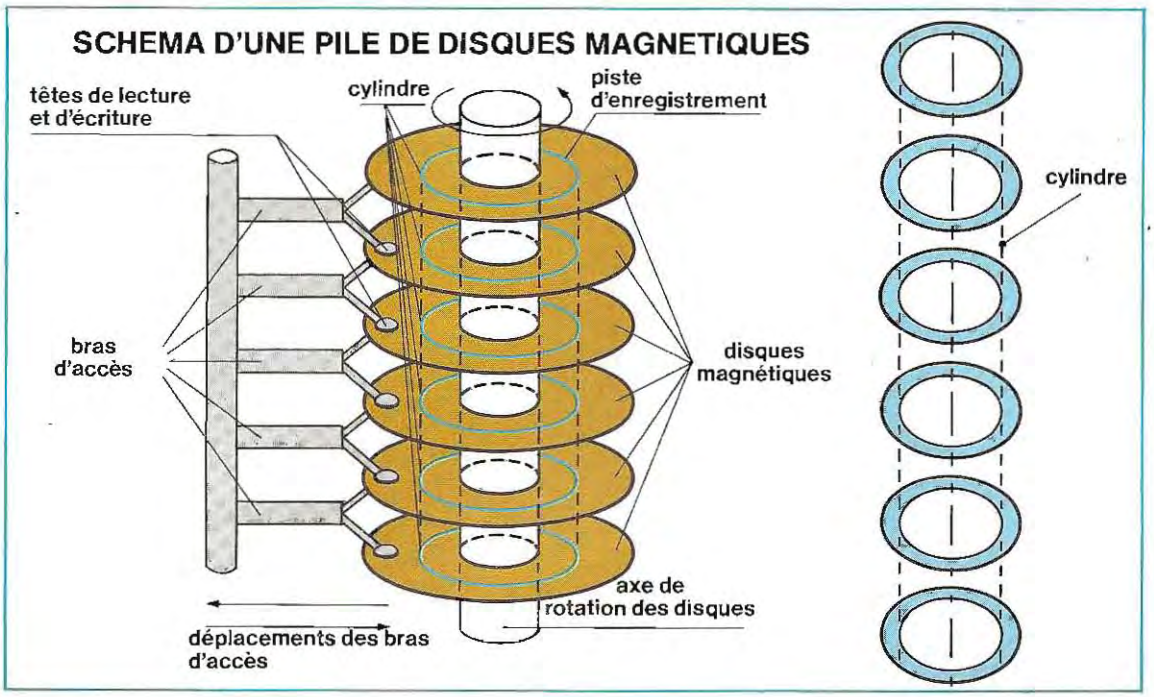
Les informations sont enregistrées sur les pistes concentriques suivant les mêmes principes physiques que sur les bandes. Chacune des faces d'un disque rassemble quelques centaines de pistes divisées en secteurs, eux-mêmes séparés par des intervalles (ou gaps) dans lesquels aucune donnée n'est enregistrée. Dans le cas



Le dispac constitue la mémoire de masse à accès direct des gros systèmes.

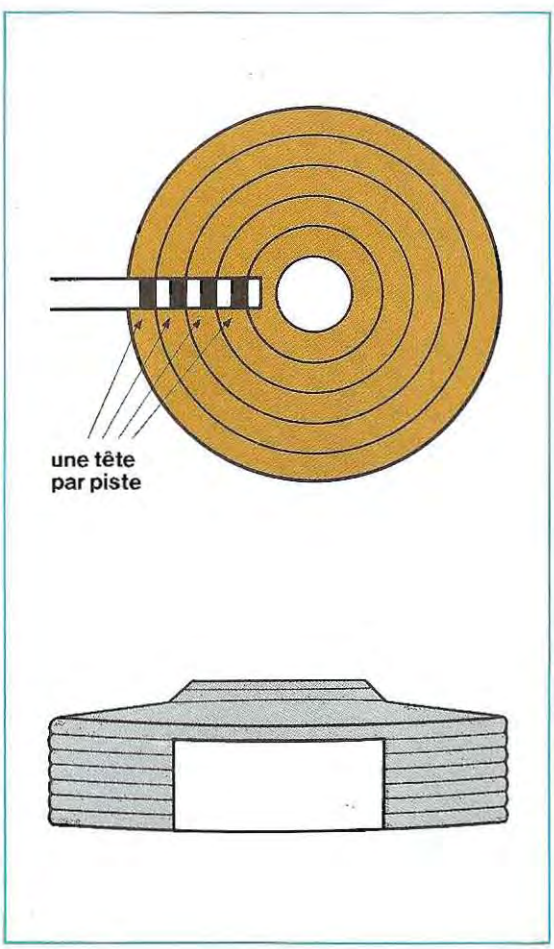


Intema



d'un dispac, c'est un seul bras mécanique qui porte la tête correspondant à la face inférieure d'un disque et la tête correspondant à la face supérieure du disque placé au-dessous. Le déplacement de tous les bras étant solidaire, les têtes se trouvent toutes placées sur une verticale, chacune étant positionnée devant la même piste de chaque disque. L'ensemble de ces pistes constitue un **cylindre** (voir schéma en haut de la page). Il est donc possible de lire ou d'écrire en même temps sur toutes les pistes qui appartiennent à un même cylindre, sans déplacer les bras, ce qui réduit considérablement le temps d'accès. On peut encore réduire ce temps en utilisant des têtes fixes, ce qui élimine le mouvement du bras d'accès (figure ci-contre). La rotation des disques fait que le contenu d'un secteur ne peut être lu ou écrit que lorsque ce secteur passe au-dessus (ou au-dessous) de la tête de lecture et d'écriture. Le temps d'accès aux informations varie donc selon l'ensemble de facteurs suivants :

- temps de positionnement de la tête sur la piste où se trouve le secteur recherché ;
- temps d'attente nécessaire à ce que le secteur se trouve au-dessus ou au-dessous de la tête ;
- vitesse de transfert des données (lecture ou écriture).



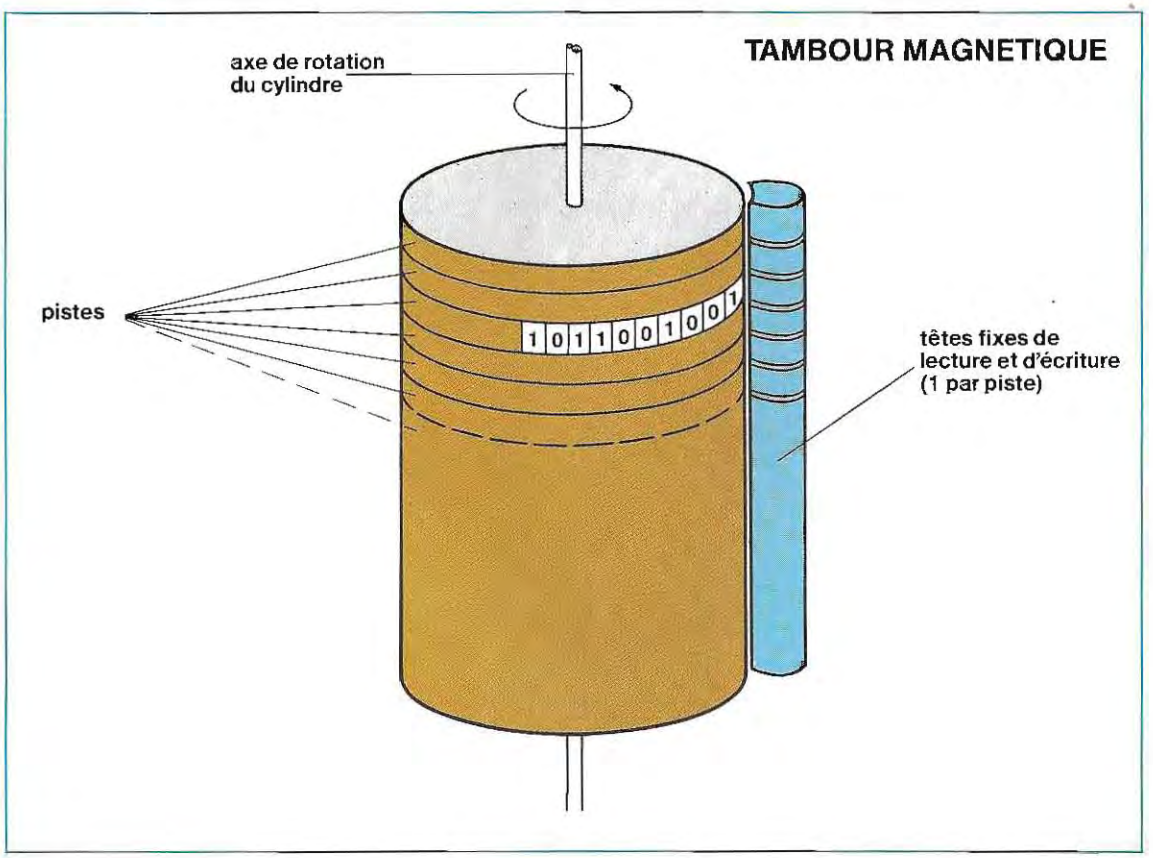
Ces temps sont d'ailleurs extrêmement brefs, puisqu'ils oscillent entre une dizaine et une centaine de millisecondes. La capacité d'un module varie entre quelques millions et 500 ou 900 000 000 de caractères.
L'échange des informations entre l'unité de disques et l'ordinateur s'effectue à des vitesses de transfert comprises entre quelques dizaines de milliers de caractères et 1 800 000, voire 2 000 000 caractères par seconde
Les disques fixes sont, en général, enfermés dans des conteneurs en plastique pourvus des ouvertures nécessaires aux bras d'accès (voir schéma en bas de la page 228).

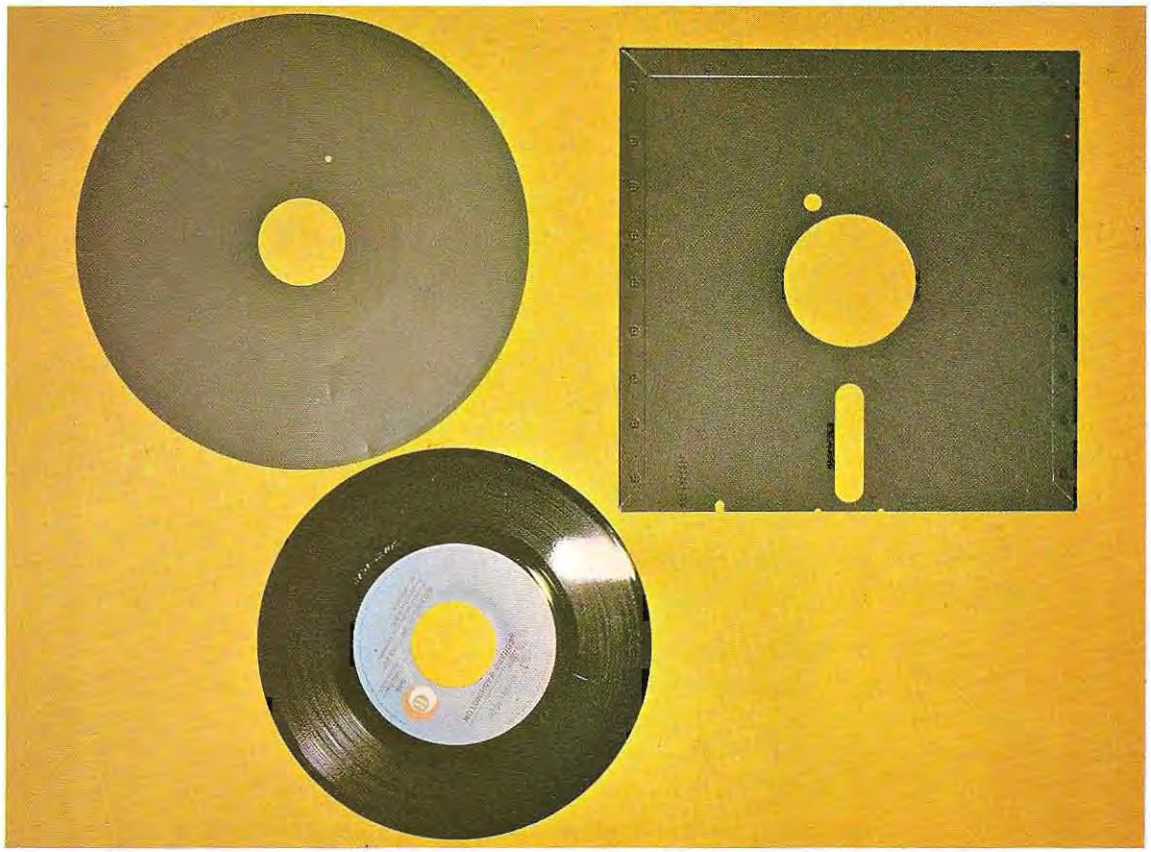
Les tambours magnétiques

En raison de leur coût élevé, les tambours magnétiques servent plutôt sur les gros ordinateurs.
Il s'agit de mémoires à accès direct constituées d'un cylindre métallique, dont la surface latérale est recouverte d'une mince couche de substance ferromagnétique, et qui tourne à une vitesse élevée et constante.
Une tête de lecture et d'écriture est positionnée en face de chaque piste, à une distance de quelques centièmes de millimètre. On peut subdiviser les pistes en secteurs.
La plupart du temps les têtes sont fixes et il y en a autant que de pistes (voir schéma ci-dessous).
La capacité d'un tambour magnétique dépend de sa taille, du nombre de ses pistes et de la densité d'enregistrement des données sur les pistes.
Pour calculer le temps d'accès à une information, il faut additionner le temps de lecture proprement dit et le temps d'accès à la portion de piste concernée.

Les disquettes ou disques souples

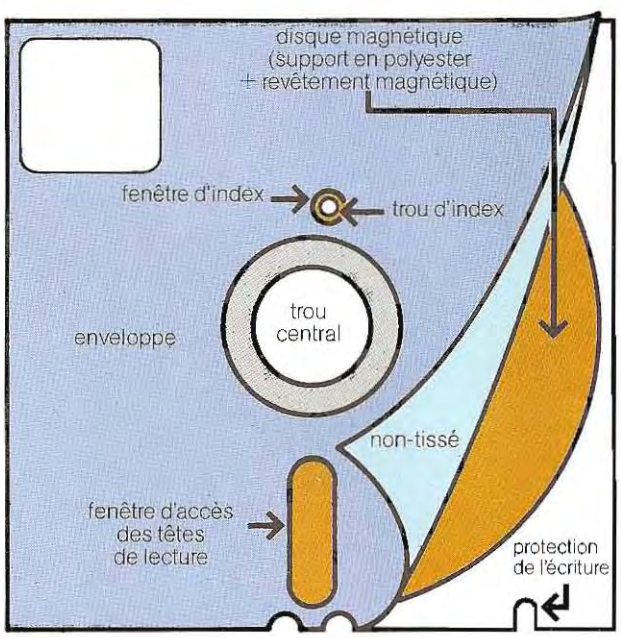
Les disquettes (floppy disk en anglais) reposent sur le même principe que les disques mais elles coûtent moins cher et elles sont plus faciles à manipuler.
Fabriquées dans une matière souple, les disquettes sont protégées par une enveloppe pourvue d'une petite fenêtre qui permet l'accès des têtes de lecture et d'écriture.





Centre THC

Une disquette (floppy) et un 45-tours ordinaire. Cette disquette 8"-DF-DD (8 pouces - double face - double densité) a une capacité d'environ 1 200 000 caractères.



STRUCTURE D'UNE DISQUETTE FLEXETTE FABRIQUEE PAR RHONE-POULENC

L'enveloppe
Celle-ci est fabriquée dans un PVC dont la résistance à la température est exceptionnelle.

Le non-tissé
La doublure en non-tissé de l'enveloppe est fixée de manière à assurer la rotation du disque à vitesse constante et son centrage parfait lors du chargement dans le lecteur.

Le disque
Il est composé d'une couche de polyester enduite d'une émulsion magnétique à haute densité.

Un petit trou sert à placer le disque dans son logement de manière à ce que les têtes soient alignées sur le point de départ des pistes.
Les disquettes sont également divisées en pistes subdivisées elles-mêmes en secteurs.
On distingue divers types de disquettes, selon qu'elles permettent :

1 / un enregistrement sur une seule face ou sur les deux ;
2 / un enregistrement à simple ou à double densité.

Selon les différentes combinaisons de ces éléments, la capacité d'un disque souple varie entre 80 000 et 1 200 000 octets (caractères). De plus, on peut loger plusieurs disquettes sur un même chargeur.
Ce type de mémoire est habituellement utilisé avec profit sur les micro-ordinateurs et les ordinateurs individuels. Leurs dimensions les plus courantes sont 5 pouces 1/4 et 8 pouces.

Les fichiers

L'ensemble des données enregistrées dans les mémoires de masse constituent un fichier (file en anglais). Un fichier se caractérise par l'homogénéité des informations qu'il contient, par exemple, les noms des employés d'une entreprise, des élèves d'une école ou les rubriques des articles d'un magasin.
Une donnée est un ensemble de un ou plusieurs caractères qui désigne, de façon précise et univoque, un élément quelconque. Ainsi, par exemple :

- **une personne :** Dupont Marc
- **un nombre :** 235
- **un sigle :** A2

Les données sont donc :

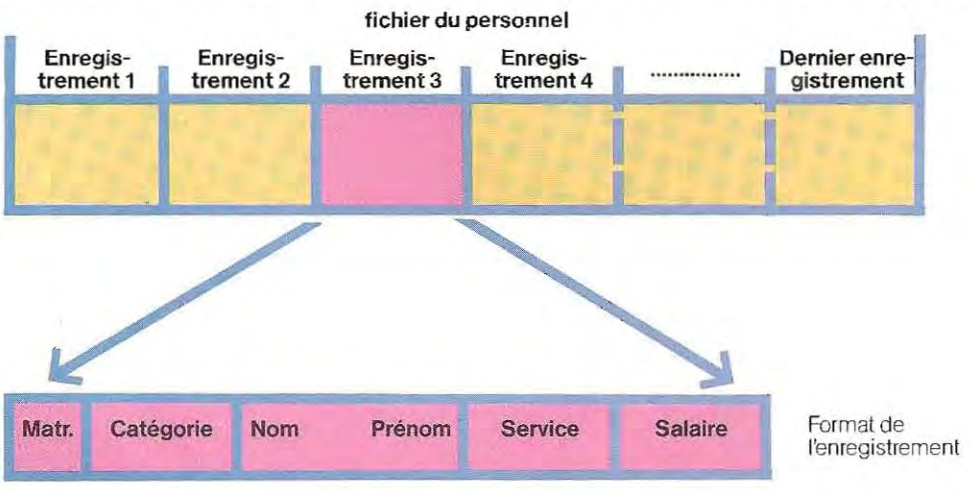
- **alphabétiques,** lorsqu'elles se composent de lettres et d'espaces ;
- **numériques,** si elles se composent uniquement de chiffres ;
- **alphanumériques,** lorsqu'elles se composent de lettres, de chiffres et d'espaces.

On appelle **zone** (field en anglais) l'emplacement de la mémoire qui est occupé par une donnée, et on lui donne, en général, un nom.

nom 40 caractères	
nom de famille 25 car.	**prénom 15 car.**

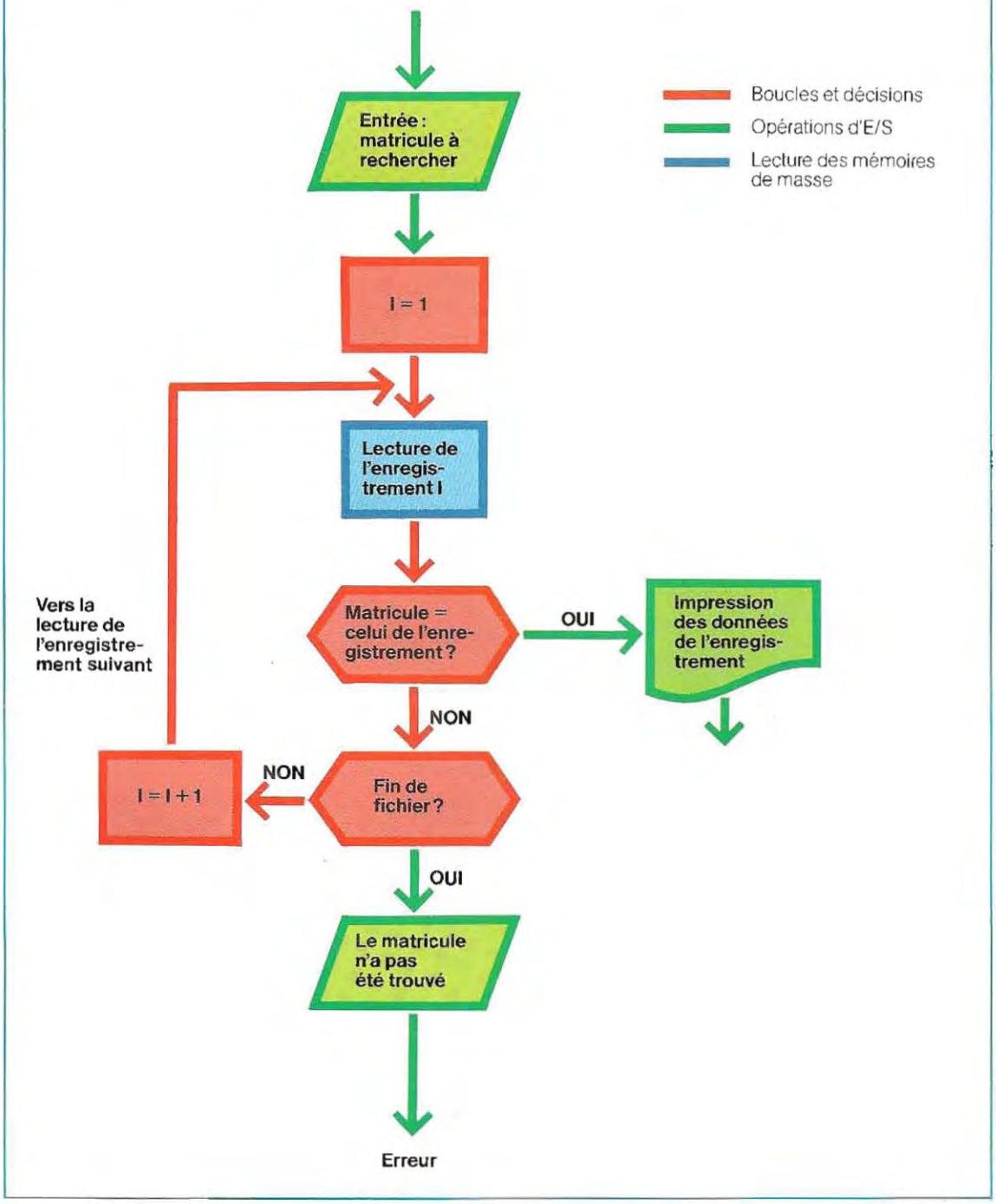
Dans cet exemple, la zone est divisée en deux éléments ou **sous-zones,** de longueur déterminée : vingt-cinq caractères pour la sous-zone nom de famille et quinze pour la sous-zone prénom. Si nous prenons le cas de M. Dupont Marc, nous constatons que le nom Dupont n'occupe que six des vingt-cinq caractères disponibles, et que le prénom Marc n'a besoin que de quatre des quinze caractères de la seconde sous-zone. Il reste donc de la place inutilisée dans les deux sous-zones. Leur taille a été déterminée pour qu'elles puissent accueillir les noms et les prénoms à traiter les plus longs. Lorsque, comme

EXEMPLE DE STRUCTURE DES DONNEES DANS UN FICHIER DU PERSONNEL



ici, la longueur d'une zone ne varie pas, on parle de zone de **longueur fixe.**
Lorsque, au contraire, l'espace mémoire utilisé est directement lié au nombre de caractères dont se compose la donnée, on dit qu'il s'agit d'une zone de **longueur variable.**
Dupont Marc, 11 caractères (y compris l'espace) ; Lhenry Marguerite, 17 caractères (y compris l'espace).
L'ensemble des zones qui qualifient le

ORGANIGRAMME D'UNE RECHERCHE A L'AIDE D'UNE CLE



Marka

Les enregistreurs magnétiques associés aux ordinateurs sont parmi les plus perfectionnés.

contenu de la donnée s'appelle un **enregistrement** (record en anglais). Imaginons que nous voulons construire un fichier contenant les principales informations administratives sur le personnel d'une société.
Les données à enregistrer sont les suivantes :

1 / numéro de matricule (3 caractères) ;
2 / catégorie (2 caractères) ;
3 / nom et prénom (40 caractères) ;
4 / service (2 caractères) ;
5 / salaire (7 caractères).

L'ensemble de ces informations (1,...5) constitue un enregistrement et l'ensemble des enregistrements est un fichier. Tout enregistrement du fichier comporte cinq zones, qui contiennent chacune l'un des cinq enregistrements définis ci-dessus. Le nombre total d'enregistrements est égal au nombre d'employés. La structure de ce fichier est illustrée par le schéma de la page 231.
En général, on utilise une ou plusieurs des zones comme **clé** afin de pouvoir reconnaître avec certitude les données d'un enregistrement. Dans notre exemple, les employés sont identifiés par leur numéro de matricule et celui-ci peut donc servir de clé d'accès aux données. Il faudra alors, pour retrouver les informations concernant un membre du personnel, entrer le numéro de son matricule. L'ordinateur commencera à lire le fichier à partir du premier enregistrement en comparant le matricule contenu dans celui-ci avec le matricule entré par l'opérateur ; et ainsi de suite, avec les autres enregistrements jusqu'à ce qu'il y ait égalité entre les deux numéros de matricule. L'ordinateur fera alors apparaître les données contenues dans l'enregistrement et terminera son travail. L'organigramme de ce programme simple est représenté page 232. On ne peut utiliser cette méthode de recherche (lecture séquentielle de tous les enregistrements) que lorsque les données ne sont pas très nombreuses. Pour les fichiers importants, il existe des méthodes plus rapides mais également plus complexes.
De même que les zones qui le composent, un enregistrement peut être soit de longueur fixe, s'il ne contient que des zones de longueur fixe ; soit de longueur variable, s'il contient au moins une zone de longueur variable ou un

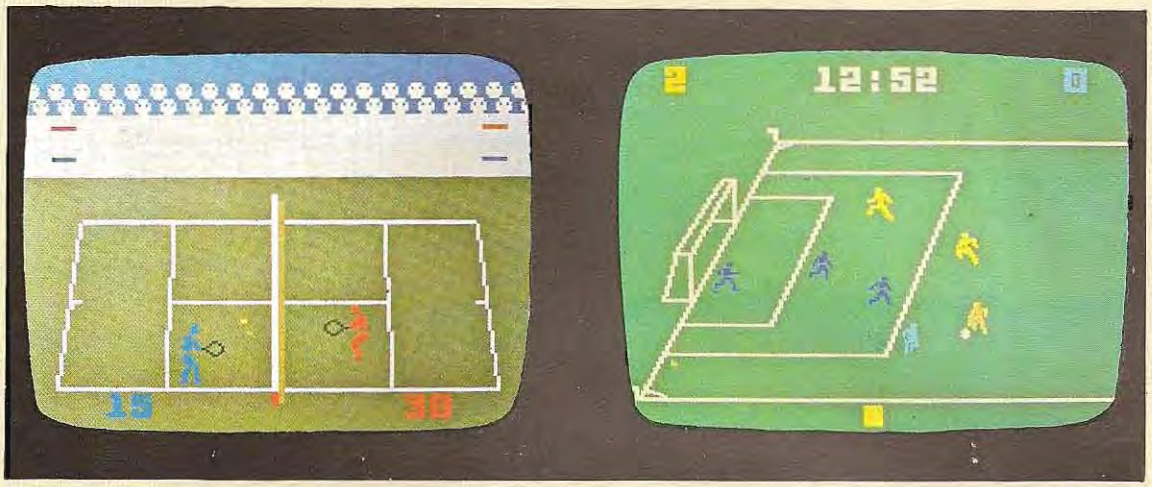
L'homme qui inventa le jeu vidéo

Bell, Edison, Fermi ; ces noms vous disent-ils quelque chose? Et celui de Higinbotham ? Non ? Pourtant, c'est cet homme qui, par une journée de 1958, en faisant apparaître une image sur son oscilloscope, inventa le jeu vidéo. Je dois avouer que, pour en arriver à cette conclusion, il m'a fallu consulter des montagnes de manuscrits et me faire une idée plus claire de l'histoire du jeu vidéo. Si vous entendiez parler d'un jeu vidéo antérieur à 1958, faites-le moi savoir ; sinon, acceptez que Willy Higinbotham occupe la place qui lui revient dans l'Histoire. Car c'est bien lui qui a inventé le jeu vidéo, n'en déplaise aux grandes sociétés qui s'en disputent la paternité.

* * *

Dans les années 1950, les visites guidées de laboratoires de recherche étaient plutôt ennuyeuses : on se bornait à vous montrer quelques photographies pour illustrer tel ou tel aspect des travaux.
C'est pourquoi Willy, qui avait déjà pris conscience de ses dons de physicien à la Cornell University et d'électronicien au MIT (Massachusetts Institute of Technology), décida de les rendre un peu plus vivantes. « Pourquoi ne pas faire participer les visiteurs à des petits jeux sur un écran, afin qu'ils puissent toucher des instruments et s'amuser à appuyer sur des boutons ?» se demandait-il. Sitôt dit, sitôt fait : Willy et ses collègues mirent au point le déroulement d'une partie de tennis destinée à l'écran de cinq pouces d'un oscilloscope.

A l'époque, les premiers ordinateurs numériques avaient déjà fait leur apparition et l'unité de recherche où travaillait Willy en avait justement construit un. Il utilisa, cependant, pour son petit jeu, un ordinateur analogique qui affichait les informations grâce à des tensions de valeur différente, et non à des impulsions ON/OFF (ouvert-fermé). L'ordinateur était connecté à un ensemble non programmable de relais électromécaniques, de potentiomètres, de résistances, de condensateurs et d'amplificateurs opérationnels. Willy reconnaît lui-même que ce rapide « bricolage » n'était pas des plus élégants mais, en tout cas, il fonctionnait très bien.
L'écran présentait un terrain de tennis vu de côté, un « T » renversé figurant le filet. Chacun des deux spectateurs désirant jouer tenait à la main un boîtier muni d'une manette et d'un bouton. La manette servait à commander l'angle de tir et le bouton à déclencher la frappe de la balle. En outre, le jeu tenait compte de certains éléments physiques inhérents au jeu réel : ainsi, quand la balle arrivait dans le filet, elle rebondissait moins fort et moins longtemps que lorsqu'elle frappait le sol.
En dépit de sa simplicité, ce jeu était passionnant. Les collègues d'Higinbotham se souviennent encore de l'enthousiasme des étudiants qu'il n'y avait plus moyen de faire partir. Je m'imagine sans aucune difficulté mon chef, Dave, les yeux rivés sur l'écran, totalement captivé par le déroulement du jeu. La balle et les limites du terrain étaient sans cesse redessinées sur l'écran, à un rythme tel qu'on ne percevait pas le scintillement de l'image. Le procédé est toujours utilisé aujourd'hui. En revanche, le système employé pour représenter le déplacement de la balle n'a pas été repris. Nous allons essayer d'en expliquer brièvement le principe.
On peut utiliser un oscilloscope pour visualiser des graphiques animés : un point lumineux dont les coordonnées sont proportionnelles à la valeur des tensions appliquées aux entrées x et y, se déplace sur l'écran. Partant de ce principe, Higinbotham utilisa des amplificateurs opérationnels pour réaliser un circuit qui simulait la trajectoire d'une balle et son impact sur le terrain. Au moment du choc, un relais était mis en action, et inversait la polarité d'un autre amplificateur. Ainsi, la balle repartait en arrière en donnant l'impression de rebondir. Elémentaire mais efficace, cette méthode permettait même de déterminer si la balle avait touché le filet car elle rebondissait alors à une vitesse différente (inférieure), ce qui conférait au jeu l'apparence de la réalité. La vitesse de la balle diminuait constamment à cause de la vitesse du vent et était simulée directement par une résistance de 10 mégohms.
La hauteur du filet et la longueur du terrain étaient réglables, et une manette permettait aux joueurs de choisir le côté du service. Grâce à une pression sur le bouton du boîtier les joueurs



Marka

étaient assurés de frapper la balle, mais il fallait, toutefois, s'y prendre au bon moment et adopter le bon angle de frappe, sinon la balle n'allait pas dans le camp adverse.
Ces détails avaient été si bien étudiés que le jeu fut, deux années durant, le point fort des Laboratoires de Brookhaven et le « clou » de toutes les visites. Un jour, pourtant, il finit par être démonté.

J'ai demandé à Willy comment il était possible qu'il n'ait pas songé alors à faire breveter cette invention, lui qui était déjà détenteur de plus de vingt brevets dont le gouvernement était, depuis, devenu propriétaire. « Et bien, c'est peut-être mieux ainsi, sinon, aujourd'hui, on ne pourrait produire de jeu vidéo que sous licence gouvernementale ». Cette idée semble amuser Willy.
La société Magnavox, en revanche, ne trouve pas cela drôle du tout car les droits sur les jeux vidéo représentent des millions de dollars, et elle a déposé une demande de brevet pour des jeux vidéo basés sur le rebondissement d'une balle... Higinbotham n'a pas l'intention de tirer profit de cette affaire, mais il est tout de même décidé à défendre sa réputation et son prestige personnels.

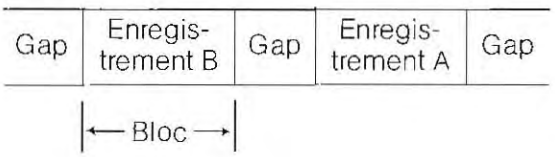
* * *

Voyons d'un peu plus près qui est cet homme. Higinbotham venait de décrocher son diplôme de physique à la Cornell University lorsque la Seconde Guerre mondiale éclata. Invité à participer aux recherches du MIT sur les radiations, il travailla à la mise au point d'une technique nouvelle, promise à un large succès, et alors connue sous le nom de Radio Detecting and Ranging. Cela ne vous dit peut-être rien, mais le sigle – RADAR – utilisé par la suite, vous est certainement plus familier ! Willy collabora ensuite au Manhattan District Project et consacra ses talents de physicien à une nouvelle technologie qui semblait fort prometteuse. En 1945, il était à la tête du Département Electronique et concevait les circuits temporisateurs qui allèrent, quelques années plus tard, égrener les dernières millisecondes précédant l'explosion de la première bombe atomique.
Quand la bombe explosa à Los Alamos, Willy se trouvait à une quarantaine de kilomètres de là. Aussi put-il observer les différentes phases de l'explosion. Il finit par me dire qu'il était remonté en voiture, ainsi que tous les autres observateurs, et que le trajet du retour s'était fait dans le plus grand silence. « Personne n'avait rien à dire ».
Durant les deux années qui suivirent, Willy occupa à Washington le poste de secrétaire de la fédération des Chercheurs américains et il milita activement en faveur de la non-prolifération des armes nucléaires, en qualité d'intermédiaire entre le Congrès et les chercheurs. Avec ses collègues, Willy qui est aujourd'hui doyen des Laboratoires de Brookhaven, a rassemblé la documentation la plus complète au monde sur les systèmes de protection nucléaire.
Et dire que c'est le même homme qui a inventé le jeu vidéo !

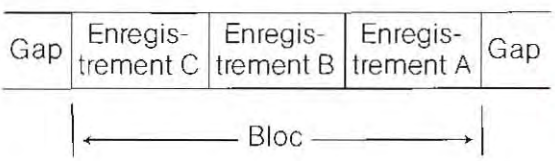
(D'après John Anderson, extrait de DATA MANAGER, avril 1983. © CREATIVE COMPUTING MAGAZINE.)

nombre variable de zones.
Les enregistrements que nous avons étudié sont des **enregistrements logiques*** ; ce sont les éléments d'un fichier, et ils contiennent une ou plusieurs informations relatives à un objet de ce fichier.
Les dispositifs de lecture et d'écriture utilisés sur les mémoires de masse permettent d'effectuer physiquement l'enregistrement d'une donnée sur des portions du support (bande, disque) limitées par deux entre-enregistrements. La partie de la mémoire concernée par cette opération est un **enregistrement physique.** Sa taille varie en fonction de celle de l'espace mémoire destiné à accueillir l'information dans l'unité centrale.
Lorsqu'on établit les enregistrements physiques, il faut à la fois réduire les temps d'accès aux informations et augmenter le plus possible le nombre des informations conservées dans la mémoire de masse, le tout, sans encombrer la mémoire centrale.
Pour en revenir aux enregistrements de notre fichier du personnel, ils peuvent se présenter de deux façons différentes sur un support magnétique (une bande, par exemple) :

Exemple 1



Exemple 2



Dans l'exemple 1, les entre-enregistrements (gaps) séparent tous les enregistrements, tandis que dans l'exemple 2, plusieurs enregistrements sont regroupés entre deux gaps pour former un bloc. Cette seconde méthode d'enregistrement permet de réduire la longueur des parties sans information et d'augmenter le nombre d'enregistrements logiques que l'on peut loger sur la bande.
En outre, le temps nécessaire aux opérations de lecture et d'écriture se trouve réduit dans la mesure où l'unité de bande livre trois enregistrements logiques en une seule fois. L'ensemble des enregistrements ainsi disposés s'appelle un bloc ou enregistrement physique, tandis que le nombre d'enregistrements contenus dans un bloc s'appelle le **facteur de blocage** (ici, trois).
On doit choisir le facteur de blocage de façon à réduire le plus possible le nombre des entre-enregistrements dans les limites consenties par l'espace mémoire disponible pour le bloc dans l'unité centrale.
Notons que les enregistrements de longueur fixe sont, en principe, regroupés suivant un même facteur de blocage et que les blocs sont donc également de longueur fixe. Lorsque les enregistrements logiques sont de longueur variable, on fixe une longueur maximale de bloc et on regroupe le plus d'enregistrements possibles par bloc, en évitant toutefois de morceler les enregistrements. Les blocs ont donc une longueur variable, inférieure ou égale à la longueur maximale prévue.
Lorsqu'il n'y a qu'un enregistrement par bloc, il y a coincidence entre enregistrements physique et logique. La longueur d'un enregistrement physique est, en règle générale, de 128 caractères sur les disquettes utilisées en micro-informatique. Dans le fichier du personnel pris comme exemple, la longueur de l'enregistrement logique est égale à la somme de la longueur des zones, soit $3 + 2 + 40 + 2 + 7 = 54$. On peut donc mettre deux enregistrements logiques par bloc ($128 : 54 = 2{,}37$).

Organisation des fichiers sur bande magnétique

Un fichier sur bande magnétique est une suite d'enregistrements logiques dont les informations peuvent se succéder sous les têtes de lecture et d'écriture. Un tel fichier se caractérise donc essentiellement par sa structure séquentielle, impliquant un mode particulier d'enregistrement. Les enregistrements peuvent être effectués dans le désordre, ou bien être rangés en ordre croissant ou en ordre décroissant. Des enregistrements écrits dans le désordre peuvent être classés à l'aide d'un programme de **tri** (en anglais, sort), généralement prévu dans le programme-système.

* Enregistrement logique a pour synonyme : article ; enregistrement physique a pour synonyme : bloc.

Les données du fichier du personnel peuvent être enregistrées selon les différentes façons suivantes :

1 / sans ordre de matricule

matr. 15 Vatin Jean
matr. 2 Arnoux Pierre
matr. 7 Durieux Robert
matr. 5 Blanchard Antoine
matr. 20 Violet François
.....

2 / en ordre croissant de matricule

matr. 2 Arnoux Pierre
.....
matr. 5 Blanchard Antoine
.....
matr. 7 Durieux Robert
.....
matr. 15 Vatin Jean
.....
matr. 20 Violet François
.....

3 / en ordre décroissant de matricule

matr. 20 Violet François
.....
matr. 15 Vatin Jean
.....
matr. 7 Durieux Robert
.....
matr. 5 Blanchard Antoine
.....
matr. 2 Arnoux Pierre

Dans cet exemple, l'ordre des matricules coïncide avec l'ordre alphabétique des noms des employés. Ce double classement n'a été adopté que pour permettre de retenir plus facilement l'exemple. En fait, il n'y a pas de classement alphabétique: seul importe l'ordre d'enregistrement sur le livre de paie.

Il va sans dire que la recherche est plus facile dans les fichiers triés en ordre croissant ou décroissant. L'organisation séquentielle des fichiers sur bande magnétique respecte, le plus souvent, l'ordre croissant.

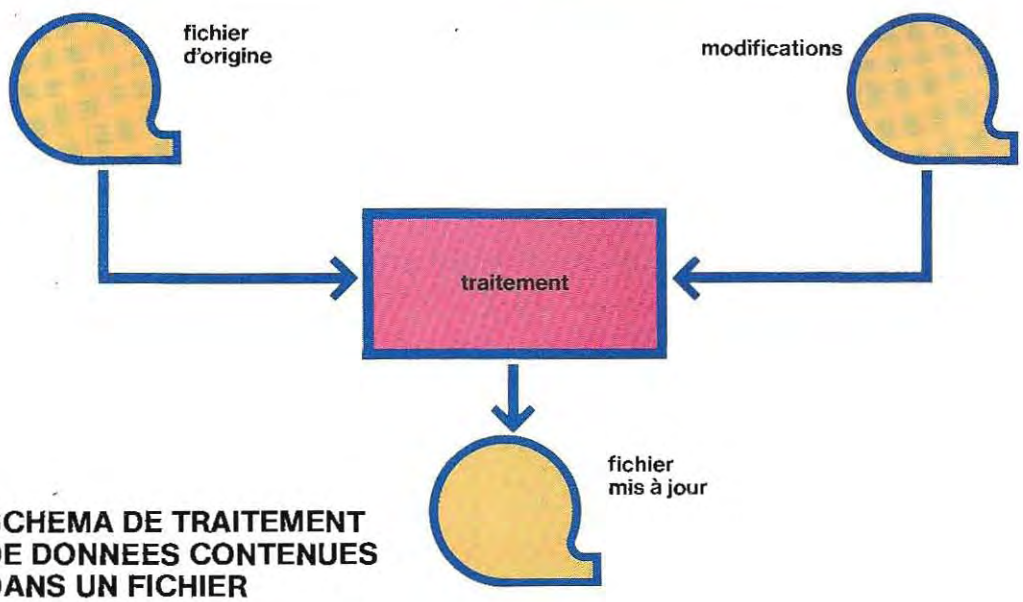
La recherche prend fin à la rencontre de l'enregistrement possédant le code demandé (enregistrement trouvé), à moins qu'un code supérieur au code recherché ne soit rencontré (enregistrement inexistant).

La mise à jour d'un fichier dépend de la méthode choisie pour l'organisation des données.

Supposons que nous devions mettre à jour le fichier du personnel trié selon l'ordre croissant de matricule, suite aux modifications survenues au cours du mois :

matr. 7 changement d'adresse
matr. 15 Vatin Jean démissionnaire
matr. 21 Violet Marie nouvelle employée

Le schéma ci-dessous illustre le traitement à effectuer.



SCHEMA DE TRAITEMENT DE DONNEES CONTENUES DANS UN FICHIER

Selon le principe de suppression physique, le fichier mis à jour contiendra les enregistrements suivants :

matr. 2 Arnoux Pierre
.....
matr. 5 Blanchard Antoine
.....
matr. 7 Durieux Robert (avec sa nouvelle adresse)
.....
matr. 20 Violet François
matr. 21 Violet Marie

Organisation des fichiers sur disque magnétique

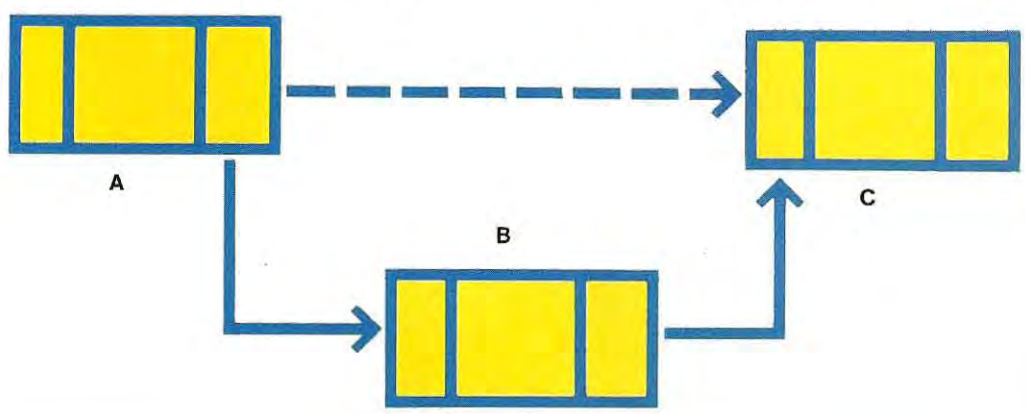
L'accès aux fichiers d'informations enregistrés sur bande est nécessairement séquentiel. En revanche, les caractéristiques physiques des disques magnétiques autorisent plusieurs modes d'accès aux informations.
On peut, en effet, structurer et relier les enregistrements afin de retrouver plus rapidement les données contenues dans les différentes zones.
Voici les quatre méthodes les plus fréquentes d'organisation d'un fichier sur disque :

- organisation séquentielle,
- organisation séquentielle avec chaînage,
- organisation séquentielle indexée,
- organisation directe.

Organisation séquentielle. Les enregistrements sont écrits l'un après l'autre suivant le critère de classement contenu dans la zone clé. Cette organisation est comparable à celle que l'on utilise sur bande magnétique : on doit, pour accéder à un enregistrement, lire tous ceux qui le précèdent.
On peut néanmoins effectuer un tri pour classer un fichier initialement écrit dans le désordre, ou, dans certaines limites, mettre à jour le fichier sans devoir le réécrire entièrement, ce qui est impossible avec une bande magnétique.
Ainsi, on peut supprimer un enregistrement en apposant une marque, ou drapeau, dans la zone appropriée. Il s'agit d'une suppression logique et non physique : l'enregistrement sera toujours présent mais il ne pourra plus être lu.
On peut également modifier un enregistrement à condition de réécrire intégralement le bloc auquel il appartient. Il est possible d'ajouter des compléments de données, grâce à des zones de dépassement situées en fin de fichier.
Cependant, en dépit de tous les avantages que nous venons de décrire et de la rapidité du support, la méthode séquentielle ne tire pas tout le parti possible des disques magnétiques.

Organisation séquentielle avec liens de chaînage. Chaque enregistrement contient, en plus des données proprement dites, des indications qui permettent de redescendre jusqu'à l'enregistrement suivant, et de former des chaînes logiques parfaitement ordonnées en fonction de la clé. Les indications de

INSERTION D'UNE DONNEE PAR LA MODIFICATION DES POINTEURS



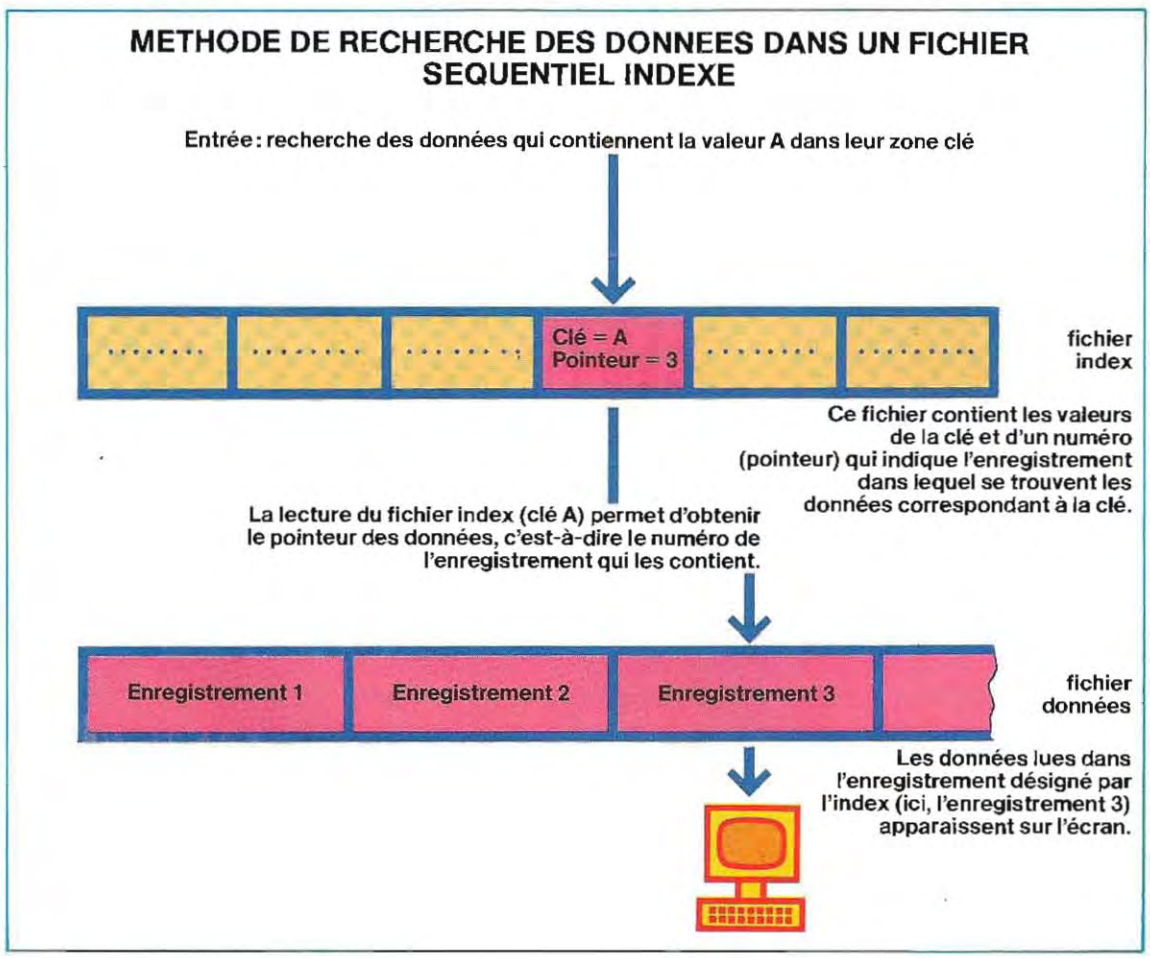
chaînage s'appellent des **pointeurs** (pointers en anglais). Les enregistrements sont alors composés de cette façon :

clé	données	pointeur

Cette méthode facilite insertions et suppressions : ces mises à jour se font par simple modification de la chaîne des pointeurs.

A / Exemple d'insertion
Pour ajouter un enregistrement B entre les enregistrements A et C, on modifie le pointeur de A pour qu'il désigne B, qu'on dote d'un pointeur vers C.

B / Exemple de suppression
Pour retirer l'enregistrement B de la chaîne A B C, il suffit de diriger le pointeur de A sur C.



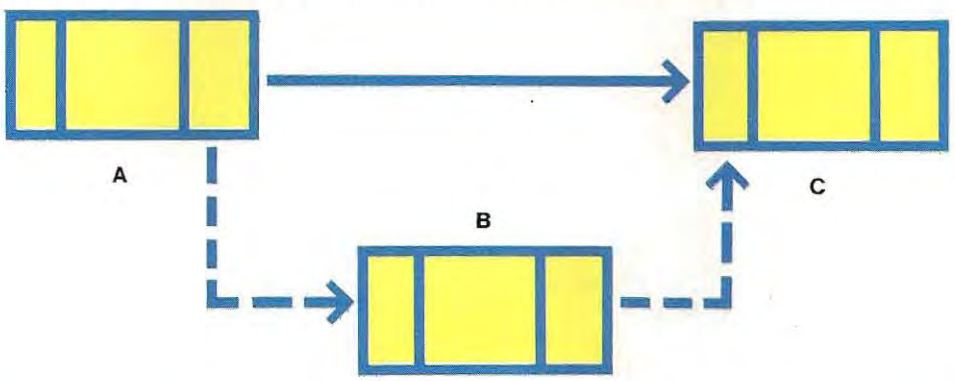
La succession physique des enregistrements reste inchangée et les nouveaux enregistrements sont insérés dans une **zone de dépassement de capacité,** périodiquement expurgée par la réécriture séquentielle du fichier.

Organisation séquentielle indexée. La zone clé est l'élément fondamental qui autorise l'introduction de ce type d'organisation dans un fichier : la valeur qu'elle prend dans chaque enregistrement permet d'identifier celui-ci sans équivoque.
La gestion du fichier se fait, là encore, de manière séquentielle. Il doit donc être classé en fonction de la clé d'accès. Mais il est cependant couplé à un autre fichier qui joue le rôle d'index.
Un fichier séquentiel indexé se compose donc de deux éléments :

- un fichier index,
- un fichier de données.

Le schéma de la page 239 illustre le principe de ce système.
C'est la table enregistrée sur le disque comme un fichier séquentiel qui constitue l'index du fichier.
Un même fichier peut être associé à plusieurs index, ce qui permet une nouvelle diminution du temps d'accès.

SUPPRESSION D'UNE DONNEE PAR LA MODIFICATION DES POINTEURS



Test 6

1 / Lesquelles de ces affirmations sur les mémoires de masse sont vraies ?
a) leur capacité est égale à celle de l'ordinateur ;
b) les bandes magnétiques sont les plus rapides ;
c) elles permettent de stocker d'énormes quantités de données.

2 / Combien un disque souple (disquette) peut-il contenir de caractères ? Donnez les ordres de grandeur habituelle minimale et maximale.

3 / Définissez : a) les fichiers ; b) les enregistrements ; c) les zones.

4 / A quoi servent habituellement les bandes magnétiques ?

5 / Remettez en ordre hiérarchique les mots suivants : octet, fichier, bit, enregistrement, ensemble de fichiers, zone.

Les solutions du test se trouvent pages 244 et 245.

Rappelons que le fichier qui contient les données doit impérativement être ordonné et que la valeur de la clé ne doit permettre d'identifier qu'un seul enregistrement. Nous aurons une meilleure idée de ce type d'organisation en l'illustrant par deux exemples classiques. On peut créer, à côté d'un fichier d'articles enregistré à partir de la piste 3 d'un disque, le fichier index suivant :

Fichier index

Pointeur de la donnée (numéro de la piste)	Clé d'accès à la donnée (dernière clé pour chaque piste)
3	1450
4	2972
5	4250
6	5587
7	6999

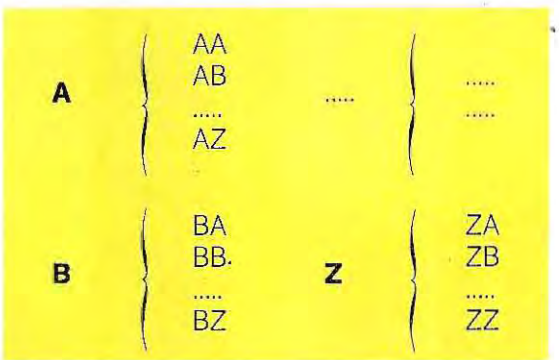
Pour accéder à une donnée de clé déterminée, on procède à une lecture séquentielle du fichier index pour retrouver la clé dans la table qu'il contient. On découvre ainsi le pointeur de la donnée, en l'occurence le numéro de la piste où se trouve l'enregistrement contenant la donnée cherchée. La lecture du fichier de données se limite aux enregistrements contenus dans la piste indiquée par le fichier index. Ainsi, la recherche de l'article dont la clé est 5350 ne nécessitera que la lecture des enregistrements du fichier de données présents sur la piste 6.

Considérons maintenant la gestion d'un fichier dont les éléments sont les mots d'un dictionnaire. Imaginons que ce fichier est divisé en 26 sous-fichiers, correspondant chacun à une des lettres de l'alphabet. Le premier niveau d'index alors créé est donc une table de 26 lettres. Si nous divisons ensuite chacun des groupes de mots ainsi formés en sous-fichiers, en fonction de la seconde lettre de chaque mot, nous créons 26 nouvelles tables, toujours composées de lettres de l'alphabet (second niveau d'index).

En somme, on fait correspondre à chaque élément du premier index l'adresse de la seconde table où sont répertoriées les

Opératrice au travail.

C. O'Rear/West Light-Grazia Neri



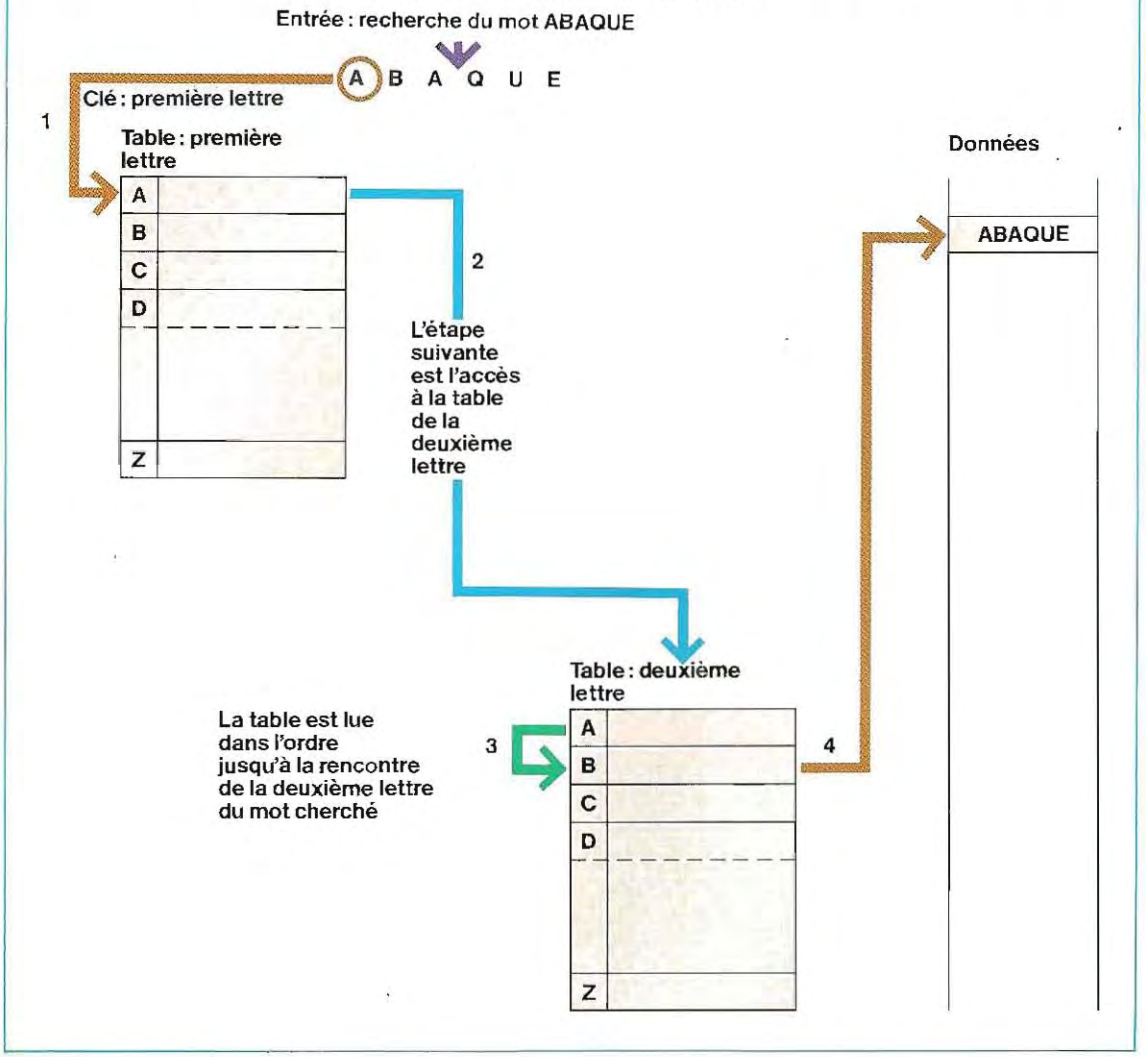
deuxièmes lettres des mots. Plus précisément, on obtient la relation ci-dessus :

L'adressage des sous-fichiers par l'index de second niveau permet de lire le fichier pour y trouver un mot déterminé. La recherche se compose des opérations suivantes :

a) accès à l'index de premier niveau et lecture séquentielle de ses éléments jusqu'à la rencontre de la première lettre du mot recherché : le pointeur renvoie à l'index de second niveau ;

b) accès à l'index de second niveau, c'est-à-dire à la table correspondante aux deuxièmes lettres et lecture séquentielle jusqu'à la rencontre de la deuxième lettre du mot recherché : le pointeur renvoie, cette fois, au sous-fichier du fichier dictionnaire ;

SCHEMA LOGIQUE DE LA RECHERCHE DANS UN FICHIER DICTIONNAIRE A DEUX NIVEAUX DE CLES



c) accès au sous-fichier du dictionnaire et lecture séquentielle jusqu'à la rencontre du mot recherché.

Organisation directe. Une fois encore, le fichier apparaît comme une structure séquentielle organisée pour contenir les enregistrements à des emplacements prédéterminés. L'adresse physique d'un enregistrement à chercher ou à écrire sera déterminée à partir de sa clé logique, à l'aide d'un algorithme de calcul, appelé **formule de conversion** (ou de randomisation).
Le fichier doit pouvoir contenir tous les enregistrements dont l'adresse sera déterminée par ce calcul et doit donc offrir plus de place

que n'en occupent les enregistrements proprements dits.
Ce gaspillage de place sur le disque est le principal inconvénient de cette organisation.

En résumé :

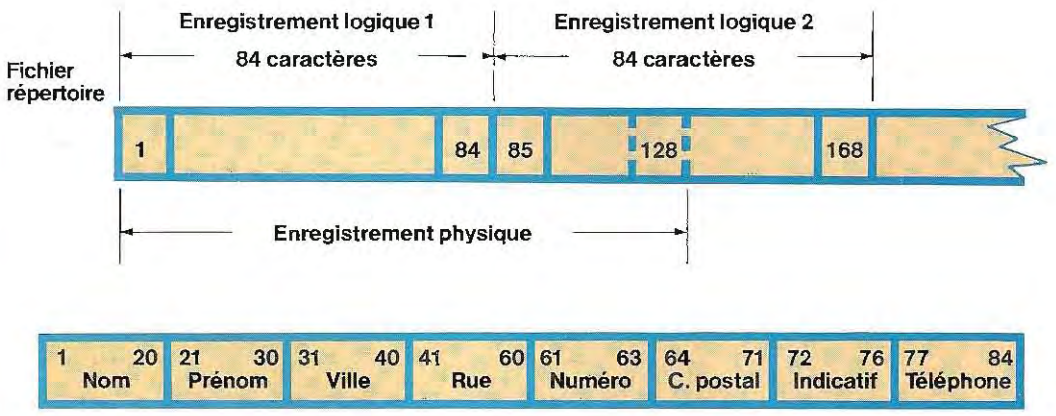
- **Les supports magnétiques de stockage des données sont principalement : les bandes, les tambours, les disques durs (fixes ou amovibles) et les disques souples ou disquettes.**
- **Un ensemble cohérent de données ayant un objet commun s'appelle un fichier.**
- **Les fichiers sont divisés en enregistrements physiques, pour les opérations de lecture et d'écriture, et en enregistrements logiques pour la programmation.**
- **Un enregistrement logique peut remplir plusieurs enregistrements physiques ou, inversement, un enregistrement physique peut contenir plusieurs enregistrements logiques, selon la longueur des zones de données.**
- **Les enregistrements logiques sont divisés en zones contenant chacune une information.**

Nous allons appliquer ces notions à l'exemple d'un répertoire téléphonique.
Les données à enregistrer sont les suivantes :

1 / Nom	20 caractères
2 / Prénom	10 caractères
3 / Ville	10 caractères
4 / Rue	20 caractères
5 / Numéro	3 caractères
6 / Code postal	8 caractères
7 / Indicatif régional	5 caractères
8 / Numéro de téléphone	8 caractères

Chaque donnée occupe une zone à laquelle a été attribuée une longueur maximale définie par un nombre déterminé de caractères. On n'aura à stocker que les données énoncées, et il suffira pour cela d'un seul fichier. Il existe des bases de données plus complexes composées de plusieurs fichiers dont chacun contient des données de structure différente. L'enregistrement logique regroupe la somme des caractères réservés pour les 8 données (zones) et a donc, ici, une longueur de 84 caractères.
Le schéma ci-dessous illustre la structure d'un tel fichier. La longueur de l'enregistrement physique dépend de la machine utilisée car elle est imposée par le matériel.
Sur la figure, on a adopté une longueur de 128 caractères car c'est la plus courante sur les micro-ordinateurs. Comme on le voit, un enregistrement physique peut contenir plus d'un enregistrement logique.

STRUCTURE DU FICHIER REPERTOIRE



Position des zones dans l'enregistrement logique

Solutions du test 6

1 / c) Les mémoires de masse sont permanentes et peuvent conserver de grandes quantités de données pendant une durée pratiquement illimitée.
La mémoire de l'ordinateur conserve les données pendant la durée de leur traitement. Les données de sortie (traitées) sont ensuite de nouveau transférées vers les mémoires de masse, soit pour y être conservées,* soit pour libérer la mémoire de traitement en vue de l'accomplissement d'une nouvelle tâche.**
Sur les ordinateurs qui travaillent en multiprogrammation, on met souvent les résultats intermédiaires « de côté » sur des disques, mémoires de masse à accès direct les plus rapides. Lors d'une pause dans l'exécution d'un programme, les données correspondantes sont transférées sur un disque pour laisser la place à celles d'un autre programme.
Le programme interrompu est repris dès que l'événement qui a provoqué son interruption disparaît, et que les ressources nécessaires à son exploitation sont disponibles. Ce mouvement des données et des programmes de la mémoire centrale aux mémoires de masse est géré automatiquement par le système d'exploitation ; il permet d'utiliser au mieux les ressources de l'unité de traitement.

---

2 / 80 000 pour les plus petits, 1 200 000 pour les plus grands. Une page, entièrement écrite, de cette encyclopédie, contient en moyenne deux colonnes de 50 lignes chacune. Une ligne se compose de 45 caractères environ. Une page remplie contient donc, à peu près, $2 \times 50 \times 45 = 4500$ caractères. En fait, une page moyenne contient 3 000 caractères environ car les photographies et les schémas occupent un tiers de la surface totale. Un tascicule comporte 24 pages, ce qui donne $24 \times 3000 = 72000$ caractères. Une disquette de 1 200 000 caractères pourrait donc abriter 16 fascicules et l'ouvrage entier pourrait tenir sur 5 disques de quelques grammes qui ne prendraient pas plus de place que quelques teuilles de papier.

---

3 / a) Les fichiers sont des parties des mémoires de masse qui regroupent sous un même nom (le nom du tichier) des données homogènes.
b) Les enregistrements peuvent être physiques ou logiques. Les enregistrements physiques constituent la structure physique du fichier, les enregistrements logiques définissent sa structure logique.
c) Les zones sont les emplacements qui contiennent les données dont sont constitués les enregistrements logiques.

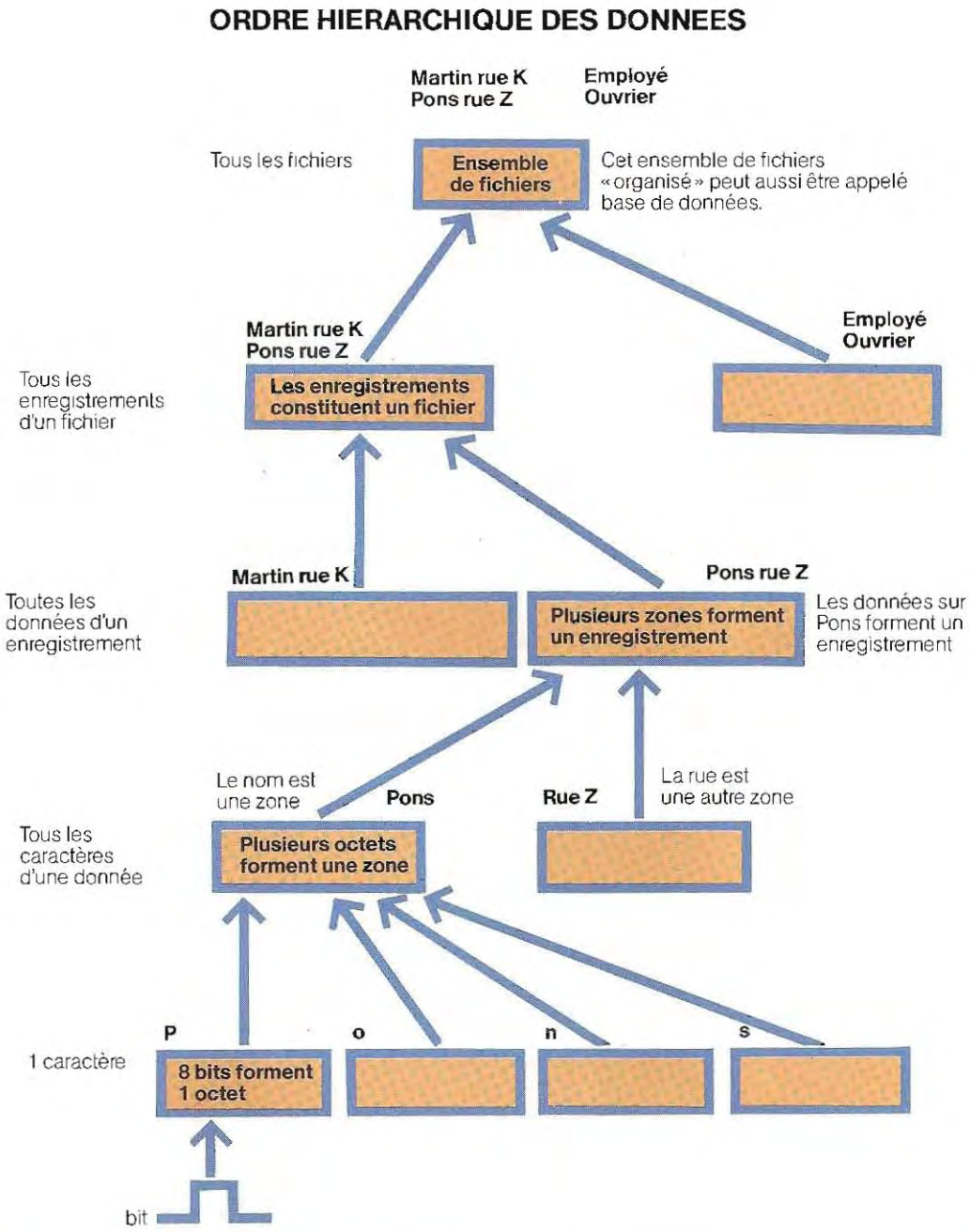
---

4 / On les utilise principalement pour sauvegarder les fichiers sur disque. En effet, leur accès est uniquement séquentiel, ce qui n'est pas pratique pour la manipulation fréquente des fichiers. Les disques, en revanche, sont à accès direct et permettent le transfert rapide des données.

---

* Fichiers permanents.
** Fichiers temporaires.

5 / Bit, octet, zone, enregistrement, fichier, ensemble de fichiers. Le schéma ci-dessous illustre la hiérarchie de ces éléments.



Lorsqu'on voudra lire, par exemple, l'enregistrement logique 1, une partie de l'enregistrement 2 sera donc également transférée en mémoire. C'est le **système d'exploitation** (c'est-à-dire l'ensemble des programmes qui régissent les fonctions de base du système) qui se chargera alors de séparer les données utiles des données inutiles.

Le système d'exploitation est, en général, fourni avec la machine. C'est un ensemble de programmes de base indispensables au fonctionnement de l'ordinateur, et notamment à la gestion des entrées et des sorties sur disque, imprimante, clavier ou écran. L'utilisateur peut ainsi programmer même s'il ne sait pas comment est construite sa machine. C'est le système d'exploitation qui traduit les instructions rédigées en langage évolué (comme le Basic) en instructions en langage machine (micro-instructions).

L'organisation des fichiers sur les micro-ordinateurs

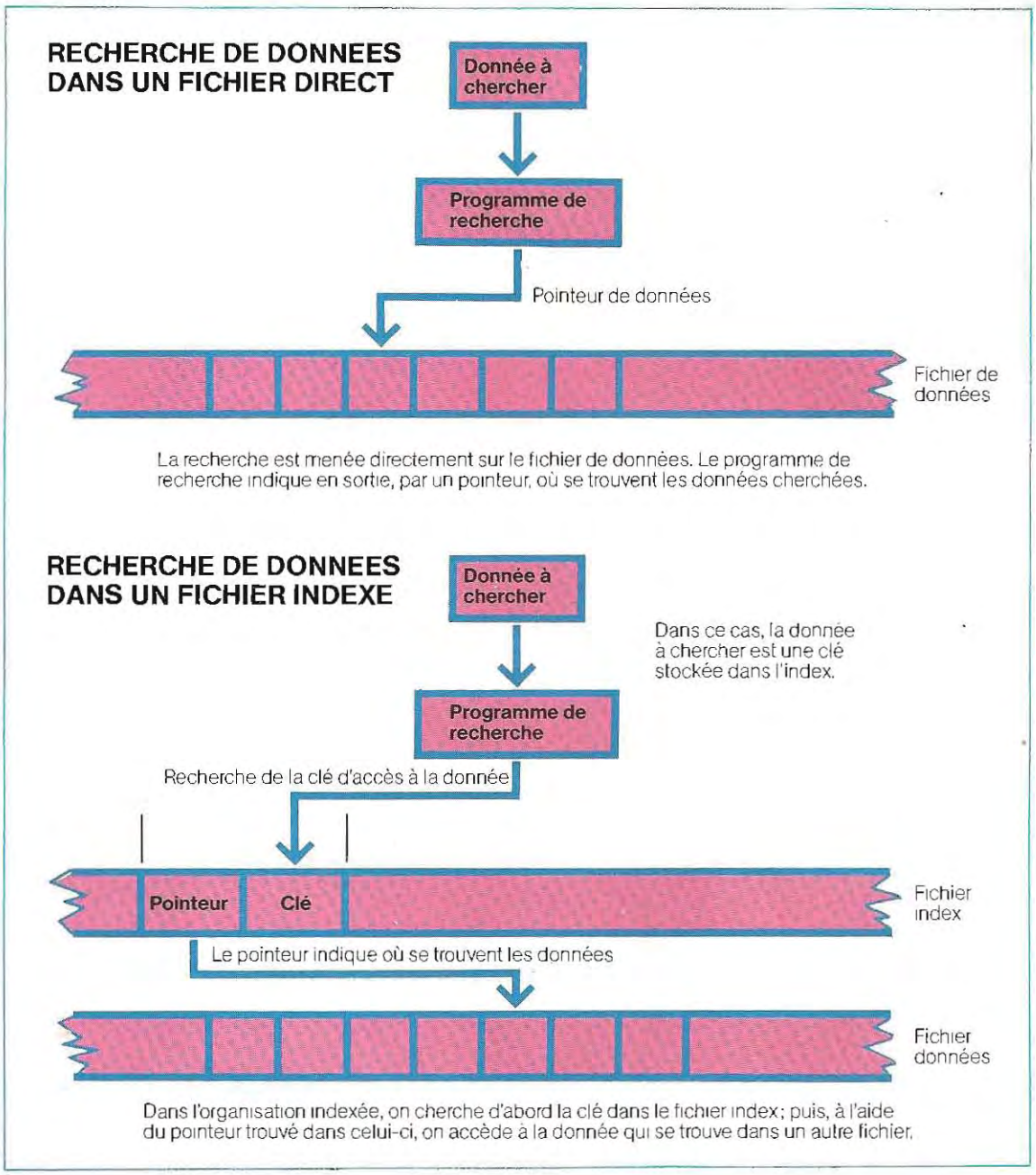
Sur les gros systèmes et sur les mini-ordinateurs les plus perfectionnés, il existe des programmes de gestion de fichiers tout prêts qui épargnent au programmeur les détails de la recherche ou de la mise en ordre des données. Ces programmes font le lien entre le programme d'application écrit par le programmeur et la partie du système assurant la gestion des fichiers. Il suffit d'indiquer, par des instructions, l'opération (une recherche par exemple), que l'on désire effectuer. Les programmes de gestion interpréteront alors l'instruction pour mettre en œuvre le traitement correspondant.

On dispose moins fréquemment de ce type de logiciel sur les petits systèmes (micro-ordinateurs et ordinateurs personnels). D'ailleurs, leur prix est élevé par rapport à celui de la machine. De plus, les programmes de gestion interne doivent être généralisés de façon à couvrir toutes les applications possibles. Leur extension entraîne l'occupation d'un plus grand espace sur les disques comme dans la mémoire et ceci est un gros inconvénient sur les petits systèmes où l'espace mémoire fait justement défaut.

Pour certaines applications, il faut écrire, en plus

Terminaux en phase de contrôle après fabrication.

C. O'Rear/West Light-Grazia Neri



du programme proprement dit, les instructions nécessaires à la recherche et à la mise en ordre des données, sans généraliser mais, au contraire, en trouvant une solution spécifique.
Notons l'existence de logiciels de gestion de fichiers qui, vendus séparément de l'ordinateur, permettent la création de fichiers, l'enregistrement, la consultation, la modification ou la suppression des données.
Les deux principales méthodes de recherche sont la **recherche logique** et la **recherche dichotomique**. On peut les appliquer soit aux fichiers index, soit directement aux fichiers données. Dans le premier cas, on doit, après avoir trouvé la clé dans le fichier index, se servir du pointeur pour localiser l'enregistrement dans le fichier données.
Le schéma ci-dessus illustre les méthodes de recherche sur fichier direct et sur fichier séquentiel indexé.
Précisons tout d'abord quels sont les principaux types de fichiers utilisés sur les micro-

ordinateurs. On distingue :

- **les fichiers directs (ou aléatoires),**
- **les fichiers séquentiels,**
- **les fichiers indexés.**

Les fichiers directs (ou aléatoires)

Dans ce type de fichier, l'adresse d'une donnée correspond au numéro de l'enregistrement qui la contient : les données du groupe 1 se trouvent dans l'enregistrement 1, celles du groupe 2 dans l'enregistrement 2, et ainsi de suite.

L'accès direct (DAM, sigle dérivé de l'anglais Direct Access Method) s'appelle aussi l'**accès aléatoire** ou **sélectif**. Il permet d'écrire ou de lire un enregistrement en indiquant sa position physique sur le disque, c'est-à-dire son adresse absolue, ou son adresse relative à l'intérieur du fichier.

On peut donc lire directement un enregistrement si l'on connaît son numéro (position physique) ou sa position relative par rapport à d'autres enregistrements.

Dans l'adressage relatif, le numéro de l'enregistrement lui-même peut être utilisé comme une clé. C'est au programme d'application de déterminer à quel endroit du fichier se trouve la donnée (numéro de l'enregistrement).

Dans l'organisation d'un répertoire téléphonique, on peut attribuer à chaque enregistrement le numéro correspondant à l'ordre dans lequel les données ont été saisies : la première adresse écrite se trouve dans l'enregistrement 1, et ainsi de suite. Pour retrouver une donnée dans le fichier, il suffira de trouver le numéro de son enregistrement.

Comme les données ont été enregistrées en ordre chronologique, il faudra, à chaque recherche, lire tout le fichier. Si, par exemple, nous voulons les numéros de téléphone de Paris, nous devrons lire tout le fichier car nous ignorons où ces numéros commencent et où ils finissent.

On peut pallier cet inconvénient, qui rallonge beaucoup la durée du traitement, en classant le fichier des données selon l'ordre alphabétique des noms de villes, avant de lancer la recherche sur Paris. Ainsi, dès que la donnée rencontrée (Pau, par exemple) aura une valeur supérieure à celle de Paris, on pourra abandonner la recherche, la ville Paris ne pouvant pas être contenue dans les enregistrements suivants. Cette procédure logique est illustrée schématiquement par l'organigramme de la page 249.

L'accès direct aux enregistrements est le principal avantage de l'organisation des fichiers directs. On a toute liberté pour lire ou écrire un enregistrement. On peut, par exemple, lire l'enregistrement 100 avant l'enregistrement 50. C'est pour cela qu'on parle de fichiers à accès aléatoire (random access, en anglais).

Les fichiers séquentiels

Ce sont des fichiers classés séquentiellement par numéro d'enregistrement et dans lesquels il n'est pas possible d'accéder à un enregistrement, en lecture ou en écriture, avant d'avoir lu ou écrit tous ceux qui précèdent.

On les utilise pour stocker des données en nombre déterminé, qu'on doit toutes lire ou écrire en même temps. C'est pourquoi les tables sont souvent contenues dans des fichiers séquentiels. Les enregistrements sont lus ou écrits dans l'ordre physique, et il n'est possible de faire une insertion dans ce type de fichier qu'en le récrivant tout entier, à moins, bien sûr, que l'ajout ne soit à placer à la fin du fichier.

En principe, les mises à jour sont donc impossibles ; cependant, certains systèmes permettent des modifications.

Les fichiers indexés

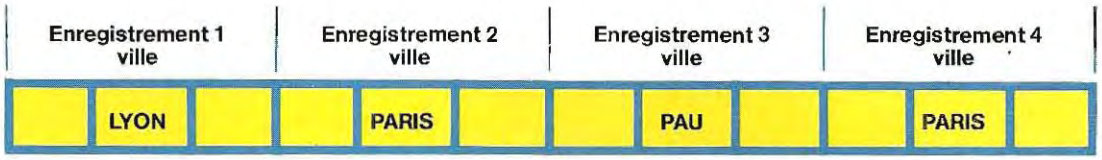
Ce sont des fichiers directs couplés à un index, lequel se trouve dans un autre fichier, ou bien dans le fichier de données lui-même. Ils servent pour de nombreuses applications et nous étudierons donc plus en détail leur fonctionnement.

Un fichier indexé comporte deux structures : le fichier index et le fichier données (voir schéma p. 250). Le fichier index est une suite d'enregistrements qui contiennent les clés des enregistrements de données et des pointeurs. Une clé correspond à un enregistrement et un seul. Parfois, des index secondaires permettent d'accéder aux enregistrements par des voies différentes.

Par ailleurs, la même clé peut apparaître plusieurs fois dans le fichier index. Si l'on a pris le nom de famille comme clé, on risque de rencontrer le cas de plusieurs personnes qui portent le même nom. Il faudra alors préciser le numéro

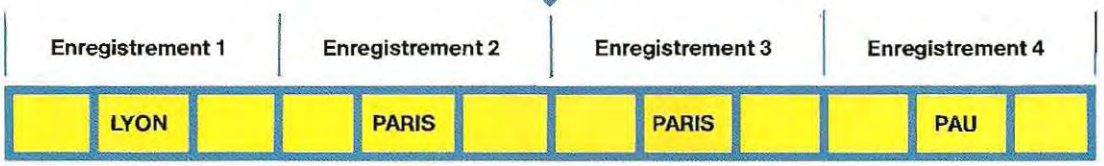
ORGANIGRAMME DU TRAITEMENT PERMETTANT D'INTERROMPRE UNE RECHERCHE DANS UN FICHIER ORDONNE

Fichier d'origine



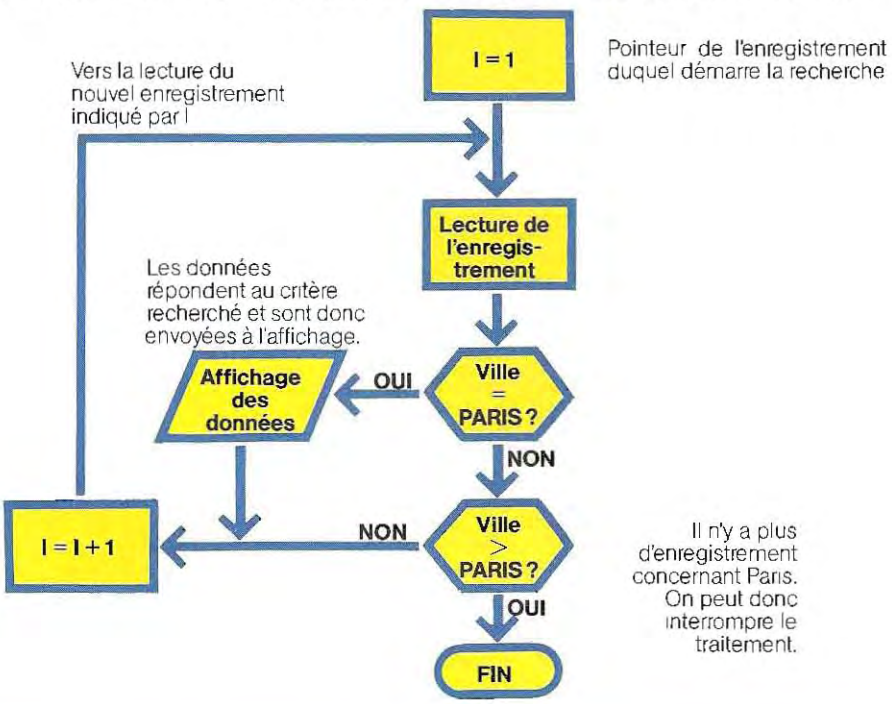
Dans le fichier d'origine, les données sont placées suivant l'ordre de leur introduction. Pour faire une recherche par ville, il faut donc lire le fichier dans sa totalité.

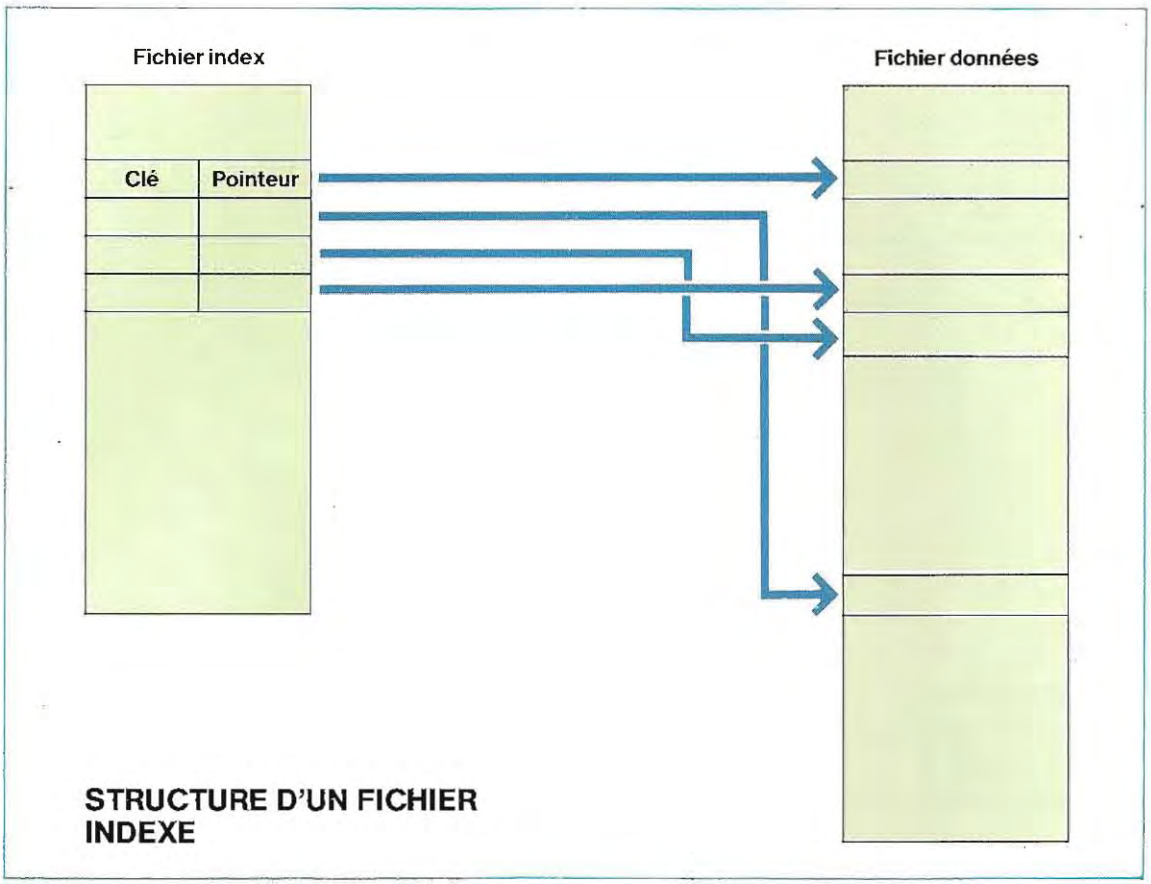
(c'est l'ordre alphabétique qui est retenu)



Le nouveau fichier est ordonné

On peut maintenant interrompre la recherche à l'enregistrement 4 puisque la donnée qu'il contient dans sa zone ville (Pau) est supérieure à la donnée choisie (Paris). En effet, le fichier étant classé dans l'ordre alphabétique des noms de villes, il ne peut pas y avoir d'enregistrements concernant Paris au-delà.





STRUCTURE D'UN FICHIER INDEXE

d'ordre de chaque enregistrement par rapport aux enregistrements ayant une même clé.

Le fichier données contient les enregistrements où se trouvent les données. Il peut éventuellement être organisé en fonction de la valeur de la clé, selon la méthode que l'on désire utiliser.

Les index secondaires permettent d'accéder aux enregistrements par d'autres clés que la clé principale (primaire). Ainsi, on peut lire les enregistrements d'un fichier clients en indiquant soit le numéro de code, soit le nom du client. L'index primaire utilisera le numéro comme clé, tandis que l'index secondaire prendra le nom comme clé secondaire. On aura ainsi deux moyens indépendants d'accès au fichier (c'est ce qu'illustre le schéma page 251).

Il est parfois nécessaire d'utiliser une clé composite, regroupant plusieurs valeurs. Pour cela, on combine les données afin qu'elles forment un tout : c'est la **concaténation**.

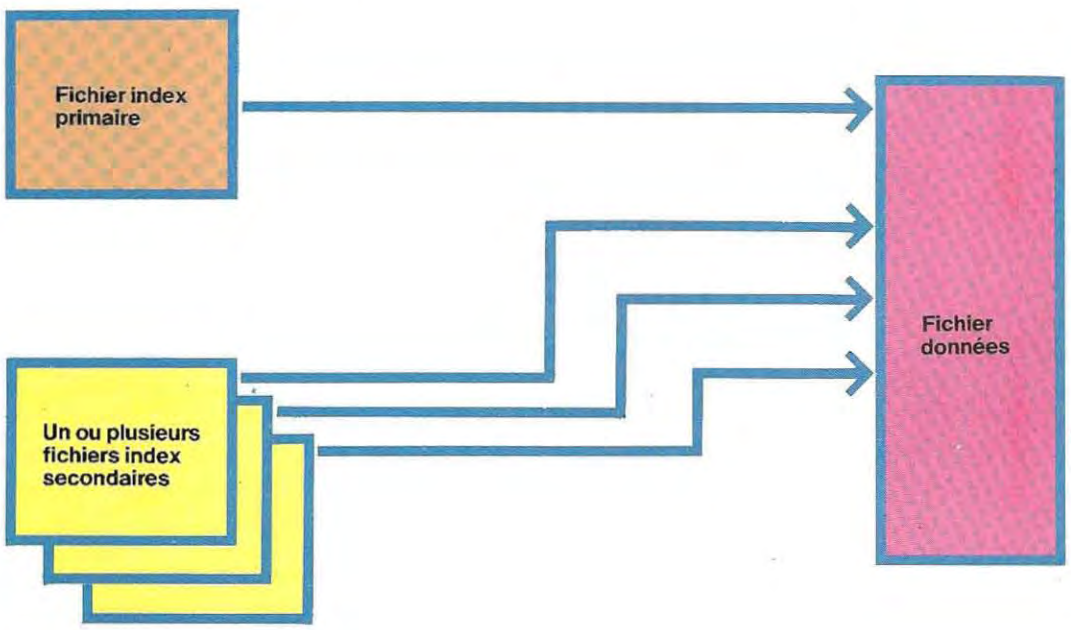
Une société peut, par exemple, souhaiter cer-

ner facilement l'activité de chacune de ses succursales. Si la clé des enregistrements se compose du code de la succursale et du numéro du client, les clients d'une même succursale seront regroupés et il sera possible de traiter les données les concernant indépendamment de celles des autres clients. Toutefois, un traitement global du fichier sera toujours possible.

Les fichiers indexés offrent la plus grande souplesse de manipulation, et permettent d'effectuer la recherche d'une donnée déterminée dans les meilleures conditions, mais ce sont néanmoins les plus difficiles à gérer. Les programmes de gestion des fichiers indexés présentent en effet une certaine complexité et s'adressent donc à des programmeurs déjà expérimentés.

Cela ne diminue en rien l'intérêt de l'organisation indexée, et nous étudierons, dans le chapitre suivant, la méthode ISAM de gestion des fichiers indexés, méthode utilisable sur les micro-ordinateurs et les ordinateurs individuels.

STRUCTURE D'UN FICHIER A INDEX SECONDAIRES



L'informatique au bureau

Le bureau est aujourd'hui encore le centre vital d'une entreprise industrielle ou commerciale. La diminution progressive, depuis les années cinquante, de la main-d'œuvre employée par les industries manufacturières, l'élargissement du secteur tertiaire et l'avènement de l'électronique ont entraîné la création de nombreux emplois de bureau. Aux Etats-Unis, par exemple, le pourcentage des personnes employées dans le secteur du traitement de l'information est aujourd'hui supérieur à celui des autres branches d'activité.

L'introduction du traitement électronique des données a permis la première phase d'une évolution qui aboutira peut-être à la disparition du papier comme support de l'information. Elle débouche déjà sur d'intéressantes innovations en matière de représentation graphique et de traitement de texte.

L'ordinateur a également supprimé les travaux ennuyeux de manipulation des chiffres. Les comptables et les directeurs des ventes mettent désormais en œuvre des méthodes de traitement des données et d'étude statistique autrefois inimaginables.

On est forcé de constater que le progrès dans les habitudes de travail passe souvent par l'automatisation du calcul et de l'écriture. Le prix d'un ordinateur personnel est d'ailleurs de plus en plus modéré. Certes, sa rapidité, la capacité de sa mémoire et la portée de ses analyses ne peuvent être comparées à celles de ses frères aînés ; néanmoins, il est tout à fait adapté à de nombreuses tâches courantes, telles que la tenue des livres de comptes dans un service de comptabilité.

L'utilisation systématique d'ordinateurs dans les bureaux entraîne très souvent une restructuration des locaux eux-mêmes. Les « postes de travail » sont de petites unités destinées à une ou deux personnes au sein d'un vaste bureau sans cloisons fixes. Chaque unité comprend le matériel et les instruments nécessaires, et notamment un écran, un clavier et une interface de connexion à la mémoire de l'ordinateur et au système de télécommunications de l'entreprise.

L'espace réservé à chaque poste de travail est conçu et aménagé pour assurer le calme indispensable à de bonnes conditions de travail.

Des études ergonomiques approfondies ont été menées (souvent à l'aide de l'ordinateur), et ont permis de souligner l'importance de certains facteurs inhérents à l'utilisation de ce matériel, telles l'intensité de l'éclairage, la distance entre les postes de travail ou la chaleur produite par les machines. Quant au bruit, si la plupart des ordinateurs fonctionnent en silence, certaines parties de la machine émettent toutefois des ultrasons qui, en général, ne sont pas perceptibles par l'oreille humaine, mais peuvent néanmoins provoquer des troubles. Tous ces aspects doivent être étudiés attentivement.

Le traitement automatique des données s'étend aux textes rédigés, et il ne faudra sans doute plus très longtemps pour que la simple machine à écrire soit troquée contre le clavier d'un ordinateur capable d'effectuer toutes les opérations ultérieures de correction et de mise en page. D'ailleurs, prévoyant cette évolution, certains fabricants de machines à écrire électroniques ont prévu l'adjonction à la machine du matériel nécessaire à la mémorisation du texte sur une disquette.

Grâce à l'ordinateur, un message peut être codifié et transmis à quelqu'un par l'intermédiaire du clavier, sur lequel on compose les indications de priorité et de confidentialité du texte, ainsi que l'adresse électronique du destinataire. Une fois arrivé, le message est inséré parmi les autres données en instance et transféré dans la mémoire selon son degré de priorité. Le destinataire sélectionne les messages reçus à l'aide d'un clavier spécial et les lit sur son écran.

De cette manière, aucune feuille de papier ne circule dans le bureau, aucun document imprimé n'a besoin d'être jeté ou classé, et aucun intermédiaire n'est plus nécessaire au transport du message. Il s'agit d'un véritable système de courrier électronique. Ce qui n'empêche pas, d'ailleurs, de déclencher l'impression de documents sur papier à la sortie de l'ordinateur dans les cas où cela est nécessaire.

Mais la plupart du temps on renonce à l'utilisation du papier, et on peut penser que cette tendance se généralisera complètement quand on disposera d'un réseau de télécommunications doté des capacités de mémorisation adéquates. De plus, l'emploi des liaisons par fibres optiques porteuses d'impulsions laser codifiées, ou par satellites de télécommunications, devrait multiplier le nombre d'applications de ces systèmes.

Il va sans dire qu'il faudra prévoir de quoi remplacer l'autorité de la signature, une fois que l'usage du document imprimé sera sorti des habitudes. Il faudra également mettre en place des moyens de prévention efficaces contre l'introduction dans les ordinateurs d'erreurs humaines qui pourraient entraîner des conséquences désastreuses.

Quant aux problèmes spécifiques posés par l'ampleur des télécommunications, par l'archivage des données et par les opérations multilingues, ils seront sans aucun doute résolus bien avant celui de la restructuration des organigrammes de l'organisation du travail de certaines catégories du personnel. Archivistes, dactylographes, employés de bureau, entre autres, devront se recycler en vue d'autres activités s'ils ne veulent pas se trouver privés d'emploi, la machine effectuant désormais leur tâche et de façon plus rapide.

A la faveur de cette adaptation à une nouvelle réalité du travail de bureau, on constate fréquemment que de nouveaux rapports s'établissent entre les salariés d'une même entreprise, les différences de niveau culturel ayant plutôt tendance à s'estomper.

Nombre de sociétés sont déjà équipées d'ordinateurs pour le travail de bureau quotidien. Les maisons d'édition scientifique, en particulier, ont mis en place des centres de traitement destinés à supprimer les pertes de temps (et d'argent) dues au travail de documentation, de consultation des sources d'information et de préparation des rapports nécessaires au travail des rédacteurs.

Désormais, l'auteur transmet son manuscrit sur une bande magnétique et l'adresse à son éditeur par téléphone. Le texte est alors introduit dans une machine de traitement de texte et enregistré dans un fichier. Toute personne intéressée (et autorisée) par l'étude concernée peut ensuite « appeler » ce texte et le voir s'afficher sur l'écran de son poste de travail.

Ce procédé est déjà adopté dans de nombreuses bibliothèques universitaires pour l'archivage sur bandes d'articles de revues scientifiques et techniques.

Les salles de rédaction des journaux ont également connu des changements radicaux dus à l'utilisation de l'ordinateur. Journalistes, rédacteurs et compositeurs disposent de consoles de visualisation reliées à un ordinateur central, et peuvent composer un journal

Brecht Einzig Ltd

Lyndsay Westhead Architects/Environement

L'emploi de l'ordinateur dans les bureaux entraîne souvent une restructuration des locaux eux-mêmes.

entier (titres, textes et illustrations) en opérant uniquement en liaison avec la mémoire du système.

L'image électronique du journal terminé est conservé sur une disquette, puis ensuite transférée à de la pellicule photo qui permet de réaliser les films nécessaires au fonctionnement de la machine à imprimer.

En fait, cette dernière opération est utile parce que l'opération finale est encore l'impression sur du papier, mais on peut très bien envisager un autre mode de diffusion des informations. Des écrans installés dans les moyens de transport en commun, ou même un certain nombre de nos actuels postes de télévision disposés dans les grands magasins, dans les restaurants, les cafés, les salons de coiffure, les laveries automatiques, entre autres, pourraient très bien remplacer les journaux pour la transmission des informations. C'est déjà un peu ce qui se passe mais à une moindre échelle avec les journaux télévisés, et il suffirait de développer le système.

Méthode ISAM de gestion d'un fichier indexé

Cette méthode (Indexed Sequential Access Method) de gestion des fichiers indexés permet d'accéder aux enregistrements aussi bien séquentiellement que directement au moyen d'une clé (ou donnée d'identification), associée à chaque enregistrement.

Selon le type d'application, on aura avantage à utiliser l'un ou l'autre des ces deux modes d'accès. Dans les deux cas, la structure du fichier à clé se révèlera extrêmement utile. Elle est, en effet, beaucoup plus souple que celles qui dérivent des autres formes d'organisation, et se révèle particulièrement adaptée aux applications complexes en temps réel et en mode conversationnel.

Les fichiers indexés sont un outil très commode, notamment pour la lecture des informations contenues dans un enregistrement par simple communication de la clé.

Supposons, par exemple, que dans le cadre de la gestion d'un magasin, on veuille connaître la quantité en stock d'un article déterminé, afin de procéder à une commande éventuelle.

Il suffira d'utiliser un fichier indexé auquel on donnera pour clé le numéro de code de l'article, et de prévoir un programme qui demande d'abord à l'opérateur le code de l'article qui l'intéresse, et qui aille ensuite lire l'enregistrement correspondant, pour enfin afficher sur l'écran la quantité en stock.

L'utilisation d'index secondaires permettra d'attribuer aux enregistrements une seconde clé, qui sera, par exemple, la désignation de l'article. On pourra ainsi, même si on ne connaît pas le numéro de code d'un article, accéder à l'enregistrement qui le concerne.

Un autre avantage du fichier indexé est qu'il permet la vérification des données introduites, dans la mesure où une donnée peut être utilisée comme clé. Ainsi, dans un programme de paie, l'enregistrement où se trouve le nombre d'heures de travail hebdomadaires d'un employé contient également son numéro de matricule.

L'opérateur tape donc le numéro de matricule qui sert de clé d'accès aux informations

L'utilisation de l'ordinateur n'a pas seulement bouleversé la notion traditionnelle d'archive. Elle a aussi apporté de nouvelles méthodes de travail permettant des recherches très rapides.

C. O'Rear/West Light-Grazia Neri

concernant un employé. Si la valeur entrée ne correspond à aucune clé, l'opérateur en est averti par la machine, et peut introduire le matricule exact.

Les opérations d'accès aux données

Cinq méthodes de lecture différentes permettent d'accéder à un enregistrement dans un fichier indexé.

- **Lecture directe par clé.** On doit indiquer une valeur de clé afin d'obtenir le pointeur de l'enregistrement correspondant. Dès que le système a eu connaissance de la clé, il recherche, dans le fichier d'index, le pointeur correspondant à l'enregistrement. En cas de recherche infructueuse, un message d'erreur est émis.
  Cette méthode de lecture directe est utile pour la consultation et la mise à jour des fichiers lorsqu'on veut traiter des enregistrements déterminés dont on connaît la clé.

- **Lecture générale par clé.** Cette méthode est semblable à la précédente à cette différence près que, si l'enregistrement demandé n'existe pas, c'est le pointeur correspondant à l'enregistrement de clé immédiatement supérieure qui sera retenu.
  On peut ainsi opérer des regroupements logiques entre les enregistrements au moyen de clés partielles. C'est un moyen très efficace de sélection des enregistrements de données.

- **Lecture séquentielle par clé.** Cette méthode permet de lire des enregistrements en ordre logique, c'est-à-dire dans l'ordre de leurs clés, indépendamment de leur implantation physique dans le fichier.
  La lecture peut partir de n'importe quel enregistrement logique, selon l'ordre désiré.
  Combinée à la précédente, cette méthode offre la possibilité de lire des groupes cohérents d'enregistrements.

- **Lecture séquentielle en ordre physique.** Elle consiste à lire les enregistrements du fichier de données dans l'ordre physique où ils se présentent, sans se préoccuper de l'index. Cette méthode présente un intérêt lorsque seule compte, pour le traitement, l'analyse des données, et non leur ordre de lecture, car les temps d'accès sont alors réduits dans la mesure où l'on supprime l'étape de recherche dans l'index.

- **Lecture directe par numéro d'enregistrement.** Elle permet d'accéder aux données par leur numéro d'enregistrement. Il est donc possible de consulter un fichier de données aussi bien en accès direct à partir de la clé qu'en mode DAM.

Il existe également plusieurs façons de pratiquer des insertions et des suppressions de données.
Le système de gestion des fichiers indexés offre la possibilité de supprimer soit les enregistrements de données, soit leur clé, soit encore les deux. Le travail de programmation s'en trouve simplifié, on gagne de la place sur le disque, et la réorganisation périodique des fichiers perd de son urgence.
Il existe une instruction spéciale pour éliminer d'un fichier des enregistrements déterminés. Cette fonction supprime la clé des enregistrements dans le fichier index, supprimant aussi les enregistrements correspondants. Mais si l'on veut supprimer la clé d'un enregistrement tout en conservant celui-ci, on doit utiliser la fonction de suppression de clé.
La clé disparaît alors de l'index, tandis que l'enregistrement qui contient les données reste intact.
Cependant, il n'est plus accessible par sa clé mais devra être lu séquentiellement.
C'est donc le programmeur qui est maître de la suppression des données.
La suppression de clé est particulièrement intéressante dans les applications où l'on prévoit de consulter les enregistrements ultérieurement, pour la constitution d'archives, par exemple. De plus, elle est indispensable pour supprimer, des index secondaires, les clés des enregistrements détruits.
La mise à jour des données en temps réel est certainement l'une des meilleures applications de l'organisation indexée.
Une donnée peut être tenue à jour au fur et à mesure du déroulement du programme d'application. Elle devient ainsi une information fiable en temps réel.
Ces modifications immédiates sont très souvent associées à la lecture des enregistrements.

C'est ainsi que, dans une banque, on peut enregistrer directement les prélèvements et les dépôts effectués sur les comptes en lisant les enregistrements à l'aide de leur clé (le numéro de compte) et en déduisant ou en ajoutant les sommes concernées avant de récrire les enregistrements dans le fichier.

La structure générale du fichier ISAM

L'organisation ISAM utilise deux structures de données : l'index et le fichier des données proprement dites. Plusieurs index coexistent lorsqu'on utilise des clés secondaires pour ouvrir différentes voies d'accès aux enregistrements de données. Les fichiers de données en ISAM sont enregistrés selon le format Basic et sont compatibles avec toutes les fonctions du Basic.

L'utilisation de programmes utilitaires comme Sort ou Merge ne présuppose donc aucune modification. Sort est un programme de tri tandis que Merge (fusion) sert à réunir plusieurs fichiers en un seul.

Il faut cependant remarquer que ces programmes entraînent la destruction des relations logiques entre le fichier de données et l'index.

Le fichier de données étant un fichier Basic à accès direct (aléatoire), il est possible de le manipuler par le biais des instructions Basic correspondantes. Quant au fichier index, il est constitué d'enregistrements contenant chacun un certain nombre de clés ; à chaque clé correspond un numéro d'enregistrement.

Lorsqu'une même clé revient plusieurs fois, le mécanisme de recherche est légèrement différent. On distingue donc deux sortes de fichier index, selon qu'ils acceptent les clés dupliquées ou les clés uniques.

Les clés sont toujours enregistrées dans l'ordre physique, mais elles sont reliées par des pointeurs qui rétablissent l'ordre alphanumérique. Une structure « en arbre » permet de maintenir cet ordre logique. L'arbre utilisé par la méthode ISAM s'appelle un arbre binaire.

La structure en arbre binaire équilibré

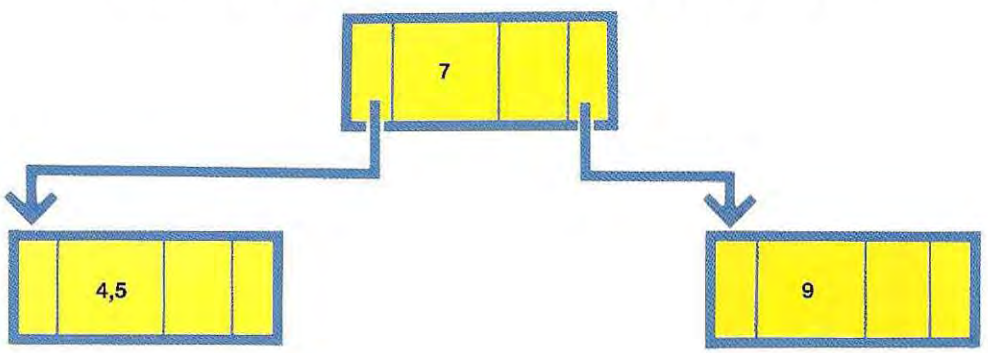
Avant d'aborder l'étude de l'arbre binaire (B-tree, en anglais) utilisé par ISAM, nous allons examiner une forme d'arbre équilibré plus simple, qui nous permettra d'en mieux comprendre le fonctionnement.

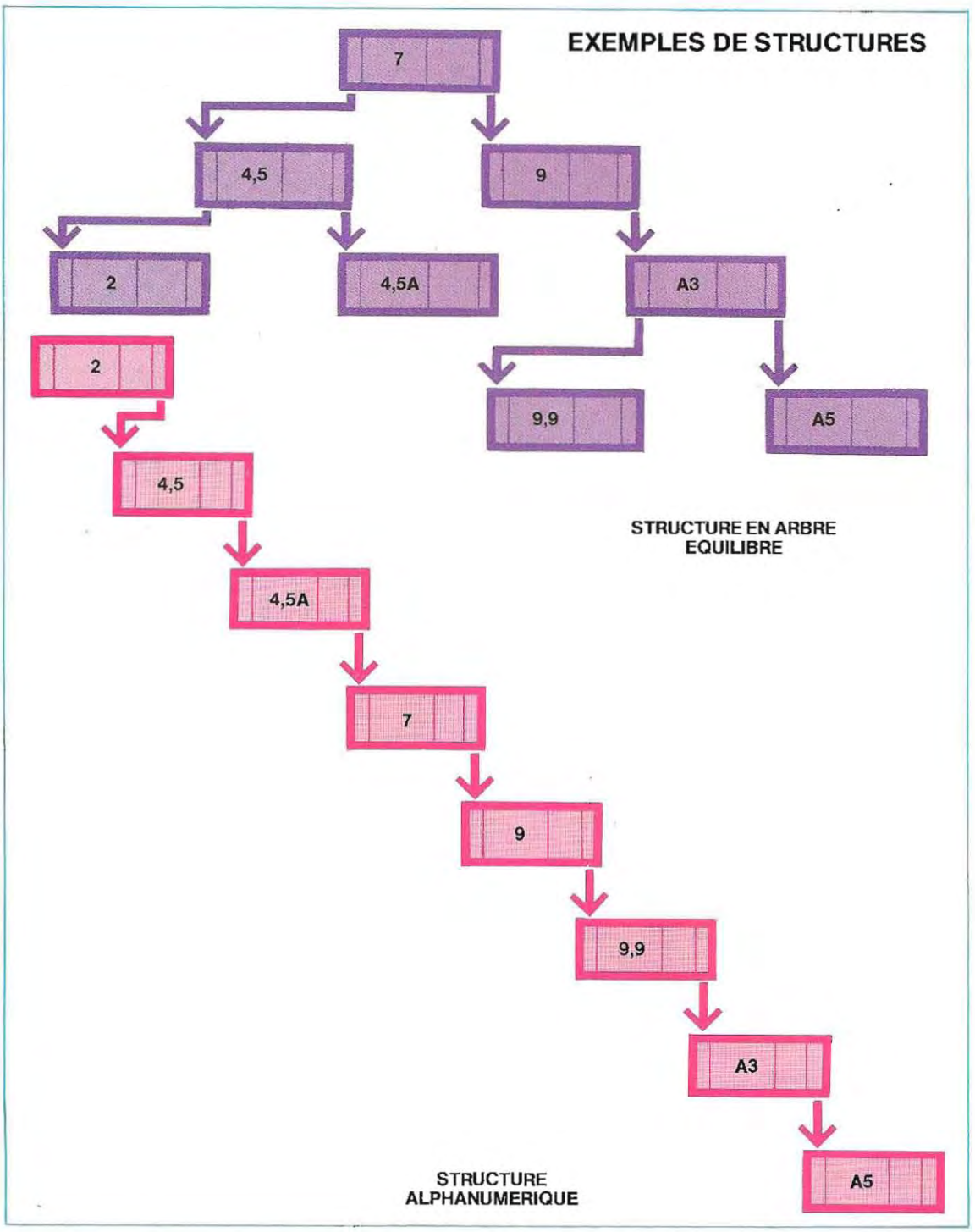
Dans un type particulier d'arbre binaire, chaque clé occupe un embranchement et deux pointeurs lui sont associés. L'un indique la clé qui précède immédiatement, l'autre celle qui suit immédiatement. Ces clés se trouvent, toutes deux, au niveau immédiatement inférieur.

Le schéma ci-dessous montre la structure binaire obtenue par l'ajout des clés 4,5 et 9 à la clé 7.

Chaque fois qu'un enregistrement est ajouté au fichier, la valeur de sa clé est comparée à celle de la clé qui se trouve tout en haut de l'arbre et que l'on appelle, en fait, la racine. Si la clé de l'enregistrement à ajouter est supérieure à la racine, on la compare à nouveau mais cette fois avec la clé indiquée par le pointeur qui part à droite de la racine. Si elle est inférieure, on la compare avec la branche de gauche. On effectue ainsi des comparaisons successives jusqu'à ce qu'on trouve la

EXEMPLE DE STRUCTURE EN ARBRE EQUILIBRE





position qui revient à la clé, au niveau le plus bas de l'arbre.

Ainsi, en ajoutant, par exemple, les clés 2, A3, A5, 4,5A et 9,9 à la structure illustrée page 256, nous obtiendrons l'arbre représenté ci-dessus en haut, à droite. Tant que l'arbre demeure équilibré, c'est-à-dire tant que le nombre de clés situées à la gauche d'une clé est comparable à celui des clés qui se trouvent à sa droite, la durée de recherche d'une clé est optimale.

On parvient ainsi à la clé A5 après trois com-

paraisons seulement (avec les clés 7, 9 et A3). Si, en revanche, les clés étaient disposées en ordre alphanumérique, on obtiendrait la structure représentée en bas, à gauche, du schéma de la page 257, et il faudrait alors sept comparaisons pour parvenir à cette même clé. Il est nécessaire de disposer d'un algorithme pour maintenir l'équilibre de l'arbre et éviter des recherches trop longues.

La structure en arbre ISAM

L'arbre binaire utilisé par ISAM est plus complexe en raison des deux types d'enregistrements qui coexistent dans cette méthode et auxquels correspondront des branches index et des branches terminales.

Les branches terminales sont les plus basses de l'arbre ; ce sont elles qui contiennent les clés et les numéros des enregistrements de données correspondants. Les index occupent, au contraire, tous les niveaux de l'arbre, excepté le niveau inférieur et constituent une voie d'accès aux branches terminales (voir schéma ci-dessous). Chaque ramification terminale possède, en outre, deux pointeurs, qui la rattachent à la branche terminale précédente et à la suivante. On peut ainsi accéder à toutes les clés en respectant l'ordre ASCII.

La longueur des enregistrements du fichier index est fixe, tandis que la longueur des clés est variable.

Le nombre de fichiers ouverts et les zones tampon

L'utilisateur doit déclarer le nombre maximum de fichiers index qui peuvent être ouverts en même temps. L'organisation ISAM bénéficie d'un programme de base* qui permet de définir ce nombre qui sera compris entre 1 et 15, par exemple.

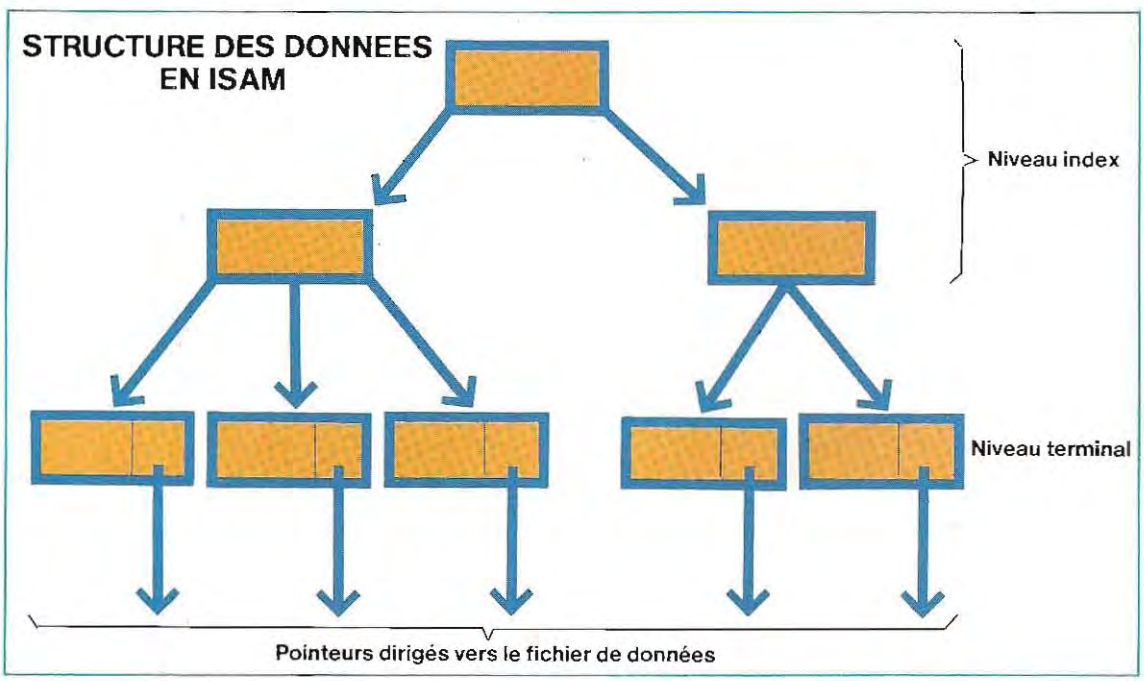
On doit attribuer aux fichiers index utilisés par ISAM des numéros différents de ceux qui sont associés aux fichiers de données. Si on a, par exemple, fait l'initialisation pour un nombre maximum de quatre fichiers en ISAM (soit 1, 2, 3 et 4), seuls restent disponibles pour les autres fichiers les numéros 5 à 15.

Si un programme en Basic a recours aux numéros 1, 2, 3 ou 4, des erreurs imprévisibles risquent de se produire. On peut éviter ces situations « conflictuelles » en prenant l'habitude de numéroter les fichiers par ordre décroissant en partant du plus élevé des nombres des fichiers disponibles.

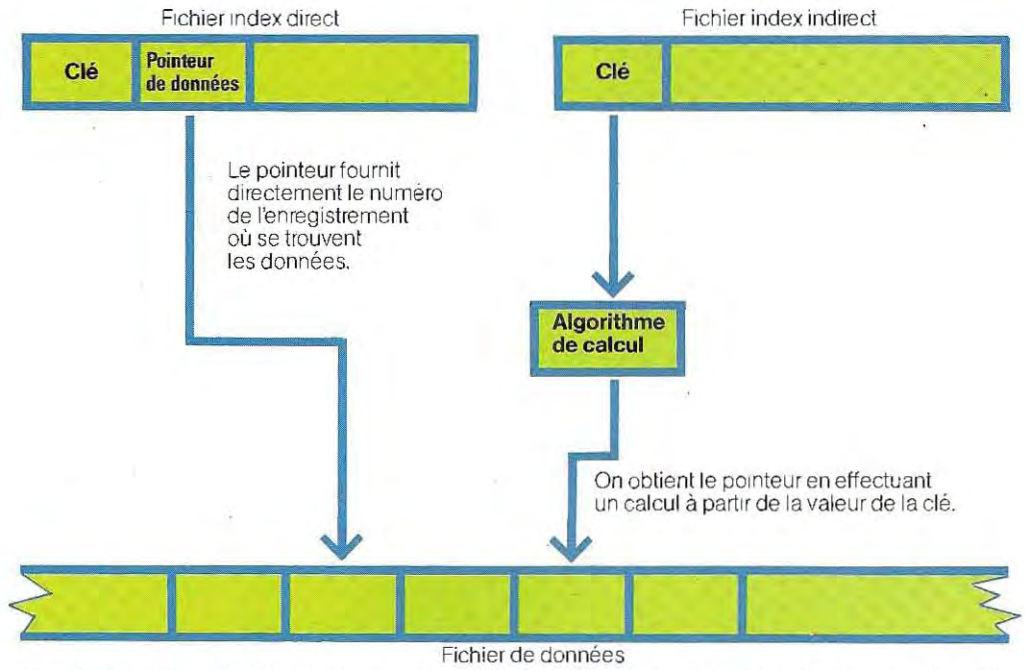
Les enregistrements du fichier index sont lus dans des zones de mémoire intermédiaires, ou zones tampon (buffers, en anglais), situées dans la mémoire centrale.

Le système réserve automatiquement un mi-

* Les programmes de base font partie du système d'exploitation et accomplissent des opérations indispensables au fonctionnement de la machine.



COMPARAISON DE LA GESTION DES DONNEES AVEC DES FICHIERS INDEX DIRECT ET INDIRECT



Dans le second cas, l'index prend moins de place puisqu'il ne contient pas de pointeur mais qu'on obtient ceux-ci en appliquant un algorithme à la valeur de la clé.

nimum de deux zones tampon par fichier afin de permettre l'exécution de l'algorithme d'équilibrage de la structure binaire.
L'utilisateur peut toutefois décider d'augmenter le nombre de ces zones tampon, dans les seules limites de l'espace mémoire disponible. Plus le nombre des fichiers augmente, plus l'espace nécessaire à leur gestion (et notamment, aux zones de contrôle des données) est grand, et moins il reste de place pour les zones tampon. Il est donc conseillé de réduire le nombre des fichiers ouverts en même temps, afin de libérer de la place en mémoire pour ces zones.
ISAM peut fonctionner avec une seule zone tampon par fichier, mais la rapidité de la méthode s'en trouve diminuée. Si ISAM peut faire tout cela, c'est qu'elle dispose d'algorithmes qui attribuent les zones tampon libérées par les derniers fichiers utilisés, tirant ainsi tout le parti possible de l'espace mémoire. Lorsqu'un fichier index a besoin d'une zone tampon supplémentaire et qu'elles sont toutes utilisées, celle qui a servi en dernier lui est affectée.

L'adressage indirect

Le plus simple, devant des fichiers indexés, est d'enregistrer, pour chaque clé, le numéro de l'enregistrement correspondant. Mais on peut également calculer l'adresse d'un enregistrement à partir de la valeur de sa clé. Cette méthode, plus compliquée que la méthode directe, permet de gagner de la place puisqu'on n'a plus besoin d'enregistrer les numéros des enregistrements à côté des clés des données.
Le schéma ci-dessus illustre les deux méthodes : dans la première, les pointeurs des données se trouvent dans l'index; dans la seconde (adressage indirect), on obtient le pointeur en effectuant le calcul adéquat sur la clé.
Les méthodes de calcul de l'adresse à partir de la clé sont parfois très complexes. Nous n'aborderons que les principales qui concernent uniquement les clés numériques.

Subdivision. On sépare les chiffres de la clé en un certain nombre de groupes que l'on

additionne. Le résultat constitue le pointeur. Soit une clé ayant pour valeur 537146. En la divisant en trois groupes de deux chiffres, on obtient 53, 71 et 46, dont la somme est égale à 170. Ce nombre est la valeur du pointeur. Lors de la saisie, les données qui ont pour clé 537146 seront écrites dans l'enregistrement 170. Lors de la recherche, il suffira d'additionner les chiffres de la clé pour retrouver les données.
Selon les applications, la clé est divisée en un plus ou moins grand nombre de groupes, qui contiennent eux-mêmes un nombre variable de chiffres.

Multiplication. Cette méthode est semblable à la précédente, mais on multiplie les chiffres au lieu de les additionner. Reprenons la clé précédente, 537146, et divisons-la en deux parties : 537 et 146. Leur produit est égal à 78402. On peut l'utiliser tel quel comme pointeur, ou n'en retenir qu'une partie, par exemple les trois chiffres du milieu. Le numéro de l'enregistrement sera respectivement 78402 et 840.

Multiplication par 11. On multiplie la clé par 11 et on prend le résultat, ou une partie du résultat, comme pointeur.

Puissance deux. Dans ce cas, le pointeur est une partie du carré de la clé. Si la clé est 271, son carré est 73441. En prenant les trois chiffres centraux, on aura 344 comme pointeur.
Toutes ces méthodes présentent les mêmes inconvénients : elles peuvent produire des « synonymes » et les données ne sont pas toujours bien réparties.
En effet, quelle que soit la méthode employée, il peut arriver que deux ou plusieurs clés différentes aboutissent à la même adresse. Si l'on prend, par exemple, les clés 7421 et 5243, on obtiendra, par la première méthode, 7421 = 74+21 = 95 et 5243 = 52+43 = 95.
Les deux données devraient donc normalement avoir le même pointeur (95). On surmonte cette difficulté en réservant une zone de l'index aux synonymes, c'est-à-dire à ces clés auxquelles l'algorithme attribue le même pointeur. Il est évident que ces synonymes doivent faire l'objet d'un traitement particulier. Il est également ennuyeux que l'algorithme

La gestion des banques de données exige d'imposantes mémoires de masse.

Marka

produise des valeurs trop proches car une partie du fichier de données est alors presque vide tandis qu'une autre partie est saturée. La seule solution est de changer d'algorithme.

Exemple d'application : la gestion du budget familial

Nous allons illustrer les techniques de manipulation des fichiers exposées jusqu'ici, en développant un programme de création et de maintenance pour le fichier d'un budget familial. La première étape, ou analyse, consiste à déterminer les résultats dont nous aurons besoin en sortie, et les entrées qui nous permettrons de les obtenir.

Le plan des comptes

Puisqu'il s'agit d'un budget, il faut d'abord opérer une première distinction entre deux types de mouvements de fonds, les recettes et les dépenses.
La **justification** de l'opération définit le type de mouvement concerné. Le **plan des comptes** est le tableau qui dénombre les différentes justifications.
Voici donc la liste de ces possibilités :

Type de compte
1 / Dépenses
2 / Recettes

Postes du compte de dépenses (10)	**Postes du compte de recettes (20)**
11 / Loyer	**21** / Salaire principal
12 / Gaz et électricité	**22** / Second salaire
13 / Téléphone	**23** / Rentrées imprévues
14 / Eau	
15 / Garde-robe	**24** / Rapports
16 / Loisirs	**25** / Intérêts
17 / Revues	
18 / Nourriture	

La numérotation constitue le code des différents postes du budget. Les numéros qui commencent par un **1** se rapportent au compte de dépenses tandis que ceux qui commencent par un **2** appartiennent au compte de recettes. L'ordinateur peut, à l'aide de ce code, déterminer s'il s'agit d'une somme à ajouter ou à déduire.
Lors de la saisie des données, il faudra accompagner le montant enregistré d'informations complémentaires :

Compteur progressif	3 caractères
Date	6 caractères
Justification	2 caractères
Montant	8 caractères
Commentaire	20 caractères
Opérateur	10 caractères
Espace disponible pour des développements ultérieurs	11 caractères
	60 caractères au total

Ces zones ont chacune une signification particulière (voir page 262 le schéma du format de cet enregistrement).

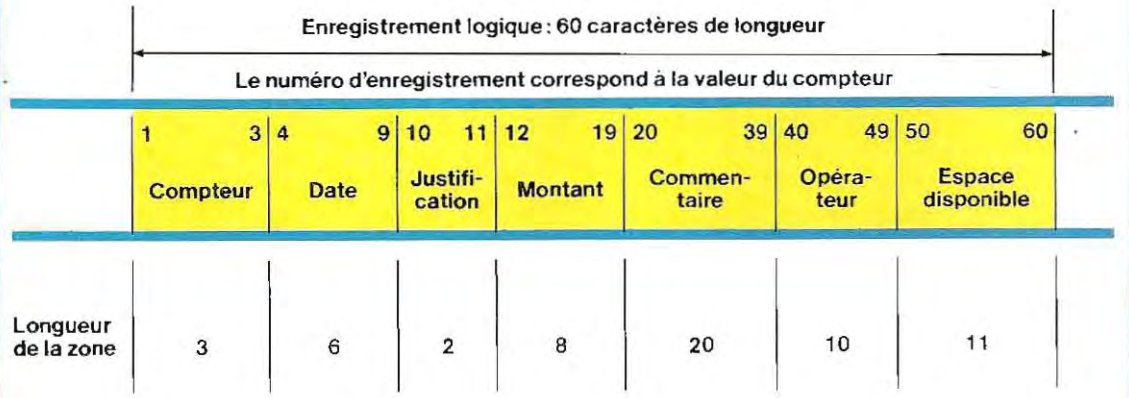
Le compteur. Il permet de retrouver l'ordre dans lequel les données ont été introduites. Il démarre à 1 pour la première opération et est ensuite incrémenté de 1 à chaque nouvelle opération. C'est, en fait, le numéro de l'enregistrement qui contient les données, et la clé qui permet de les relire. Afin d'éviter les erreurs, il doit être calculé automatiquement par la machine. Le programme doit donc, à chaque introduction nouvelle de donnée, ajouter 1 au numéro d'enregistrement et stocker ce numéro dans la zone réservée.

La date. C'est la date de l'opération exprimée sous la forme : jour (2 chiffres), mois (2 chiffres), année (2 chiffres).
Outre sa fonction d'aide-mémoire, cette zone joue le rôle d'indicatif de sélection pour l'impression des bilans. On peut donc demander des bilans des opérations d'une journée à partir de la date.

La justification. C'est la valeur numérique qui correspond à chacun des postes du budget.
Le plan qui nous sert d'exemple ne comporte que des codes compris entre 11 et 25 ; 2 caractères sont donc suffisants. Il faudra vérifier que la valeur indiquée pour cette zone à la saisie est bien comprise dans les limites prévues (de 11 à 18 et de 21 à 25 inclus).

Le montant. C'est le montant d'un mouvement de fonds. On a prévu 8 caractères, c'est-à-dire un maximum de 999 999,99. Cette limite ne joue que pour le montant d'une opération unique ; elle est beaucoup plus élevée pour le calcul des totaux.

STRUCTURE DES DONNEES DANS LE FICHIER DE GESTION DU BUDGET FAMILIAL



Le commentaire. Cette zone de 20 caractères ne sert qu'à commenter une opération ; à expliquer, par exemple, l'emploi d'une somme dans l'une des rubriques du budget.

L'opérateur. Cette zone peut contenir le nom du membre de la famille qui a effectué la dépense ou celui de la personne qui est responsable de l'encaissement. On peut s'en servir pour connaître le total des dépenses d'un membre de la famille ou le total des revenus d'une même source. Si l'on a plusieurs sources de revenus, on pourra, après en avoir précisé l'origine dans cette zone, savoir combien a rapporté, en tout, chacune d'elles.

L'espace disponible. La dernière zone est réservée aux futurs développements du fichier. Il est toujours bon, à la création d'un fichier, de prévoir de la place supplémentaire afin de pouvoir ajouter des données et de ne pas se retrouver avec un fichier inutilisable au cas où surgirait de nouveaux besoins.
Voici les caractéristiques de ces zones (le mot « caractère » est désigné par « ch ») :

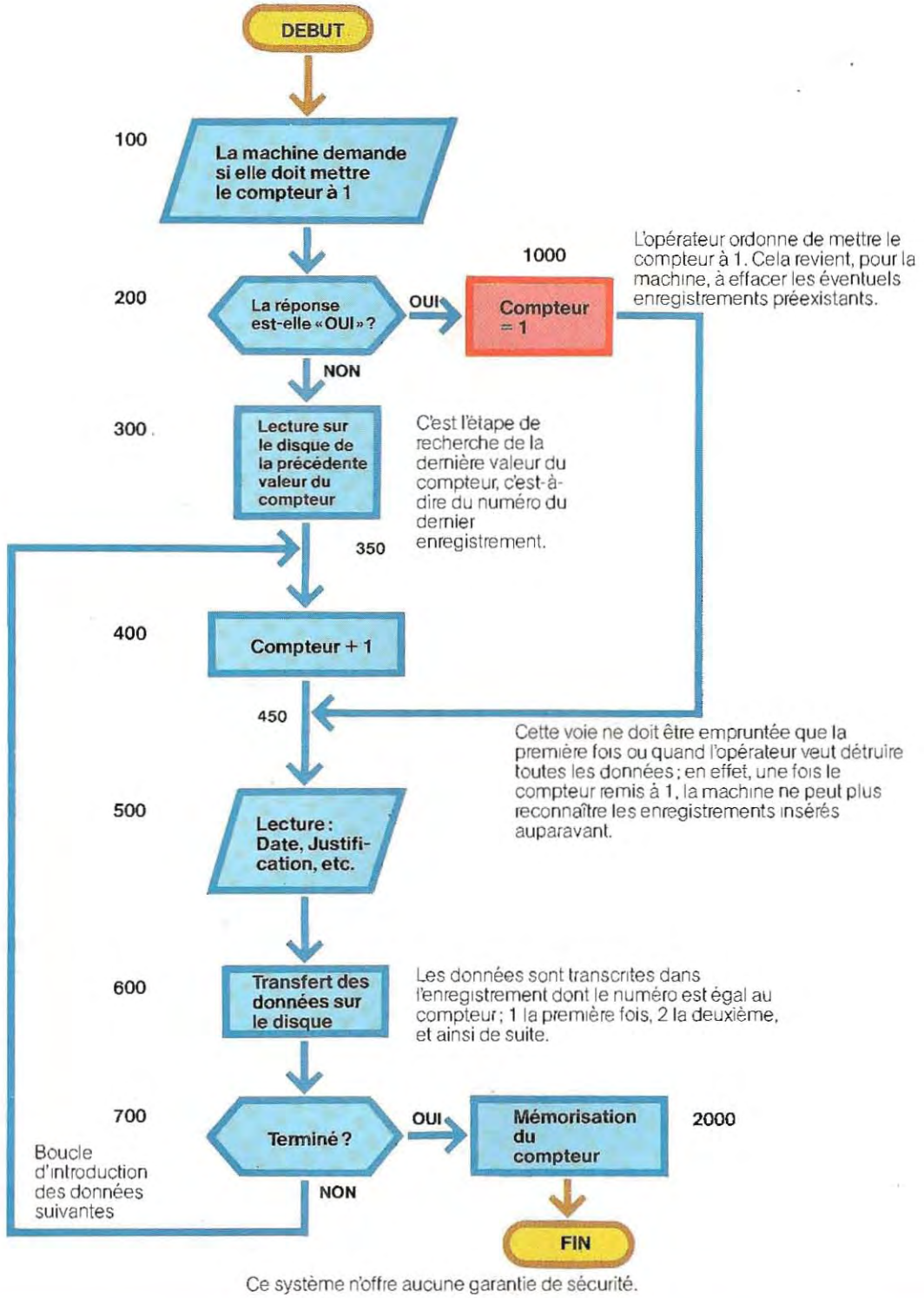
Compteur - 3 ch : augmentation automatique de 1 à chaque entrée.
Date - 6 ch : contrôle de compatibilité entre le jour et le mois (par exemple, il n'y a jamais de 31/02).
Justification - 2 ch : elle va de 11 à 18 inclus et de 21 à 25 inclus.
Montant - 8 ch : cette zone ne peut contenir que des chiffres.
Commentaire - 20 ch : pas de contrôle.
Opérateur - 10 ch : pas de contrôle.

On voit clairement la nécessité de prévoir, dans le programme, deux modules particuliers, l'un pour le calcul automatique du compteur, l'autre pour les contrôles de compatibilité et de numéricité sur la date, la justification et le montant.

Le module de calcul du compteur d'enregistrements

A la première introduction de données, et à ce moment-là seulement, il faut mettre le compteur à 1 ; toutes les saisies qui auront lieu ensuite provoqueront alors son incrémentation de 1.
Le module enregistre la dernière valeur du compteur après chaque entrée et la rappelle lorsque l'utilisateur reprend le travail. La mémorisation du compteur doit obligatoirement se faire sur disque (ou sur bande). En effet, si elle ne se faisait qu'en mémoire centrale, la donnée serait perdue à la première coupure d'alimentation (les mémoires électroniques sont « volatiles », ou non permanentes, c'est-à-dire qu'elles perdent leur contenu dès que le courant est coupé).
L'organigramme de ces opérations est représenté par le schéma de la page 263.
Ce déroulement logique recèle deux graves défauts qui compromettent le fonctionnement du programme.

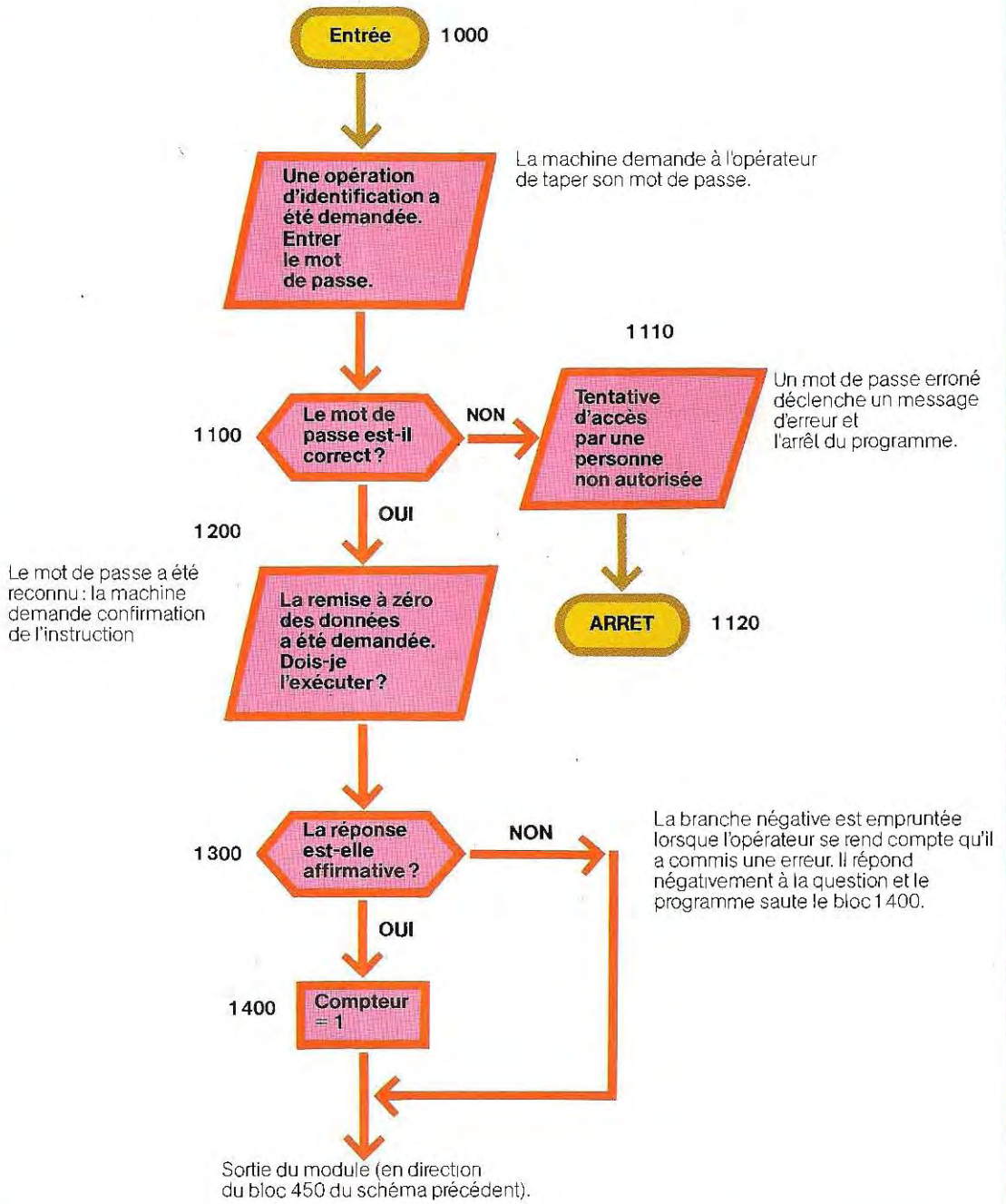
ORGANIGRAMME DE L'INTRODUCTION DES DONNEES

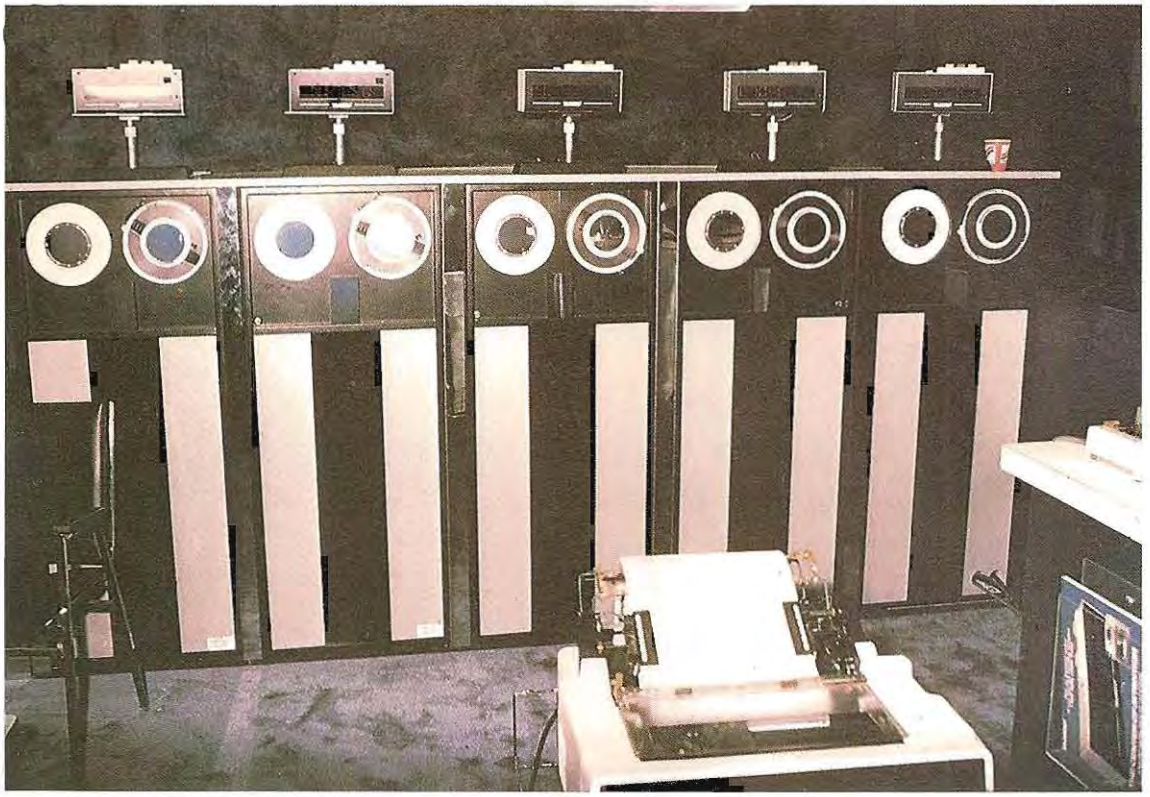


Ce système n'offre aucune garantie de sécurité.

En effet, si le bloc 1 000 (mise du compteur à 1) est utilisé à mauvais escient, tous les enregistrements précédents seront perdus. Supposons que nous en soyons à l'introduction de la donnée numéro 176 mémorisée dans le compteur. La prochaine donnée sera donc enregistrée à l'emplacement 176 + 1, c'est-à-dire dans une zone disponible.

MODULE 1000 DEMANDANT L'ENTREE D'UN MOT DE PASSE ET UNE CONFIRMATION POUR TOUTE OPERATION QUI POURRAIT ENTRAINER LA PERTE DE DONNEES





Marka

Vue partielle (dérouleurs de bande) de la mémoire de masse d'une banque de données.

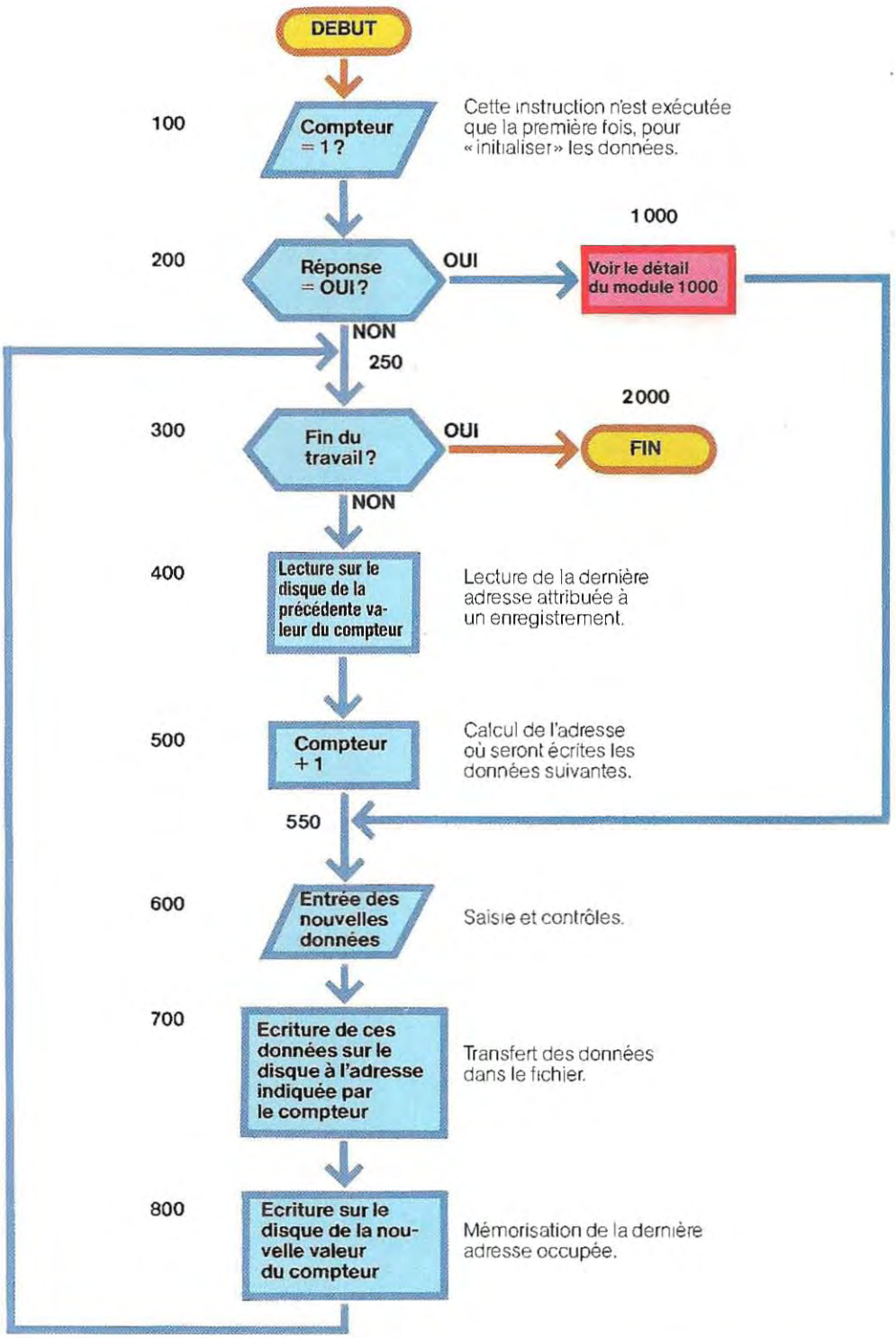
Inversement si, après la donnée 176, nous déclenchons par erreur l'exécution du module 1 000, la donnée suivante sera écrite en position 1 (puisque le compteur aura été remis à 1 par le module 1 000) et non, comme elle aurait dû l'être, en position 177. En s'écrivant par-dessus le précédent contenu de la position 1, elle l'« écrasera », c'est-à-dire l'effacera. En outre, la machine ne connaîtra plus le nombre des données réellement présentes.

La seule façon de réduire ces risques d'erreur est de subordonner l'exécution du module 1 000 à la reconnaissance d'un mot de passe et à une demande de confirmation. On a recours à cette double vérification toutes les fois qu'une distraction ou une mauvaise utilisation risqueraient d'entraîner la perte des données.

Dans notre exemple, la perte des données n'est qu'apparente ou, du moins, limitée, puisqu'il suffit de remettre le compteur à 176 pour les récupérer. Dans d'autres cas, elle est irrémédiable et lourde de conséquences. Qu'on essaie d'imaginer ce que signifie la destruction des fichiers contenant les informations économiques d'une grande entreprise. L'organigramme de la page 264 représente le module 1 000 amélioré : les contrôles appropriés ont été ajoutés pour protéger les données contre les erreurs humaines.

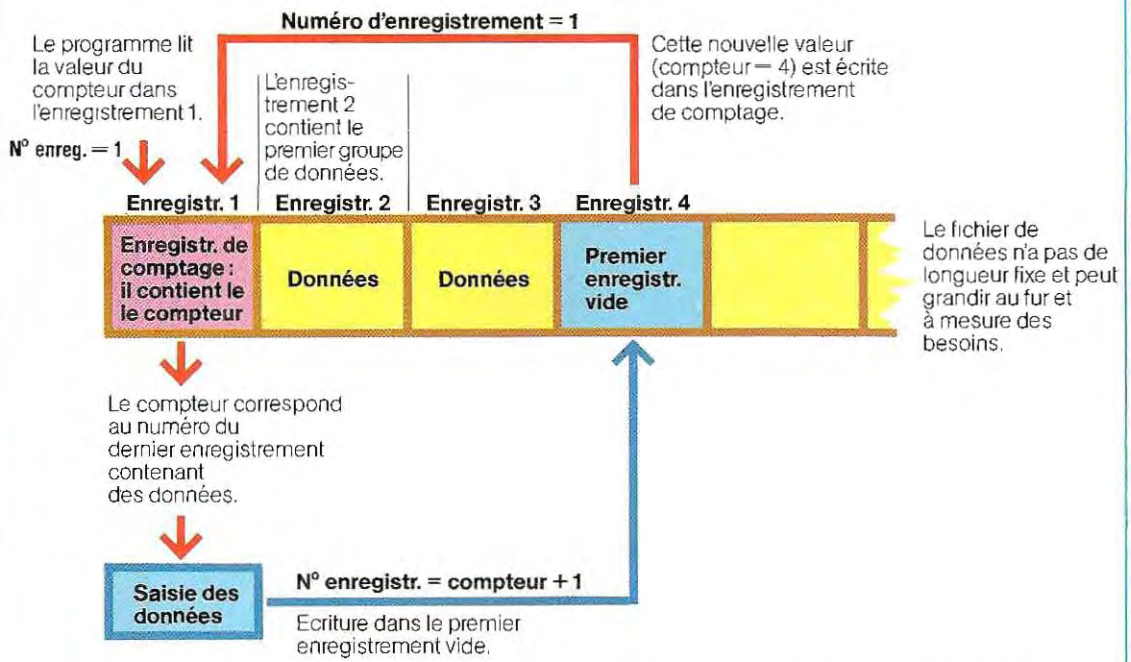
Venons-en maintenant au second défaut du programme. Imaginons que nous en sommes à l'enregistrement 121 au cours d'une saisie de données. Cette valeur n'a pas été mémorisée sur le disque puisque, comme le montre le schéma de la page 263, ce transfert n'a lieu qu'en fin de travail. C'est alors que se produit une panne de courant. Puis le courant est rétabli. Mais le compteur n'est plus à 121 (il est à zéro, à « blanc » ; ou, pire, il contient un nombre aléatoire) et toutes les données saisies avant la panne ne peuvent plus, malgré leur présence dans le fichier, être considérées comme acquises puisque le système en a perdu l'adresse (la position). Contre ce risque, il n'y a qu'une solution : mémoriser le compteur à chaque introduction de données. L'organigramme définitivement corrigé est représenté page 266.

ORGANIGRAMME DE SAISIE DES DONNEES MODIFIE POUR LA MISE A JOUR DU COMPTEUR A CHAQUE ENTREE

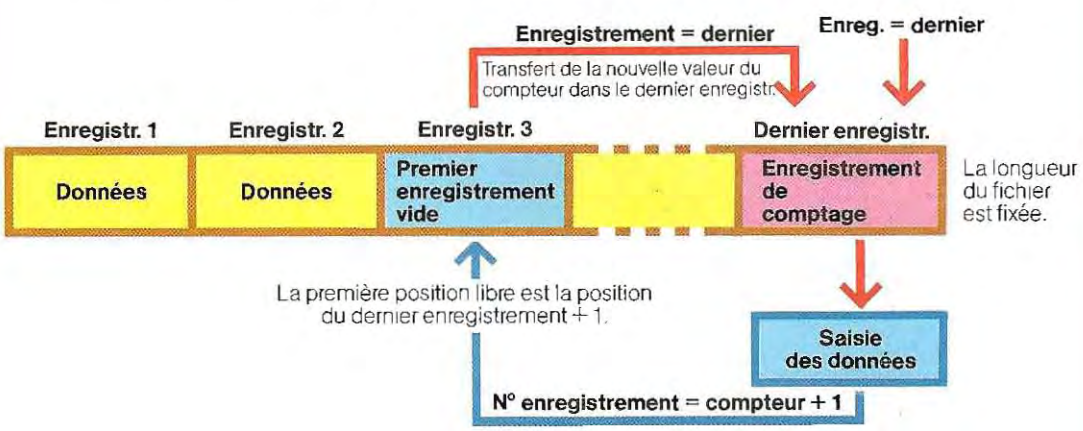


C'est alors que se pose la question de savoir où va s'inscrire le compteur. Il est possible d'utiliser un enregistrement donné du fichier, dont toutes les zones seront vides, excepté celle du compteur. Bien qu'appartenant au fichier de données, cet enregistrement n'est qu'une zone de comptage temporaire dans lequel on peut lire ou écrire le compteur. S'il est placé au début du fichier, le premier enregistrement disponible pour l'écriture des données est alors le numéro 2 et la donnée 1 n'existe pas. Cette solution permet de ne pas spécifier d'avance la longueur du fichier qui pourra s'accroître éventuellement au fur et à mesure des besoins.

STRUCTURE DES FICHIERS DE DONNEES CONTENANT UN ENREGISTREMENT DE COMPTAGE



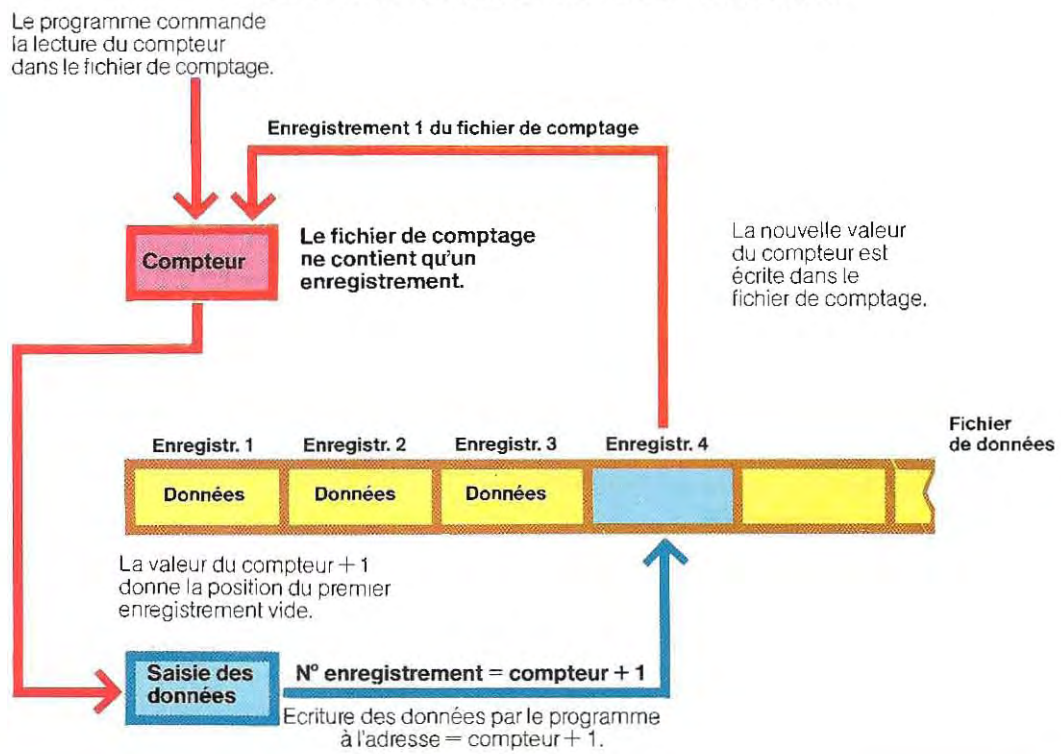
L'enregistrement de comptage se trouve en position 1. Les données commencent au numéro 2 et la longueur du fichier n'est pas prédéterminée.



L'enregistrement de comptage se trouve en fin de fichier. Les données commencent au numéro 1 mais la longueur du fichier est limitée.

En revanche, si l'enregistrement de comptage se situe à la fin du fichier de données, la numérotation commence au numéro 1 puisque le premier enregistrement n'est plus destiné au comptage. Il faut déterminer à l'avance la longueur du fichier, l'adresse de l'enregistrement de comptage étant celle du dernier enregistrement du fichier. Les schémas de la page 267 représentent ces deux possibilités et les voies d'accès aux enregistrements correspondants.
Une seconde solution consiste à créer un nouveau fichier ne contenant qu'un enregistrement : on y mémorise le compteur. Les données sont alors numérotées à partir de 1, et l'on conserve les avantages d'un fichier de longueur indéterminée. Mais on doit alors gérer simultanément deux fichiers : celui des données et celui du comptage.
Le schéma ci-dessous illustre cette structure

GESTION DU COMPTEUR SUR UN FICHIER DE COMPTAGE SEPARE : INTERACTION ENTRE LES DEUX FICHIERS



et son utilisation. Quelle que soit la méthode choisie, lecture et écriture du compteur sont effectuées par les modules 400 et 800 de

l'organigramme général (page 266). En revanche, le contenu des modules varie en fonction de la méthode adoptée. On peut voir ci-dessous le module de lecture tel qu'il se présente quand on a recours à un fichier de comptage appelé DIR ; le module d'écriture (800) se décomposerait de façon analogue.

LECTURE DE LA VALEUR DU COMPTEUR DANS LE FICHIER DIR

Module : 400.

Fonction : lecture de la zone compteur dans le fichier DIR.

Nom du fichier : DIR.

Longueur : 1 enregistrement (60 caractères).

Position de la zone compteur : 1 – 3

Sortie : valeur numérique du compteur.

L'écriture (module 800) se déroule de façon identique.

Le module de contrôle des données en entrée

Lors de l'introduction (saisie) des données, il faut contrôler l'exactitude formelle des zones Date, Justification et Montant.

Cette opération (détaillée page 270) doit s'effectuer dans le module d'introduction des données (600 sur le schéma de la page 266), afin que seules des données exactes puissent être acheminées vers la suite du programme. Le module (ou « bloc ») 620 concernant le contrôle de la date, fera l'objet d'un développement ultérieur car il revient souvent dans les programmes. Considérons, pour l'instant, qu'il permet de vérifier la compatibilité entre le jour et le mois, et de répondre par une variable d'erreur. Si celle-ci est nulle, la date est exacte ; sinon elle est erronée et la

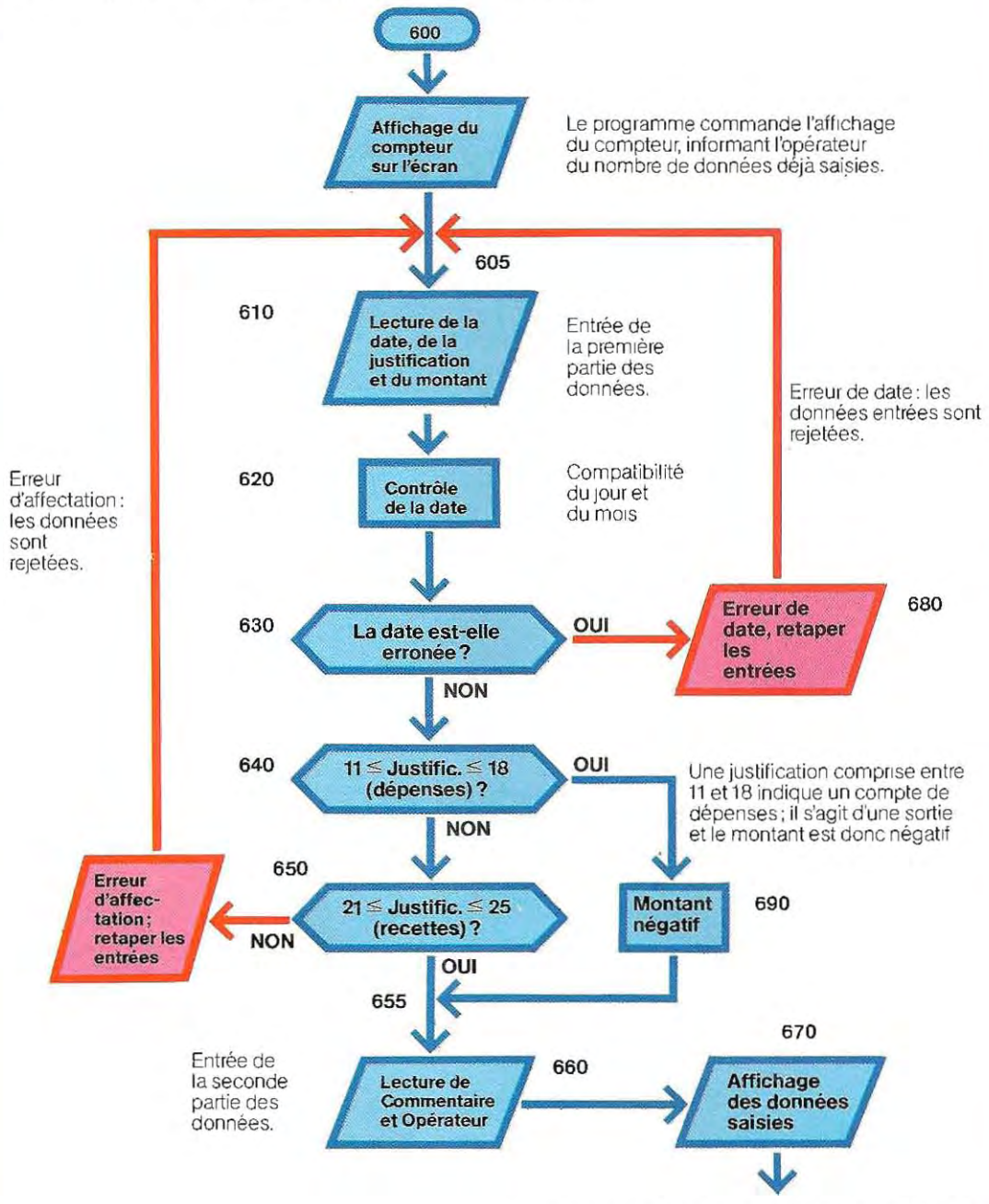
ORGANIGRAMME DETAILLE DU MODULE DE SAISIE ET DE CONTROLE DES DONNEES

Module : 600.

Fonction : entrée au clavier des nouvelles données et contrôle formel.

Entrée : compteur

Sortie : zones de données prêtes à être transférées sur disque.



valeur de la variable renseigne alors sur le type d'erreur commis. Ce module peut également servir à limiter dans le temps l'utilisation des programmes. Si, par exemple, nous posons comme condition que l'année ne peut être postérieure à 1995, le programme n'acceptera plus les saisies en 1996. Dans le cas où un utilisateur ne dispose que de la version compilée du programme (c'est-à-dire en langage machine), il ne peut modifier lui-même cette date limite. Seule peut le faire la personne qui détient le programme source. Naturellement, après les corrections, on devra recompiler le programme.
La seule ressource de l'utilisateur serait de fournir une fausse date à l'ordinateur, en continuant, par exemple, de taper 1995 au lieu de 1996. Naturellement, cette solution n'est envisageable que pour les programmes à usage personnel et certainement pas sur des programmes d'utilisation officielle (fiscale par exemple).
Cette technique de péremption permet de louer des programmes dont la date d'échéance ne sera remise à jour que si l'utilisateur a payé sa redevance.

En résumé :

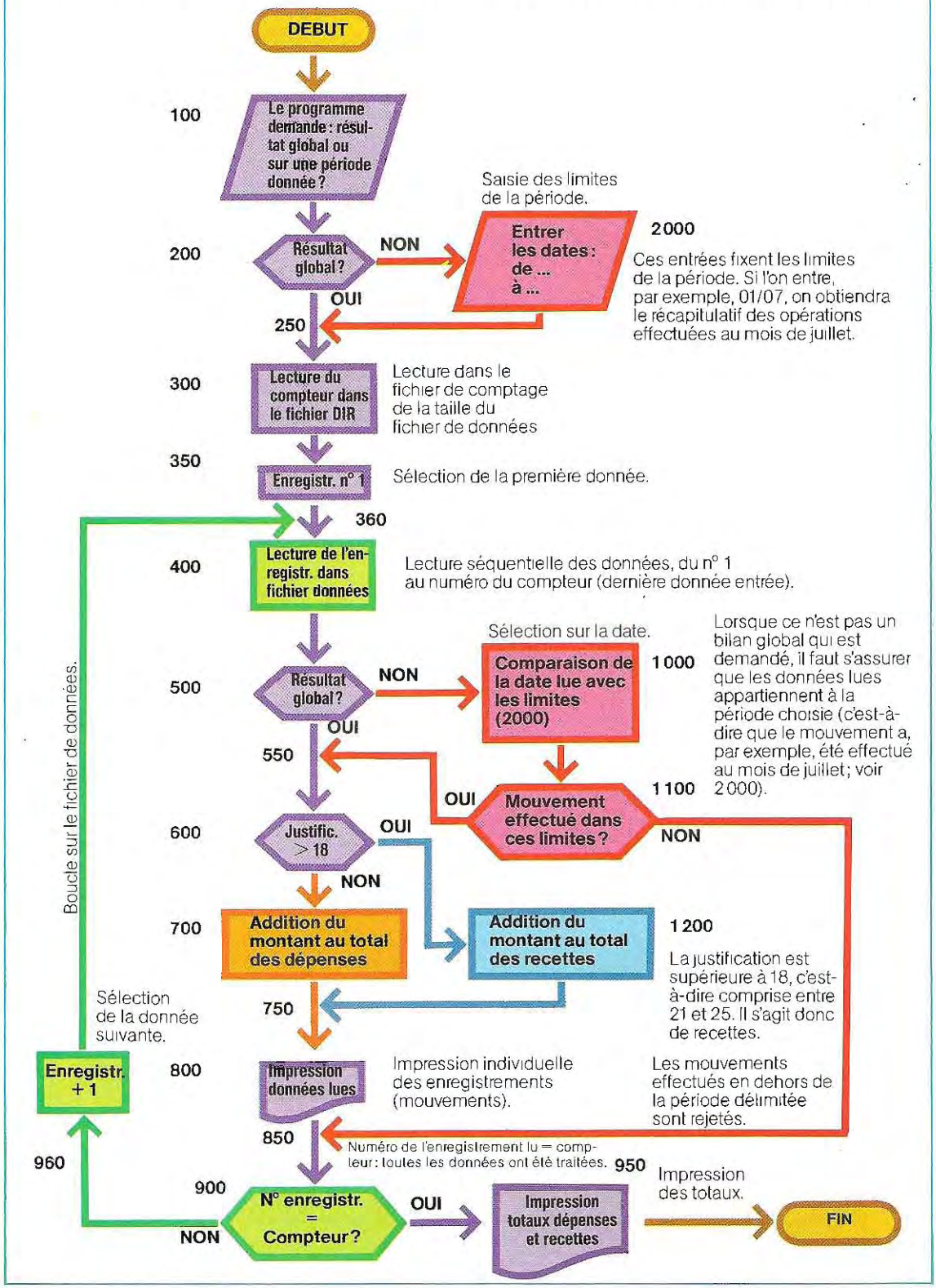
- **la structure de l'enregistrement de données et les longueurs des zones sont schématisées page 262 ;**
- **l'organigramme du programme de saisie et de contrôle des données complet se trouve page 266 ;**
- **l'initialisation est effectuée par le module 1000 dont l'organigramme se trouve page 264 ;**
- **le compteur de données est mémorisé dans un fichier auxiliaire de comptage (voir schémas pages 268 et 269) ;**
- **les opérations de saisie et de contrôle des données sont détaillées page 270. En traduisant dans l'ordre les organigrammes des pages 264, 266, 269 et 270 en instructions, nous obtiendrons notre programme.**

Le module 600 permet la saisie des données. Il faut, à présent, compléter la procédure par le développement des calculs.
Ce nouveau traitement doit lire tout le fichier de données et additionner séparément les montants des dépenses et des recettes. Nous aurons ainsi une vue générale des mouvements effectués et le résultat global (la différence entre le total des recettes et le total des dépenses). En plus de ce résultat, il faut prévoir la possibilité d'effectuer des bilans partiels entre deux dates déterminées. Nous pourrons ainsi demander soit une situation globale (prenant en compte les données du début à la fin), soit la situation d'une période déterminée.
L'organigramme de ce traitement est représenté page 272.
Les programmes « auxiliaires » concernant des données particulières (justification, commentaire, opérateur) feront l'objet d'une explication générale ultérieure.
Quant aux modules qui apparaissent page 272, ils seront développés en détail lorsque nous rédigerons le programme après avoir étudié les instructions Basic nécessaires.

La recherche des données dans les fichiers

Une fois le fichier constitué (phase de création), il faut pouvoir modifier ses données (phase de mise à jour).
Pour mettre à jour un enregistrement, on doit d'abord le lire, puis, en fonction de son contenu, effectuer les ajouts et les modifications qui s'imposent.
Accéder aux données selon une logique déterminée constitue la phase principale d'utilisation d'un fichier.
Pour la mise à jour, l'utilisateur recherche l'enregistrement possédant la clé voulue et en extrait les données correspondantes pour les modifier.
Pour un traitement particulier (listing par exemple), il s'agit d'extraire du fichier une partie des données. On peut ainsi rechercher dans un fichier d'adresses celles qui se situent dans une même ville, ou dans un fichier du personnel tous les employés d'une même catégorie.
Il est donc indispensable de disposer de programmes de recherche qui retrouvent tous les enregistrements dont les données répondent

ORGANIGRAMME DU PROGRAMME DE CALCUL ET D'IMPRESSION DU BUDGET



aux critères voulus.
Les deux méthodes de recherche les plus utilisées sont la recherche séquentielle et la recherche dichotomique.

La recherche séquentielle

La recherche séquentielle d'une donnée implique la lecture successive de tous les enregistrements du fichier et leur comparaison avec la valeur de la clé de recherche. S'il y a égalité, la donnée trouvée est imprimée, sinon la recherche se poursuit jusqu'à la fin des enregistrements.
Le nombre moyen de lectures à effectuer s'obtient par la formule suivante :

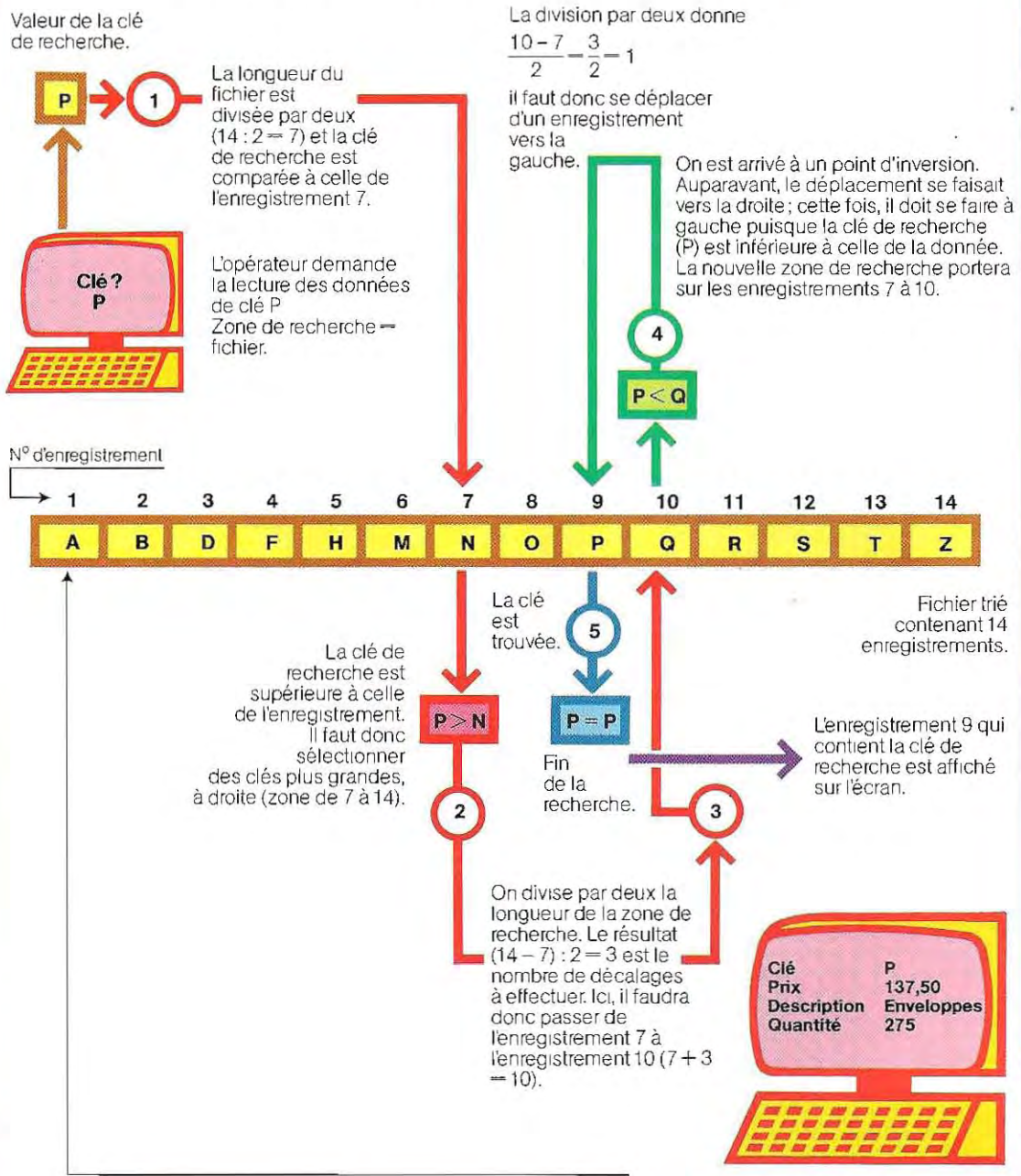
$$\text{nombre moyen de lectures} = \frac{\text{longueur du fichier} + 1}{2}$$

Dans un fichier de 1 000 enregistrements, il nous faudra donc en lire 500 en moyenne avant de trouver celui qui nous intéresse. De plus, comme nous l'avons vu, la donnée recherchée peut très bien apparaître dans plusieurs enregistrements ; il faudra donc poursuivre la recherche jusqu'à la fin du fichier. Le seul moyen d'accélérer cette opération est d'ordonner préalablement les données.

Deux étapes du travail effectué sur les banques de données. L'information doit être parfaitement planifiée et structurée afin d'utiliser au mieux la capacité de la mémoire.

Marka

LOGIQUE D'UNE RECHERCHE DICHOTOMIQUE



Clé	Prix	Description	Quantité
Contenu	A	318	Stylos à bille	106
Nombre de caractères	1	6	20	3

Longueur de l'enregistrement = 1 + 6 + 20 + 3 = 30 caractères.

La recherche séquentielle s'avère très longue. Aussi ne l'applique-t-on qu'à des petits fichiers, pour lesquels le gain de temps permis par des méthodes plus rapides, mais difficilement mises en œuvre, serait négligeable.

La recherche dichotomique

Elle ne peut se pratiquer que sur des fichiers ordonnés. On procède à des divisions binaires (en deux parties) successives du fichier. On lit ainsi l'enregistrement situé au milieu, on compare sa clé à la clé de recherche et on rejette alors l'une des parties (la droite si la clé de recherche est inférieure à celle de l'enregistrement lu, la gauche dans le cas contraire, en supposant le fichier classé par ordre croissant). On répète l'opération jusqu'à la découverte de la donnée recherchée. Prenons un exemple (voir schéma page 274).

- Zone clé, 1 caractère alphabétique.
- Prix des articles, 6 caractères numériques*.
- Description de l'article, 20 caractères alphabétiques.
- Quantité en stock, 3 caractères numériques*.

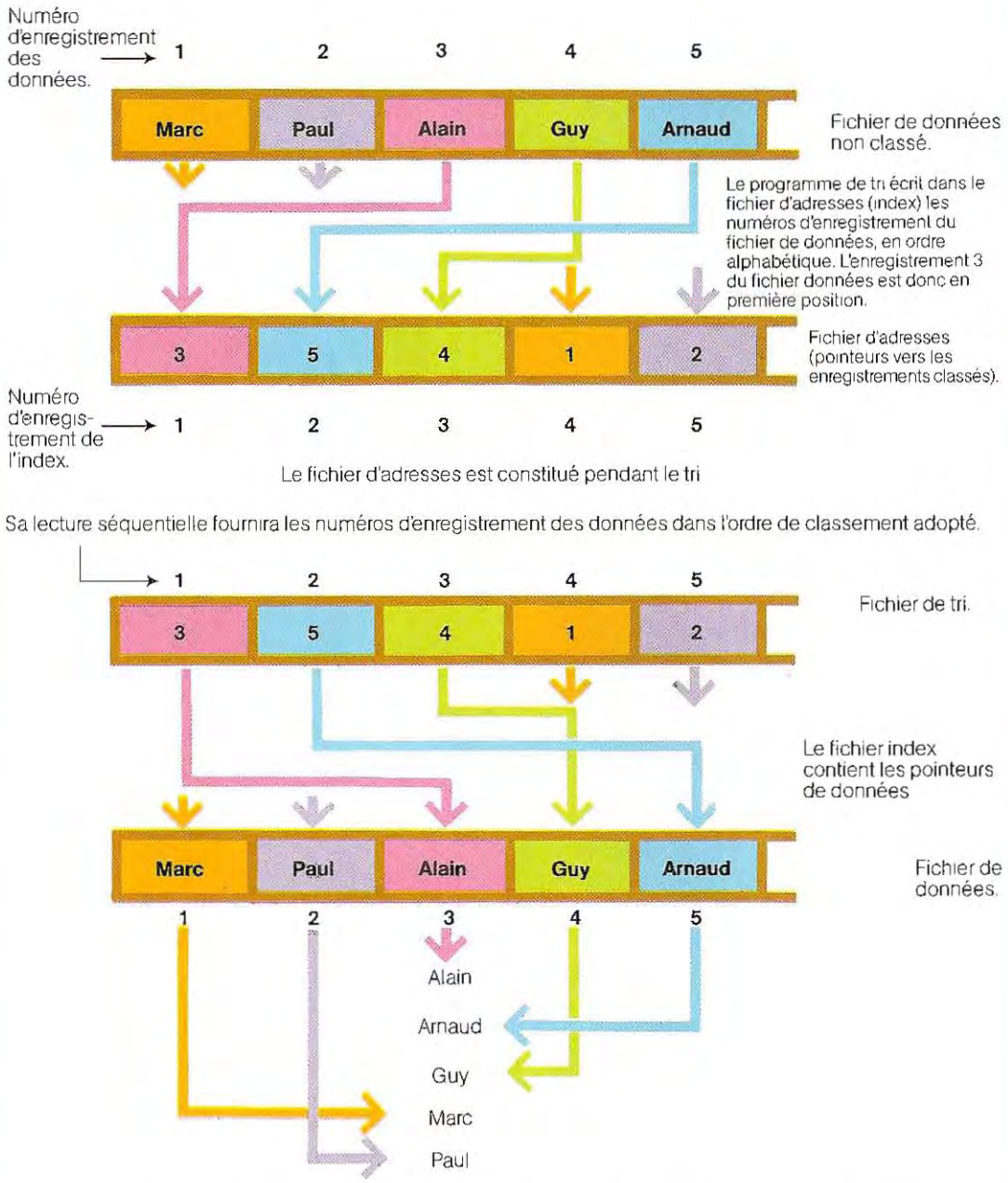
L'utilisateur veut connaître le contenu de l'enregistrement dont la clé est P. La recherche dichotomique s'effectue comme suit.
On divise d'abord (point **1**) la longueur du fichier par 2 ; ici, 14 : 2 = 7.
Ensuite (point **2**), la clé de l'enregistrement 7 (N) est comparée à la clé de recherche (P). Or, P est placé après N (P > N) ; l'enregistrement de clé P se trouve donc à droite, entre les enregistrements 7 et 14 (fin du fichier). Cette zone est à son tour divisée en deux. (14 – 7) : 2 = 3 (on ne prend que la partie entière du quotient). On obtient, en nombre d'enregistrements, l'ampleur du déplacement à effectuer vers la droite, à partir du n° 7.
En se décalant, vers la droite, de trois enregistrements à partir du numéro 7, on va lire l'enregistrement 7 + 3 = 10 (point **3**).
Cet enregistrement ayant une clé supérieure à P (point **4**), celui qui est recherché se trouve, cette fois, à sa gauche. De même que tout à l'heure notre donnée se trouvait entre les enregistrements 7 et 14, on la localise, maintenant entre les enregistrements 7 et 10. Un nouveau découpage en deux de la partie à étudier (10 – 7) : 2 = 1 indiquera le déplacement à effectuer vers la gauche à partir de la position actuelle.
L'enregistrement 10 – 1 = 9 qui est lu contient P dans sa zone clé (point **5**). La recherche s'achève et les données de l'enregistrement 9 sont affichées à l'écran.
Avec cette méthode, on n'a lu que trois des quatorze enregistrements du fichier (7, 10 et 9) alors qu'en recherche séquentielle nous aurions dû effectuer neuf lectures (l'enregistrement de clé P et les huit précédents). L'avantage de la recherche dichotomique est évident.
Une réserve s'impose cependant. On ne peut faire une recherche dichotomique que sur un fichier ordonné. Sinon, il faudra d'abord en classer les données. Cette méthode dichotomique impose un programme plus complexe que la méthode séquentielle et ne fait donc pas forcément gagner de temps lorsque le fichier ne comporte qu'un petit nombre d'enregistrements.

Le classement des données

Le classement des données est l'une des tâches principales de la gestion des fichiers. C'est, en effet, une condition fondamentale pour l'application de certaines techniques de traitement (comme la recherche dichotomique) et pour l'obtention de listings faciles à consulter.
Lorsqu'on ne peut obtenir de listing des données classées, un fichier, même de modestes dimensions, s'avère très difficile à exploiter, car il est pratiquement impossible d'y trouver les éléments que l'on recherche. C'est un peu comme si l'on disposait d'un annuaire téléphonique où les noms ne seraient pas répertoriés par ordre alphabétique mais dans l'ordre chronologique de souscription du contrat d'abonnement ! M. Martin, abonné de longue date, figurerait donc avant M. Leblanc et il serait impossible de retrouver

* Dans les ordinateurs individuels, les nombres sont souvent enregistrés sur les disques comme des caractères. Le nombre 579, par exemple, est enregistré sur trois octets, qui contiennent, dans l'ordre, les chiffres 5, 7 et 9, codés comme des caractères, respectivement par les valeurs hexadécimales 35, 37 et 39 (se reporter au code ASCII). Avec d'autres ordinateurs, on peut, au contraire, enregistrer les nombres tels quels sur les disques, ce qui occupe moins de place. Nous reviendrons plus loin sur cette question.

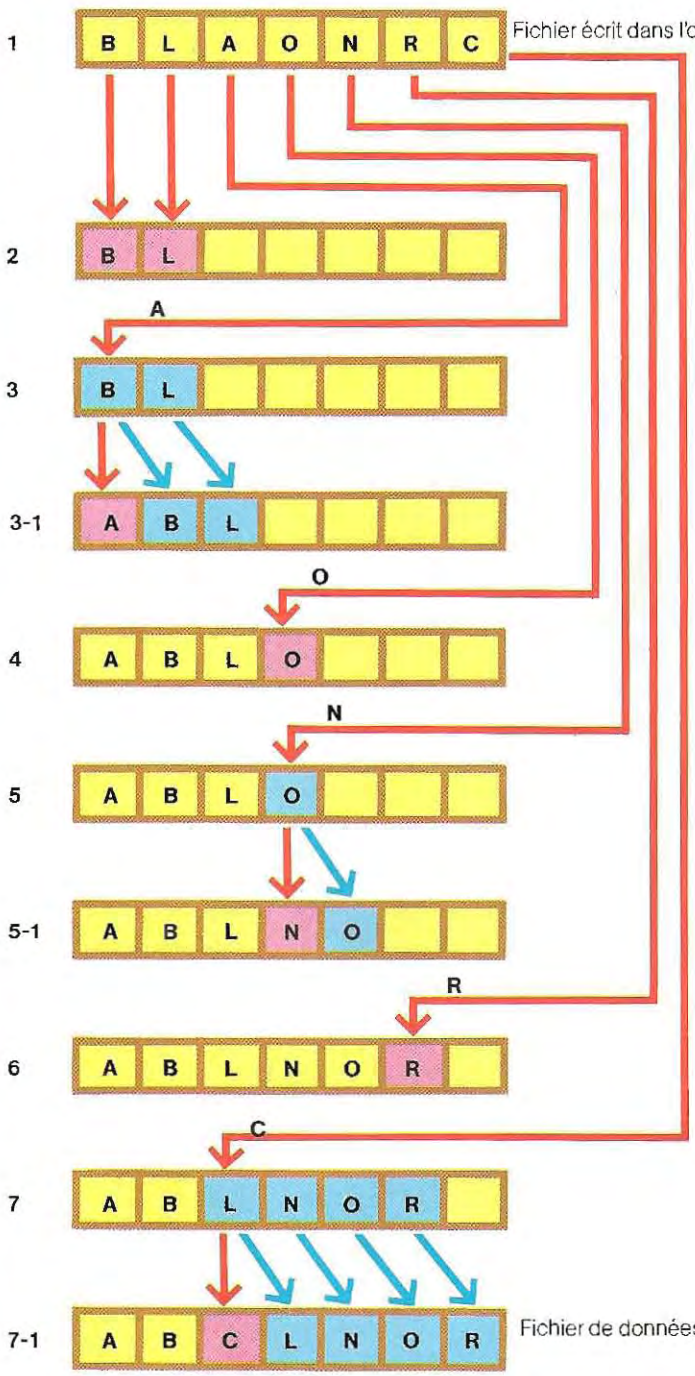
TRI DES DONNEES A L'AIDE D'UN FICHIER INDEX



un numéro de téléphone donné puisqu'aucune logique de recherche ne serait applicable en temps réel. De même, pour l'utilisation des données d'un fichier ; c'est pourquoi toutes les procédures de gestion de fichiers

possèdent des programmes de classement des données, appelés des programmes de tri.
L'importance de ces programmes est telle

TRI DES DONNEES PAR INSERTION DANS UN FICHIER PARALLELE



Fichier parallèle dans lequel les données sont transférées de façon ordonnée.

qu'ils sont inclus dans presque tous les systèmes d'exploitation, et généralement écrits en Assembleur pour accroître leur rapidité d'exécution.

A ces programmes de tri « utilitaires », c'est-à-dire accessibles à l'utilisateur, ce dernier ne fournit que le nom du fichier, la structure de l'enregistrement (nombre et longueur des zones de données) et la clé en fonction de laquelle il veut ordonner le fichier.

On distingue deux types de programme de tri suivant la structure des résultats qu'ils obtiennent. Certains recopient le fichier, en l'ordonnant, sur une nouvelle partie du disque, créant ainsi un nouveau fichier ordonné, et sous un nom différent du premier. D'autres constituent un fichier d'adresses qui contient (dans l'ordre) les adresses des données du fichier d'origine.

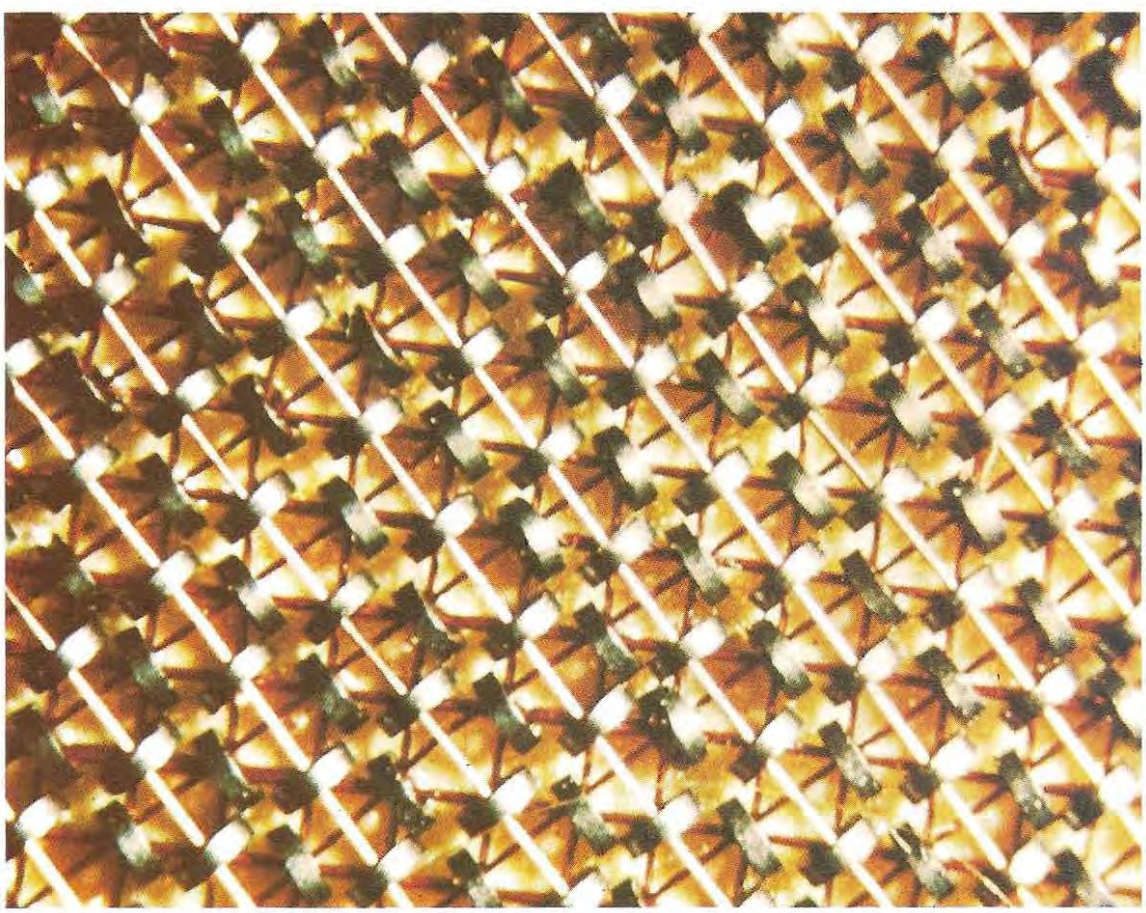
Pour lire les données dans l'ordre, il faudra donc, dans ce dernier cas, lire d'abord le fichier d'adresses, puis l'enregistrement de données pointé par l'adresse trouvée. Cette méthode est illustrée par le schéma de la page 276.

Cette méthode est plus rapide (puisqu'on n'a pas à recopier tout le fichier mais seulement les adresses) et elle permet d'occuper moins de place sur le disque en évitant de recopier des données déjà existantes (redondance)..

Si tous les mini-ordinateurs sont proposés avec des programmes de tri, certains micro-ordinateurs en sont dépourvus. C'est alors le programmeur qui doit se charger de les écrire lui-même. Ce qui implique une méthode de travail rigoureuse, que nous allons étudier en l'illustrant d'un exemple de tri par insertion (algorithme) avec constitution d'un fichier « parallèle » (recopie). On se reportera au schéma de la page 277.

1 / Après avoir lu le premier enregistrement de données (qui contient B), on le transfère dans le premier enregistrement du fichier de sortie, c'est-à-dire du fichier parallèle qui

Détail d'un ancien dispositif de mémoire centrale à tores de ferrite.



Marka

Test 7

1 / Laquelle de ces affirmations s'applique à un fichier séquentiel ?
a) on peut accéder aux enregistrements dans n'importe quel ordre ;
b) les données sont écrites dans l'ordre croissant de la zone clé ;
c) les enregistrements sont écrits et lus l'un à la suite de l'autre.

2 / Lesquelles de ces affirmations s'appliquent à un fichier indexé ?
a) l'index est une zone spéciale qui contient des informations sur les enregistrements ;
b) il permet d'ordonner rapidement les données ;
c) un fichier indexé peut posséder plusieurs index.

3 / En quoi consiste un tri ?
a) à extraire les données dans l'ordre ;
b) à chercher une valeur déterminée parmi les données d'un fichier ;
c) à recopier un fichier.

4 / Enumérez les principales différences entre recherches séquentielle et dichotomique.

5 / Un fichier du personnel doit contenir : le prénom, le nom, la qualification, la catégorie, l'ancienneté, le salaire de base, les jours d'absence.
Définissez le format de l'enregistrement en choisissant des longueurs pour les différentes zones.

6 / Définissez la structure des enregistrements d'un fichier de gestion des stocks qui doit contenir les informations suivantes : article, quantité en stock, cote d'alerte, date du dernier prélèvement, nom et prénom du fournisseur.

7 / Pour la question 6, on peut gagner de la place en codant certaines zones. Essayez de coder la date de prélèvement (jour, mois, année) et le nom du fournisseur (on supposera qu'il y a 100 fournisseurs au maximum).

Les solutions du test se trouvent pages 286 et 287.

contiendra les données ordonnées.

2 / Le deuxième enregistrement (qui contient L) est transféré en deuxième position du fichier de sortie puisque son contenu est supérieur à celui du premier (L > B).

3 / L'enregistrement 3 du fichier de données contient A, c'est-à-dire une donnée inférieure à celle qui sont déjà présentes dans le fichier de sortie (B et L). Il faut donc le mettre en position 1.

3-1 / La position 1 étant déjà occupée par la lettre B, il faut la libérer en décalant le contenu du fichier de sortie d'une position vers la droite. Ainsi, la lettre L passe de la position 2 à la position 3, tandis que B vient en position 2.

L'enregistrement 1 est disponible pour A.

4 / La lettre O (4e donnée) est placée en dernière position puisqu'elle est supérieure à toutes les autres lettres du fichier de sortie.

5 / Cette fois, le nouvel enregistrement à recopier (N) doit être placé avant O que l'on décale donc d'une position (étape 5-1).

6 / La lettre R (enregistrement 6 du fichier de données) est placée en dernière position.

7 / Pour insérer la lettre C (enregistrement 7), il faut décaler les quatre derniers enregistrements du fichier de sortie (L, N. O, R) afin de libérer une place entre B et L.

7-1 / Le dernier enregistrement de données (C) est inséré à cette place et le fichier de sortie est complet.

Les banques de données spécialisées

Dans un monde en constante évolution, la ***demande d'information*** *en matière économique et scientifique, tant dans le secteur privé que public, ne cesse de s'accroître.*
Or la masse de documentation publiée chaque jour dans le monde entier est telle qu'il est devenu impossible de s'en tenir régulièrement informé. A moins d'en faire un véritable métier.
Les organisateurs de bases et de banques de données rassemblent, sélectionnent et mémorisent une abondante documentation sous une forme condensée. Certaines banques de données extrêmement spécialisées ont choisi de centraliser toute la littérature qui se rapporte à un unique secteur, mais elles sont rares et les banques à caractère pluridisciplinaire sont aujourd'hui beaucoup plus nombreuses. Enfin, mentionnons les banques de données collectant des ***informations statistiques, démographiques*** *et* ***commerciales.*** *Elles sont très nombreuses et leur utilité augmente d'autant plus que l'arrivée des informations se fait à un rythme sans cesse croissant.*
Le documentaliste qui travaille dans un petit centre ne peut consulter que quelques-unes des publications susceptibles de l'intéresser. Au contraire, s'il a la possibilité d'interroger une banque de données, il obtient une liste à jour des ***références bibliographiques*** *concernant son sujet et ne laisse ainsi échapper aucune information utile.*
Il existe également des banques de données spécialisées dans la ***diffusion des brevets,*** *et qui permettent aux entreprises, même de petite taille, de se tenir au courant de l'apparition, dans leur secteur d'activité, de procédés ou de produits nouveaux et constituent soit un risque de concurrence soit, au contraire, une heureuse opportunité.*
Les banques de données favorisent également la transmission rapide des informations dans des domaines aussi variés que la médecine, le commerce international, le droit, entre autres.
Le marché de l'information est particulièrement développé dans les pays dont la structure de production privilégie les industries de transformation, et qui sont constamment éperonnés par les progrès accomplis à l'extérieur, tant sur le marché des facteurs de production (nouvelles technologies) que sur celui des produits finis. Une telle économie devrait mobiliser toutes ses ressources dans l'innovation technologique et commerciale afin de conserver et (si possible) d'améliorer sa position. Les premières entreprises qui se sont intéressé à la ***« documentation en ligne »,*** *sont apparues aux Etats-Unis, pays où des systèmes très performants de traitement des données pouvaient s'appuyer sur le réseau de télécommunications le plus moderne du monde. Lorsque, par la suite, ces services se sont développés et que le nombre des utilisateurs potentiels s'est accru, les sociétés américaines ont commencé à proposer leurs fichiers de l'autre côté de l'océan. Entre-temps, dans tous les pays à haut niveau technologique, de telles entreprises d'information ont été créées en Europe sur le modèle américain, proposant des fichiers comparables, ou revêtant (INSEE) un caractère national et rassemblant des données originales.*
Pour permettre à ce patrimoine de se développer sur un plan européen et pour acquérir leur autonomie technologique, les services de Postes et Télécommunications de la CEE ont créé ensemble le premier réseau européen, EURONET. Il est le fruit d'un accord aux termes duquel la libre concurrence est établie entre toutes les banques de données européennes. Les utilisateurs y accèdent, en effet, de façon indifférenciée. Ainsi, l'ouverture des frontières, souvent illusoire dans le domaine des échanges commerciaux, se réalise dans celui des télécommunications.
Opérationnel depuis 1980, EURONET offre le support des télécommunications aux banques de données européennes réunies sous le label DIANE (Direct Information Access Network in Europe). Son caractère d'internationalité est ce qui fait la particularité de ce réseau.
L'utilisateur qui désire interroger un fichier situé dans l'un des pays membres souscrit un contrat d'utilisation d'EURONET et doit disposer d'un terminal connectable au réseau téléphonique. Le raccordement au réseau est ensuite pris en charge automatiquement par les ordinateurs qui constituent les différents

« nœuds » (points d'accès).
Il suffit à l'utilisateur européen de téléphoner au nœud d'une grande ville de son pays pour pouvoir, sans autre frais que celui d'une communication téléphonique interurbaine, se relier aux banques de données d'Allemagne, de France, de Grande-Bretagne, de Hollande, d'Italie, de Belgique ou du Luxembourg. La distance ne retentit pas sur le prix, et l'utilisateur n'a à payer que son appel. De plus, les tarifs d'EURONET sont très souvent inférieurs aux tarifs américains.
Toutefois, il est à peine plus compliqué de consulter les banques de données situées aux Etats-Unis. Il suffit de s'abonner au réseau de télécommunications assurant le lien entre le pays de départ et les nombreux réseaux américains (networks). A ces frais assez peu élevés s'ajoutent ceux du contrat avec le ou les réseaux américains qui, d'ailleurs, comme la plupart des banques de données reliées à EURONET, ne facturent que l'utilisation effective et n'imposent pas de minimum.
Les fichiers accessibles par ces réseaux ont des contenus, et donc des utilisations, extrêmement variés. Il faut établir une distinction fondamentale entre banques de données et bases de données, en fonction de leur contenu.

Les **bases de données** sont des fichiers organisés que l'on peut interroger à partir d'un terminal et qui contiennent ce qu'on appelle des publications secondaires. Il s'agit de comptes rendus synthétiques de revues, de monographies, de livres, d'exposés de conférences, destinés à fournir à des usagers très différents un moyen de sélection efficace des documents traitant d'un sujet précis.
Elles sont souvent établies par des organismes (associations liées à un secteur de production, institutions universitaires ou encore entreprises privées spécialisées dans le traitement de l'information) qui éditent déjà des résumés de revues ou d'articles. Parfois les fichiers couvrent alors plusieurs disciplines.
Pour chaque article, citation, ou monographie recensés, on indique : le titre (en anglais et dans la langue d'origine si elle est différente de l'anglais) ; l'auteur ; la source (revue, article, monographie) et tous les autres éléments qui permettent d'identifier l'original ; une liste de descripteurs ou mots clé, tirés du texte et permettant d'en préciser la portée et de s'assurer qu'il concerne la recherche menée ; et enfin, le résumé, à la fois concis et complet, du document.
Le résultat final d'une recherche parmi ces fichiers fournira l'ensemble des références

L'exploitation des banques de données est particulièrement utile dans le domaine de la recherche scientifique. Ainsi, pour les travaux menés à l'Observatoire de Greenbank, en Virginie (Etats-Unis).

Marka

bibliographiques que le documentaliste aura pu sélectionner en combinant un certain nombre d'éléments qui définissent de façon univoque le sujet choisi.

Ce service s'avère très utile aux personnes éloignées de la source d'information principale, en leur épargnant la difficulté de choisir les publications nécessaires à leur travail. C'est notamment le cas des gens qui ne disposent pas de bibliothèque à proximité de chez eux, ou qui ont besoin, pour une utilisation particulière, d'une revue qu'ils ne parviennent pas à trouver parmi celles qui sont à leur disposition.

Avec les bases de données, ils disposent ainsi d'un outil de sélection rapide des documents en rapport avec leur sujet. Pour préparer l'interrogation d'une base de données, on sélectionne les fichiers spécialisés qui semblent correspondre le mieux au thème de la recherche, et également certains fichiers pluridisciplinaires susceptibles d'être intéressants.

La ***stratégie de la consultation*** *découle de la nature des fichiers interrogés. Si l'on craint de recueillir un trop grand nombre de références par une approche trop générale, on peut restreindre le champ de la recherche en définissant des conditions plus précises (date de mise à jour, type de source, pays d'origine, auteur, etc.). Les instructions de recherche contiennent la suite logiquement ordonnée des mots que le documentaliste s'attend à trouver dans le titre ou dans le texte et qui spécifient les caractéristiques du sujet recherché.*

Voici un bref exemple de stratégie de recherche avec ses instructions d'extraction et l'impression de l'une des références bibliographiques trouvées.

Le but de la recherche, menée sur la base de données PAPERCHEM, développée par The Institute of Paper Chemistry et accessible par le réseau américain DARDO, était de trouver le compte rendu d'une conférence intitulée Energy from Biomass Conference.

L'abréviation CT (conference title) utilisée dans cet exemple est un opérateur qui désigne un certain type de documents et que PAPERCHEM connaît.

Voici les instructions de consultation :

SELECT CT = ENERGY (1W) CT = BIOMASS, ce qui signifie : « Sélectionne, parmi les titres de conférence, le mot ENERGY ; tu dois trouver le mot BIOMASS dans la même zone ». L'opérateur (1W) précise que les deux mots doivent se trouver dans l'ordre indiqué (Biomass doit venir après Energy dans le titre) et qu'ils peuvent être reliés par un troisième mot.

?SELECT CT = ENERGY (1W) CT = BIOMASS
33 83 CT = ENERGY (1W) CT = BIOMASS

L'impression de la référence trouvée permet d'obtenir :

1. le titre du compte rendu ; 2. son auteur ; 3. le titre de la conférence (c'est la zone dans laquelle la recherche a été effectuée) ; 4. les pages du document d'origine qui contiennent ce compte rendu ; 5. la date de la conférence ; 6. le nombre total de comptes rendus ; 7. la classification du document ; 8. la langue d'origine ; 9. le résumé ; 10. les descripteurs.

Quant aux ***banques de données,*** *elles présentent une caractéristique fondamentale. Leur interrogation fournit directement les données finales recherchées, car les fichiers des banques de données ne contiennent pas de publications secondaires (descripteurs d'originaux) mais, essentiellement, des copies de documents originaux consultables à partir d'un terminal.*

On accède ainsi par l'écran aux publications des organismes de statistiques nationales ou des centres universitaires qui rassemblent eux-mêmes des données provenant de leurs propres recherches ou d'institutions internationales telles que la FAO ou l'OCDE.

Ces données peuvent servir à des traitements ultérieurs.

Ces fichiers sont destinés à des utilisateurs spécialisés, et associent souvent aux procédures normales d'interrogation (comparables à celles des bases de données bibliographiques), des fonctions parfois très complexes de traitement numérique des données trouvées. On peut ainsi combiner plusieurs séries numériques, insérer, traiter et conserver des données à usage confidentiel, obtenir des représentations graphiques, intercaler des données manquantes, extrapoler à partir de modèles mathématiques fournis par la banque ou mis au point par l'utilisateur.

(D'après A. Beverini. Extrait de BANCA DATI, in CELLULOSA E CARTA n° 4, juillet-août 1983.)

Les différents points exposés mettent en évidence les phases essentielles d'un programme de tri : comparaison des données et déplacements consécutifs au sein du fichier de sortie. Nous reviendrons sur ce sujet lorsque nous aurons étudié les principales instructions du Basic.
Recopier le fichier de données présente deux inconvénients : d'une part, on occupe davantage de place dans la mémoire de masse puisqu'à l'issue du tri on a deux fichiers au lieu d'un et, d'autre part, l'exécution du tri est beaucoup plus lente puisqu'il faut déplacer les données du fichier de sortie pour inscrire presque tous les enregistrements. En utilisant un index (ou fichier d'adresses), non seulement on économise la mémoire de masse, mais surtout on a la possibilité d'exécuter les déplacements en mémoire centrale (l'index ne contient que des pointeurs : il est beaucoup plus petit et l'ensemble d'un fichier index peut donc rester en mémoire centrale), ce qui est beaucoup plus rapide.

Le compactage des données

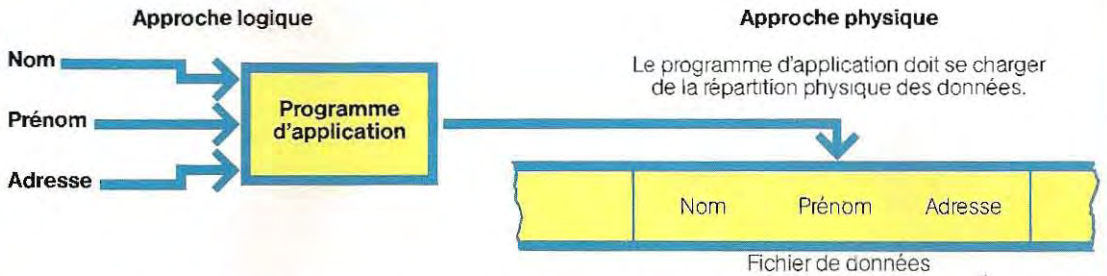
Pour certaines applications, on est obligé de condenser ou « compacter » les données avant de les inscrire sur les supports de mémoire physique.
Sur les gros systèmes qui traitent d'énormes quantités de données, le fait de gagner quelques caractères par donnée peut entraîner une réduction importante de l'espace occupé. Une meilleure exploitation des mémoires justifie alors une plus grande complexité des programmes (qui doivent compacter les données à l'écriture, puis les étendre à nouveau à la lecture). Sur les micro-ordinateurs, au contraire, ces techniques compliquées sont rarement adaptées, exception faite de quelques cas particuliers que nous étudierons par la suite.
Les principales méthodes de compactage reposent sur l'élimination des zones et des caractères vides et sur la codification de certaines combinaisons de caractères.
Les zones de données vides. Il vaut mieux les éliminer que les remplir de blancs. On obtient ainsi un enregistrement plus court que les autres. En contrepartie, on devra utiliser des programmes plus lourds, capables de traiter des enregistrements de longueur variable.
L'élimination des caractères inutiles. Dans les zones de format ASCII, on peut éliminer les blancs qui se trouvent entre deux caractères ainsi que les zéros dans les zones numériques. Une fois encore, le programme sera plus complexe car il devra être conçu de façon à retrouver combien d'espaces ou de zéros ont été éliminés afin de pouvoir reconstituer les données sous leur forme initiale avant de les présenter en sortie.
Le codage des combinaisons de caractères. Lorsque des combinaisons de caractères reviennent souvent, on peut les représenter par un symbole. Ainsi, dans le fichier du personnel, nous trouvons les noms Rousseau, Lenoir et Leblanc.
En associant, par exemple, « ousseau » à 1 et « blanc » à 2, les données deviennent :
R1, Lenoir et Le2.
Cette technique n'est intéressante que si les séquences ainsi codifiées reviennent fréquemment.

Les banques de données des gros systèmes

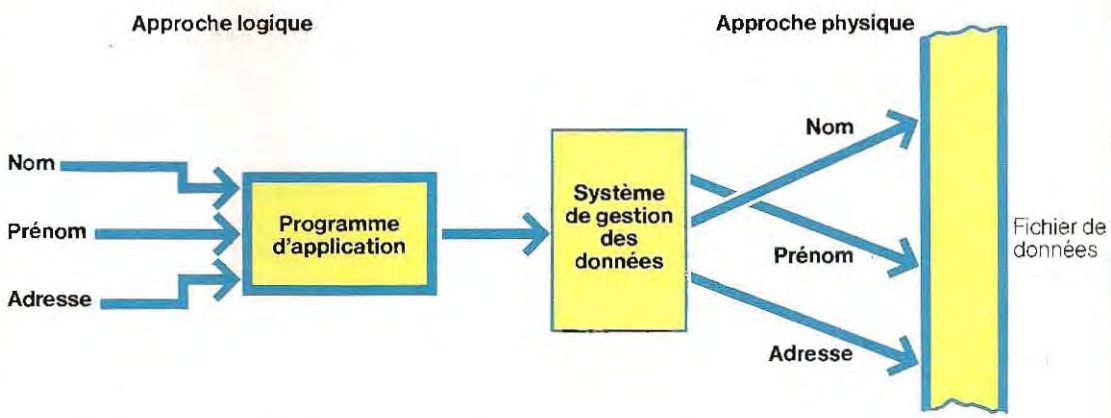
Les gros ordinateurs et certains mini-ordinateurs possèdent des programmes de gestion de données très complexes.
Grâce à ces programmes l'utilisateur n'a plus à s'occuper de la structure physique des fichiers mais simplement à définir leur structure logique.
Les programmes-système se chargeront ensuite de la traduire en schémas physiques et de gérer fichiers, zones et enregistrements. Le programmeur ne connaît plus l'adresse physique des données puisque l'espace mémoire disponible (disques, tambours) est géré par le système. Les schémas de la page 284 illustrent les différences entre ces deux méthodes.
Dans le premier cas (pas de programme de gestion des données), le programmeur doit définir la structure logique de son fichier (ici, le nom, le prénom et l'adresse), la structure physique des enregistrements et leur position dans le fichier.
Dans le second cas, il n'a que la structure logique à définir. C'est le système qui décide où et comment écrire les données, à l'aide des programmes de gestion.
Sur les systèmes les plus évolués, la gestion des données se fait parfois au moyen d'un

langage spécialement étudié, différent des langages d'application.
Dans ce cas, le langage de gestion des données est lié à un langage normal de programmation (Basic, Cobol, etc.) et l'utilisateur peut se servir indifféremment des instructions des deux langages.
L'importance de cette question a été à l'origine de la création à Washington, en 1959, du CODASYL (Conference on Data System Language), organisme chargé d'unifier les recherches effectuées dans le monde entier et de publier des normes à observer dans le développement des systèmes de bases de données (structures logiques et relations entre les données).

GESTION DES DONNEES ET PROGRAMMES D'APPLICATION

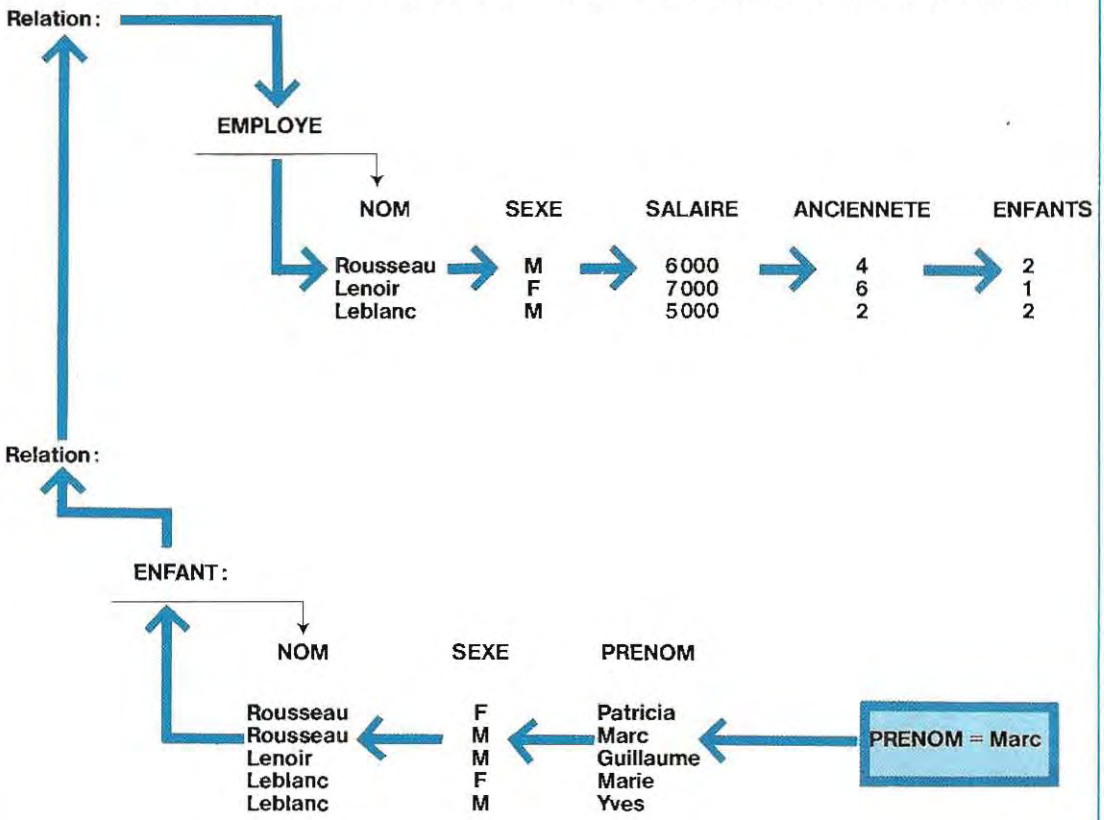


En l'absence de système de gestion des données évolué, les programmes d'application doivent tout prendre en charge, y compris l'adressage des données dans les mémoires de masse.



Sur les gros systèmes, des programmes de gestion des données simplifient beaucoup le travail de programmation. Le système prenant totalement en charge leur gestion physique, l'utilisateur n'a plus à définir que la structure logique de ses données.

EXEMPLE DE RECHERCHE DANS UNE BASE DE DONNEES RELATIONNELLE



Le « domaine » PRENOM (PRENOM = Marc), à travers les relations ENFANT et EMPLOYE, permet de sélectionner toutes les données de l'employé Martin.

Les bases de données relationnelles

Elles permettent l'indépendance totale des structures logique et physique des fichiers. Dans une base de données relationnelle, il n'est défini qu'une structure logique et toutes les données sont reliées entre elles par des « relations ». On peut, en partant de n'importe quelle donnée ou « domaine » (qui constitue la clé de la relation), accéder à toutes les autres données associées, au moyen des relations qui existent entre elles.

Le schéma ci-dessus illustre l'exemple d'un fichier du personnel où coexistent deux relations : Employé et Enfant. A partir d'une valeur (Marc) du domaine PRENOM, on obtient toutes les informations sur l'employé.

Cette base de données, qui est la forme la plus évoluée de gestion des données, n'est proposée que sur de gros ordinateurs.

Solutions du test 7

1 / c) les enregistrements ne peuvent être ni lus, ni écrits sélectivement dans un fichier séquentiel. Les données sont enregistrées séquentiellement, c'est-à-dire dans l'ordre de leur introduction.

---

2 / b et c) l'index est une zone du fichier (ou un fichier à part) où sont enregistrés les clés et les pointeurs vers les enregistrements de données. Il peut, en général, y avoir plusieurs index, correspondant chacun à une clé d'accès différente.
Dans un fichier du personnel qui contient les informations suivantes :
– prénom et nom – qualification, catégorie, service – salaire de base, taux horaire des heures supplémentaires – jours de congé, d'absence, de maladie –
il peut être utile de disposer de plusieurs index : un premier pour les noms (qui constituent la clé principale), d'autres pour les données secondaires, de façon à pouvoir accéder aux enregistrements selon un autre critère que le nom. Ainsi, on peut, grâce à l'index des jours de congé, obtenir la visualisation de tous les employés qui devront bénéficier d'une certaine période de vacances, ou encore se servir de l'index des services pour établir la liste du personnel d'un même service. Sur les systèmes les plus évolués, on peut procéder à une recherche « croisée », qui combine les zones clé à l'aide d'opérateurs logiques. On pourra ainsi, par exemple, sélectionner les employés du service d'entretien qui ont droit à 20 jours de congé et n'appartiennent pas à la catégorie A. D'un point de vue logique, cette instruction correspond à la formule suivante : (service = entretien) ET (congés = 20) ET (catégorie ≠ A). Le symbole ≠ signifie « différent de » (en Basic, on écrira « catégorie < > A »).

---

3 / a) c'est ainsi qu'on appelle l'opération de classement. Rappelons que le tri d'un fichier consiste à extraire les données dans l'ordre, selon une zone définie comme clé. Dans l'exemple précédent (**2**/), l'existence de plusieurs index (nom, congés, service) permet d'obtenir des listes du personnel ordonnées différemment (autant qu'il y a de zones clé). On peut également combiner plusieurs conditions au cours d'un même tri. On pourra ainsi demander une liste du personnel par service, où les noms des employés apparaîtront par ordre alphabétique à l'intérieur de chaque service.

---

4 / La recherche séquentielle consiste à lire tous les enregistrements d'un fichier jusqu'à la découverte de celui que l'on cherche. La recherche dichotomique ne lit que les enregistrements obtenus par divisions binaires successives, ce qui va beaucoup plus vite. On ne peut l'utiliser que sur des fichiers ordonnés, ce qui nécessite souvent un tri préalable. Lorsqu'un fichier contient peu de données, le tri et la recherche prennent autant de temps que la recherche séquentielle.

---

5 / Cette solution n'a qu'une valeur indicative puisque le résultat dépend des longueurs des zones utilisées. Une moyenne raisonnable donnerait :

prénom et nom	50 caractères
qualification	2 caractères (code de deux lettres)
catégorie	2 caractères (code)
ancienneté	2 caractères (de 0 à 99 ans)
salaire	5 caractères (jusqu'à 99 999 francs)
jours d'absence	2 caractères (jusqu'à 99)
TOTAL	63 caractères

On utiliserait donc ainsi 63 caractères. En adoptant une longueur de 80 caractères, on disposera d'une marge de 17 caractères pour d'éventuels ajouts.
Pour avoir un ordre de grandeur de la capacité de stockage d'une disquette, calculons le nombre d'employés dont on pourrait stocker les données sur des disquettes de 250 Ko et d'1 Mo (1 Ko = 1 000 octets ; 1 Mo = 1 méga-octet = 1 000 Ko).
La disquette de 250 Ko contient 250 000 caractères, ce qui correspond à :

$\frac{250000}{80}$ = 3125 données de 80 caractères.

La disquette d'1 Mo contient 1 000 000 de caractères, ce qui correspond à :

$\frac{1000000}{80}$ = 12 500 données.

Une seule disquette peut donc contenir le fichier du personnel d'une entreprise de 12 000 employés.
Dans notre exemple, des données numériques comme le salaire ont été considérées comme des chaînes de caractères. En effet, les systèmes d'exploitation les plus répandus en Basic représentent en mémoire de masse les valeurs numériques sous la forme de caractères. Certaines machines utilisent cependant le format numérique où les nombres sont traités comme tels. On économise alors de la place car le format numérique binaire en occupe moins que le format caractère (ASCII). Ainsi, en binaire le nombre 60 n'occupe qu'un octet (60 en base 10 = 00111100 en base 2), alors qu'en code ASCII, il lui faudrait deux octets, un pour chaque caractère (6 et 0, représentés respectivement par 0110110 et 0110000).

---

6 / Cet exercice est semblable au précédent. Voici un exemple de solution :

article	20 caractères (désignation)
quantité	4 caractères (jusqu'à 9 999)
cote d'alerte	4 caractères
date	8 caractères (jour = 2 caractères, mois = 2, année = 4 ; ex. : 21 071983)
fournisseur	20 caractères
TOTAL	56 caractères, ce qui donne 70 si l'on prévoit une marge raisonnable.

---

7 / Il y a deux façons très simples de gagner de la place dans le fichier précédent. On peut déjà coder la date par les deux derniers chiffres de l'année : 83 au lieu de 1983. On peut également désigner les fournisseurs par des numéros : Rousseau = 0, Lenoir = 1, etc. On établit ainsi une codification des 100 fournisseurs à l'aide des nombres compris entre 0 et 99. Ces deux modifications permettent d'économiser 2 caractères pour la date et 18 pour le fournisseur. Les données n'occupent plus que 36 caractères et l'on pourra donc adopter une longueur de 50 caractères au total pour conserver une marge. On pourra ainsi faire tenir 5 000 enregistrements de ce fichier sur les petites disquettes et 20 000 sur les plus grandes.

GLOSSAIRE

Les mots clés de la gestion des fichiers

Accès. Opération de lecture ou d'écriture des données sur les mémoires de masse (disques durs ou souples, etc.).

Adressage indirect. Il permet de repérer une donnée à l'aide d'un fichier index contenant des pointeurs. La recherche se fait en deux temps : le premier pointeur désigne un enregistrement de l'index qui, lui-même, contient l'adresse de la donnée.

Adresse. Valeur numérique indiquant la position d'un enregistrement.

Algorithme. Méthode de calcul utilisée pour la résolution d'un problème donné.

Banque de données. Ensemble de données relatives à un certain nombre de sujets et organisées en vue de leur utilisation par des programmes correspondant à des applications distinctes. Elles présentent une caractéristique fondamentale : les informations que l'on obtient quand on les interroge d'un terminal correspondent aux données finales recherchées. En effet, ces banques ne contiennent pas de publications secondaires mais des copies de documents originaux.

Base de données. Ensemble de fichiers organisé selon un même schéma logique. On distingue bases et banques de données, même si les deux termes sont parfois confondus. Les bases de données désignent, notamment, les fichiers où sont enregistrées les publications secondaires. On peut les interroger à partir d'un terminal. Ce sont des résumés normalisés et concis de revues, livres, monographies ou rapports de conférences, destinés à fournir à des usagers très divers un outil de sélection efficace de documents en rapport avec leur recherche.

Buffer. Voir **Zone tampon**.

Chaînage. Les enregistrements sont dits chaînés lorsque chacun d'eux contient un pointeur qui désigne l'enregistrement suivant. Cette technique, utilisée notamment dans les banques de données, permet de former une liste dans laquelle chaque enregistrement est chaîné à son successeur, quelle que soit sa position physique en mémoire. Le pointeur qui sert au chaînage constitue alors une zone supplémentaire de chaque enregistrement.

Clé. Zone d'un enregistrement permettant de le rechercher.

Compactage (ou condensation). Technique qui permet d'écrire une donnée sur un nombre plus réduit de caractères

DAM (Direct Access Method). Méthode d'accès direct.

Data base. Voir **Base de données**.

Dictionnaire. Liste des noms ou étiquettes de données prévus dans le cadre d'un fichier ou d'un programme.

Fichier. Espace physique et logique destiné à rassembler un ensemble de données homogènes.

Fusion. Ce terme désigne l'opération (et, par extension, le programme correspondant) qui consiste à réunir les données contenues dans des fichiers différents.

ISAM (Indexed Sequential Access Method). Méthode d'accès séquentiel indexé.

Liste. Voir **Chaînage**.

Mémoire virtuelle. Des blocs entiers du programme ne pouvant tenir simultanément en mémoire centrale sont transférés vers la mémoire de masse, puis ramenés en mémoire centrale lorsqu'ils sont nécessaires à l'exécution. Ces opérations, effectuées par le système, sont extrêmement rapides et permettent d'augmenter artificiellement la taille de la mémoire centrale.

Méthodes d'accès. Méthodes de lecture et d'écriture des données. Les principales sont l'accès aléatoire (random access), l'accès séquentiel (SAM) et l'accès séquentiel indexé (ISAM).

Pointeur. Valeur numérique indiquant la position d'un enregistrement ou d'une donnée.

Random. Voir **Méthodes d'accès**.

Recherche dichotomique. Méthode de recherche des données fondée sur des divisions successives en deux parties.

Relationnelle (Base de données). Forme particulière de structuration logique des données. Les zones (ou domaines) sont liées logiquement par des relations.

Répertoire. Index des données ou des fichiers existants et de leurs caractéristiques physiques (longueur et emplacement dans la mémoire de masse).

SAM (Sequential Access Method). Méthode d'accès séquentiel.

Secteur. Division physique d'un disque. C'est la plus petite unité adressable par le système d'exploitation, lors des opérations de lecture et d'écriture.

Temps d'accès. Temps nécessaire à l'acquisition d'une donnée. Il comprend le temps de positionnement de la tête de lecture et le temps de transfert de la donnée de la mémoire de masse à l'unité centrale.

Tri. Classement d'un fichier en fonction d'une clé.

VTOC (Volume Table Of Content) ou Directory list. Liste du contenu d'un disque.

Zone. Donnée élémentaire. Un enregistrement de fichier stock comprend, par exemple, une zone de description de l'article, une zone pour le prix, etc.

Zone tampon. Zone de mémoire utilisée pour le stockage momentané des données. Sur les micro-ordinateurs, les données transitent toujours par une zone tampon au cours des échanges avec la mémoire de masse. Ainsi, lorsqu'on écrit sur une disquette, la donnée passe d'abord de la mémoire centrale à une zone tampon, avant d'être transférée sur la disquette.

Etablissement des programmes

La rédaction d'un programme est précédée par l'analyse, étape très importante, abordant les problèmes conceptuels, logiques et physiques. Lorsqu'il s'agit d'un gros système, il est indispensable de répartir cette activité entre plusieurs informaticiens, chacun d'entre eux ayant une fonction bien définie à assumer.
Sur les ordinateurs individuels et les micro-ordinateurs, c'est souvent une seule personne, l'utilisateur lui-même, qui effectue à la fois l'analyse et la programmation proprement dite.
La marche à suivre dans cette phase d'analyse est la même pour toutes les applications et peut se résumer en trois points essentiels :

1 / Définition des sorties (résultats) nécessaires.
2 / Vérification des données en entrée et des algorithmes de calcul.
3 / Choix du matériel.

Une fois ces trois points définis, on peut passer à la conception des organigrammes et à l'écriture des programmes.

Définition des sorties nécessaires

La première étape consiste à définir le type et le volume des sorties à effectuer.

1 / Editions : leur nombre, le type de données qu'elles contiennent, et la périodicité requise.
2 / Fichiers : type ; disposition et nature des données à enregistrer sur disque.

Configuration classique d'un ordinateur individuel (familial) de taille moyenne.

IRET

Cette définition précise des problèmes qui se posent permet d'établir non seulement les variables de sortie, mais aussi le type d'imprimante le mieux adapté à l'application.
On peut, en effet, en fonction de la longueur prévisible du listage et du temps nécessaire, définir la vitesse d'impression la plus appropriée (en caractères par seconde), la densité de l'impression et le format.
Pour un ordinateur individuel, le choix de l'imprimante la mieux adaptée à l'écriture des résultats d'un programme est évidemment dicté par la configuration du système. Au moment même de l'achat, il faudra définir à un niveau plus général le type de travail que l'on aura besoin d'effectuer. L'imprimante « idéale » sera celle qui permettra de concilier contraintes économiques et volume de travail à accomplir régulièrement.
Il existe sur le marché de très nombreux types d'imprimantes, mais les plus performantes d'entre elles n'intéressent pas directement le grand public. Nous nous bornerons ici à étudier les plus classiques et leurs caractéristiques fondamentales, déterminant leurs possibilités d'utilisation.

Imprimantes à aiguilles
80 colonnes (impression de 80 caractères par ligne, sur des feuilles de dimensions réduites) ;
100 caractères par seconde (soit 1,25 ligne par seconde, environ).

132 colonnes (papier grand format) ;
140 caractères par seconde (1 ligne par seconde).

132 colonnes ;
200 caractères par seconde.

Les imprimantes à aiguilles sont, en général, bidirectionnelles : leur tête d'écriture imprime les caractères, aussi bien de gauche à droite que de droite à gauche, ce qui permet de gagner du temps.

Imprimantes à marguerite
Leur vitesse d'impression varie de 20 à 50 caractères par seconde, en moyenne.

Imprimantes par ligne (ou parallèles).
Les machines de ce type écrivent plusieurs centaines de lignes (jusqu'à 2 500) par minute.

Les imprimantes à aiguilles sont les plus utilisées sur les micro-ordinateurs, et le choix s'effectuera en fonction de la rapidité d'exécution souhaitée et du format de papier adopté.
Les imprimantes à marguerite coûtent environ deux fois plus cher et ne sont donc employées que lorsqu'une impression d'excellente qualité est nécessaire.
Quant aux imprimantes par ligne, elles coûtent plus de dix fois plus cher que les imprimantes à aiguilles et leur achat n'est donc justifié que si l'on dispose au moins d'un mini-ordinateur très puissant.
La seconde étape de définition des sorties permet de fixer le volume d'informations à mémoriser sur disque. En calculant la somme des emplacements physiques nécessaires à leur stockage, on définit approximativement la taille des fichiers et donc la capacité des disques à utiliser.
Nous avons déjà vu que la capacité des disques souples varie entre 80 000 et 1 200 000 caractères tandis que celle des disques durs est de 5 000 000 d'octets (5 Mo) au minimum, et va jusqu'à 400 000 000 (400 Mo).
Le coût d'un système informatique dépend environ pour moitié du type de disque qui lui est associé. Le choix de ce support a donc des répercussions considérables et ne doit pas être fait à la légère.
Il faut toutefois tenir compte des développements futurs envisagés : opter pour une solution plus coûteuse permet de disposer de la marge de sécurité nécessaire.

Vérification des données en entrée et des algorithmes de calcul

Après avoir défini les sorties, on doit s'assurer que l'on dispose de toutes les données nécessaires en entrée, et que l'on maîtrise les algorithmes indispensables aux calculs éventuels.
Cette étape vise donc essentiellement à :

- vérifier que les données utiles sont disponibles,
- s'assurer de la nécessité de conserver sur disque certaines données,
- définir les méthodes de calcul.

Certaines données, ne servant qu'au cours

des calculs, n'apparaissent jamais sur les listages.
Mais parfois il faut tout de même les conserver sur disque et donc en tenir compte dans l'évaluation de la capacité de mémoire de masse nécessaire.

Choix du matériel

Conditionnée par le type d'application envisagé, cette dernière étape est celle qui pèsera le plus lourdement sur l'écriture des programmes.
A ce stade, on a défini les périphériques du système (imprimante et unité de disque). Toutefois, il reste encore à choisir l'unité centrale.
On distingue principalement trois grands types de micro-ordinateurs aux performances différentes :

- **les micro-ordinateurs 8 bits ;**
- **les micro-ordinateurs 16 bits ;**
- **les micro-ordinateurs 32 bits.**

Si les premiers peuvent adresser 64 000 emplacements mémoires (octets) au maximum, les deuxièmes atteignent couramment 1 000 000 (1 Mo), et jusqu'à plusieurs millions d'octets pour les plus puissants.
Le choix, parmi les deux premiers, dépend donc, entre autres, de la quantité de données à traiter.
Sur les micro-ordinateurs 8 bits, on doit, face à un important volume de données, fractionner les programmes, ce qui accroît leur complexité.
En revanche, les machines 16 bits coûtent un peu plus cher et les extensions de mémoire sont onéreuses.
L'évolution très rapide de la technologie réduit chaque année les coûts de production. De plus, les systèmes d'exploitation ne sont pas toujours en mesure d'utiliser la totalité de leur capacité d'adressage.
Le système d'exploitation constitue en effet le dernier facteur qui guide le choix. Il est préférable d'opter pour des machines munies de compilateurs et utilisant des systèmes d'exploitation largement diffusés. Ainsi, les programmes seront portables et l'on aura l'avantage de disposer d'un système d'exploitation constamment mis à jour.

Les systèmes d'exploitation

Un ordinateur est, au départ, un ensemble de circuits électroniques, capable de recevoir et de traiter des signaux électriques de type binaire. Il peut accepter, sous forme binaire, les données à traiter et les indications indispensables sur l'enchaînement des opérations à effectuer. La suite ordonnée des codes de ces différentes opérations constitue le programme.
Pour utiliser une machine de ce type, nous serions contraints d'employer le système binaire, afin d'écrire directement les codes et les données en langage machine. Même les opérations les plus simples devraient être « décrites » sous cette forme dans chaque programme.
On comprendra sans peine que travailler sur ce type de matériel exigerait des programmeurs dotés de facultés tout à fait hors du commun. Cette situation constituait pourtant la réalité quotidienne des informaticiens il y a seulement vingt-cinq ans. A cette époque, rédiger des programmes signifiait, en effet, combiner des 1 et des 0 en des suites interminables, avec une probabilité d'erreurs extrêmement élevée.
C'est pourquoi les langages symboliques ont été introduits par la suite. Ce sont les langages d'assemblage (Assembleur) et les langages évolués (Fortran, Basic, Cobol, etc.).
Le système d'exploitation représente l'ensemble des programmes de service et des compilateurs ou interpréteurs, destinés à la traduction (compilation ou interprétation) des programmes en langage symbolique.
Il convient de distinguer radicalement les systèmes d'exploitation destinés aux utilisateurs isolés (et travaillant sur micro-ordinateurs), de ceux qui permettent la multi-utilisation ou, plus précisément, l'exécution simultanée de plusieurs programmes (sur mini-ordinateurs et gros systèmes principalement).
Dans ce dernier cas, le système, plus complexe, doit « savoir » répartir les ressources disponibles (unité centrale, mémoires, disques, périphériques) entre les différents programmes qui s'exécutent en même temps et cela, à une vitesse telle qu'aucun utilisateur ne soit gêné par la présence des autres si cela est possible.

Les fonctions du système d'exploitation

Prenons l'exemple plus concret d'un système à microprocesseur de la catégorie des micro-ordinateurs et des ordinateurs individuels, et présentant par conséquent les caractéristiques d'une machine de puissance moyenne :

- un seul utilisateur ;
- mémoire utilisateur de 64 000 octets ;
- unité d'affichage 24 lignes × 80 colonnes ;
- deux unités de disques souples ;
- un clavier ;
- une imprimante.

Supposons que la programmation soit prévue en langage Basic. La première condition que doit satisfaire le système d'exploitation est évidente : il doit être capable de traduire le Basic en langage machine. Un premier choix s'impose entre deux types de systèmes d'exploitation : ceux qui ne contiennent qu'un interpréteur et ceux qui possèdent également un compilateur. L'exécution des programmes est généralement plus rapide sur les systèmes équipés d'un compilateur.
L'analyse de la configuration du matériel détermine les autres fonctions du système d'exploitation. L'unité d'affichage, le clavier et l'imprimante sont des dispositifs d'entrée-sortie, que les programmes d'application doivent pouvoir utiliser en précisant simplement le périphérique concerné et les données à lire ou à écrire. Le système d'exploitation devra se charger de la gestion des fonctions complexes régissant le matériel, c'est-à-dire des mécanismes d'impression et de reconnaissance des touches frappées au clavier.
L'utilisation de disquettes suppose également un contrôle exercé par les programmes du système.
Enfin, il doit nécessairement pouvoir interpréter des instructions particulières entrées au clavier, tels les ordres de mise en marche ou d'interruption de l'exécution d'un programme.
Un système d'exploitation se compose donc, en général, de trois parties principales :

1 / le module de traitement des commandes ;
2 / le module de gestion des unités de disque ;
3 / le système de gestion des opérations d'E/S.

Nous allons expliquer les fonctions correspondant à chacune de ces parties et leur utilisation.

Le module de traitement des commandes. C'est le premier sollicité lorsqu'on branche la machine. La configuration d'un micro-ordinateur comprend souvent deux unités de disque. La première, en général désignée par la lettre A ou le chiffre 0, doit contenir la disquette du système d'exploitation. La mise sous tension déclenche l'exécution d'un programme d'amorçage (bootstrap), qui réside en permanence dans la mémoire. Le seul rôle de ce programme consiste à lire le système d'exploitation sur la « disquette système » (unité A ou 0) pour le transférer en mémoire. Après ce « chargement » du système, la machine peut interpréter et exécuter des instructions. Elle signale à l'opérateur qu'elle est prête à recevoir des instructions par un tableau (le plus souvent) qui résume les principales caractéristiques du matériel*, ou par un symbole d'« état prêt » (prompt). Les illustrations des pages 293 et 294 montrent cette phase d'initialisation sur les écrans de deux machines de type différent. On remarquera, dans les deux cas, l'apparition du symbole de « système prêt » à côté du curseur.
A ce stade, on ne peut donner que des instructions propres au système d'exploitation, commandant des opérations de base indépendantes du langage que l'on veut utiliser.

On a donc deux possibilités :
- demander l'exécution d'opérations de base ;
- entrer un programme en langage symbolique.

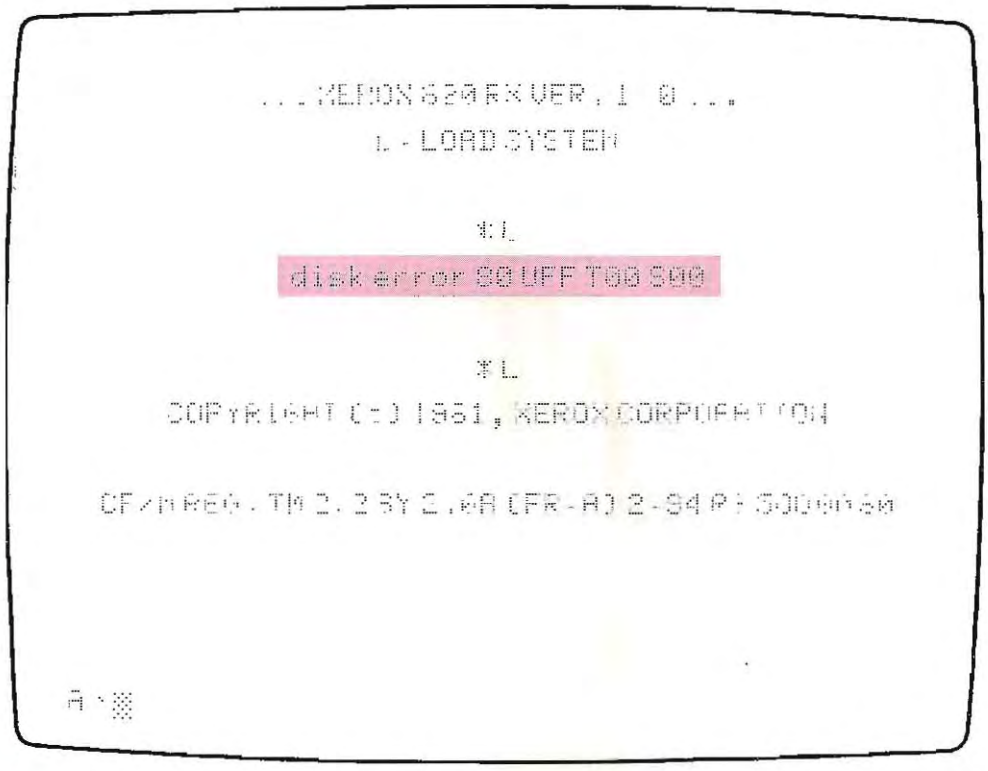
Pour écrire un programme en Basic, il faut communiquer (par ex.) l'instruction MBASIC à la première machine (Xerox 820) et l'instruction BA à la seconde (M20 Olivetti).

---

*Les caractéristiques du matériel (hardware), c'est-à-dire la capacité de la mémoire, le nombre des unités de disque, etc., définissent ce qu'on appelle la configuration du système. Le tableau résumant les caractéristiques de notre machine en est un exemple. On parle également de la configuration du logiciel (software) d'un système, pour désigner l'ensemble des programmes qui sont à la disposition de ses utilisateurs.

ECRAN DU SYSTEME XEROX 820 LORS DE SA MISE SOUS TENSION

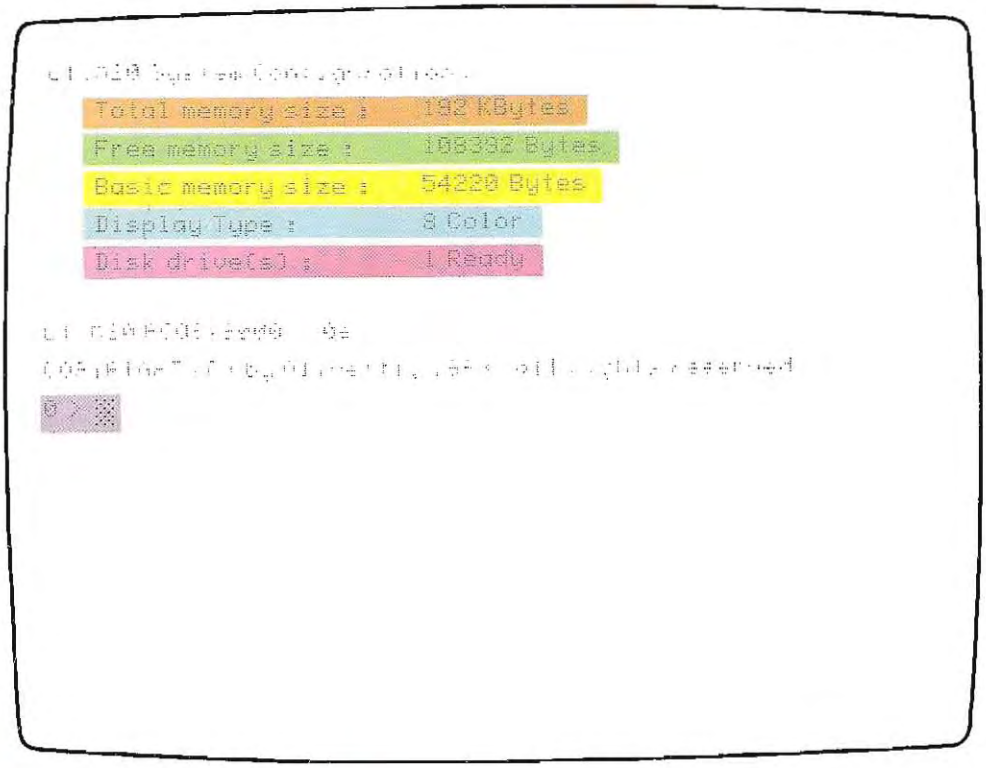
Le Xerox 820 utilise le système d'exploitation CP/M qui réside sur une disquette et n'est accessible qu'après son chargement en mémoire. Lors de la mise sous tension, on a délibérément laissé l'unité de disque A (celle de gauche) ouverte, unité où doit normalement se trouver la disquette système.



Le programme d'amorçage (ou d'initilisation), qui réside en permanence en mémoire ROM, essaie de lire le CP/M, mais l'unité de disque n'est pas préparée. Il émet donc un message d'avertissement, qui mentionne le disque de gauche (puisque c'est là que devrait se trouver le système d'exploitation).
Sur certaines machines, on peut inverser les unités de disque en modifiant une connexion électrique, ce qui permet de charger le système même si l'unité A est en panne.
Une fois le disque système inséré dans le lecteur de gauche, le système d'amorçage chargera le CP/M en mémoire vive.
L'utilisateur pourra choisir entre le chargement de son programme ou l'envoi de commandes directes.

ECRAN DU SYSTEME M20 OLIVETTI LORS DE SA MISE SOUS TENSION

Les six premières lignes donnent à l'utilisateur des informations sur la configuration du système.



La capacité totale de la mémoire est de 192 Kbytes (Koctets).

La mémoire disponible pour l'utilisateur est de 108 392 octets.

La zone mémoire réservée aux programmes en Basic est de 54 220 octets.

L'écran peut afficher 8 couleurs différentes.

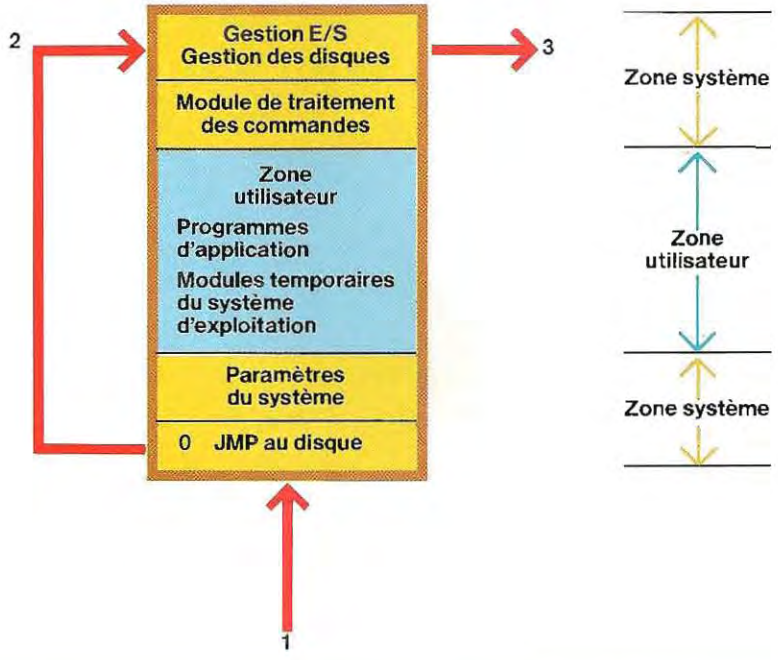
Une seule unité de disque est disponible «en ligne» (connectée). Cela signifie qu'une seule unité est opérationnelle mais la machine peut néanmoins être équipée d'un second lecteur. C'est le cas du M20. Mais, au moment où la photo a été prise, l'une des deux (numéro 1) unités était ouverte.

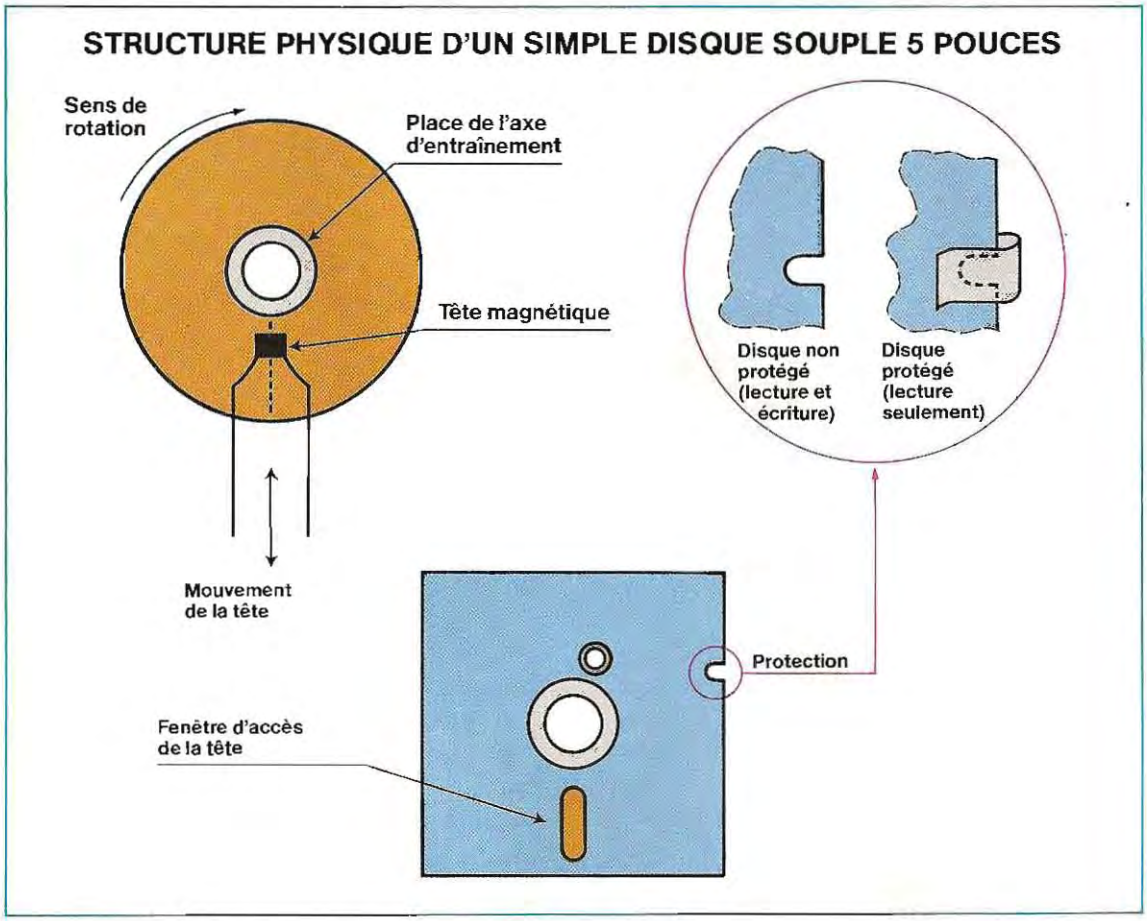
Les deux lignes suivantes indiquent le nom du système d'exploitation (PCOS), sa version ainsi que sa date de création.

Le «prompt» («système prêt») apparaît sur la dernière ligne à côté du curseur. Les ordres transmis au système d'exploitation viendront s'écrire à partir de cette position.

Le module du système d'exploitation chargé de l'interprétation des commandes identifie cette instruction qui demande le passage en Basic. Il charge l'interpréteur en mémoire (en général, le Basic est seulement interprété dans un premier temps ; ce n'est qu'ensuite qu'il est éventuellement compilé) et se prépare à recevoir les lignes du programme.
Si l'on veut exécuter des opérations de base, on entre leur nom symbolique, éventuellement accompagné des paramètres nécessaires.
Après avoir analysé l'instruction, le module de traitement des commandes exécute l'opération demandée.
Les commandes et, donc, les modules correspondants du système d'exploitation, peuvent être résidents ou temporaires.
Les modules résidents, présents en permanence dans la mémoire dès le chargement du système d'exploitation, sont exécutables immédiatement.
Les modules temporaires ne sont chargés que sur demande, et contiennent des commandes propres à l'utilisateur, et stockées dans la partie de la mémoire appelée zone utilisateur. C'est également dans la zone utilisateur que sont chargés les programmes d'application avant leur exécution.
Le schéma ci-dessous montre un exemple de répartition de l'espace mémoire (memory map).
Le premier emplacement de mémoire est indiqué par le symbole 0, et contient une instruction de saut au module de gestion du disque (JMP est l'abréviation de jump, signifiant saut en anglais).
La mise sous tension remettant à 0 le compteur d'instructions (ou compteur ordinal), l'instruction de saut située à l'emplacement 0 (étape **1**) est la première exécutée. Elle entraîne l'exécution du module de chargement du système (gestion des disques, étape **2**) ; la machine attend ensuite les instructions de l'opérateur (étape **3**).
On peut également voir, sur le schéma ci-dessous, illustrant l'espace mémoire, une zone réservée aux paramètres du système (pointeurs, adresses, etc.).

EXEMPLE DE REPARTITION DE L'ESPACE MEMOIRE





La gestion des fichiers sur disque souple. Avant de poursuivre l'étude des opérations exécutées par un système d'exploitation, il nous faut revenir de façon plus détaillée sur la structure physique et logique des disquettes (floppy disks).

Les disquettes sont constituées d'un support souple recouvert d'une fine couche de substance magnétique. L'état magnétisé correspond à la valeur 1 et l'état non magnétisé à la valeur 0.

Lecture et écriture sont effectuées par une tête magnétique qui se déplace le long d'un rayon de la disquette tandis que la rotation de celle-ci permet à tous les points de sa surface de se trouver tour à tour au-dessous de la tête.

Les disquettes sont protégées par une enveloppe percée d'une fenêtre, permettant l'accès de la tête de lecture (voir le schéma ci-dessus).

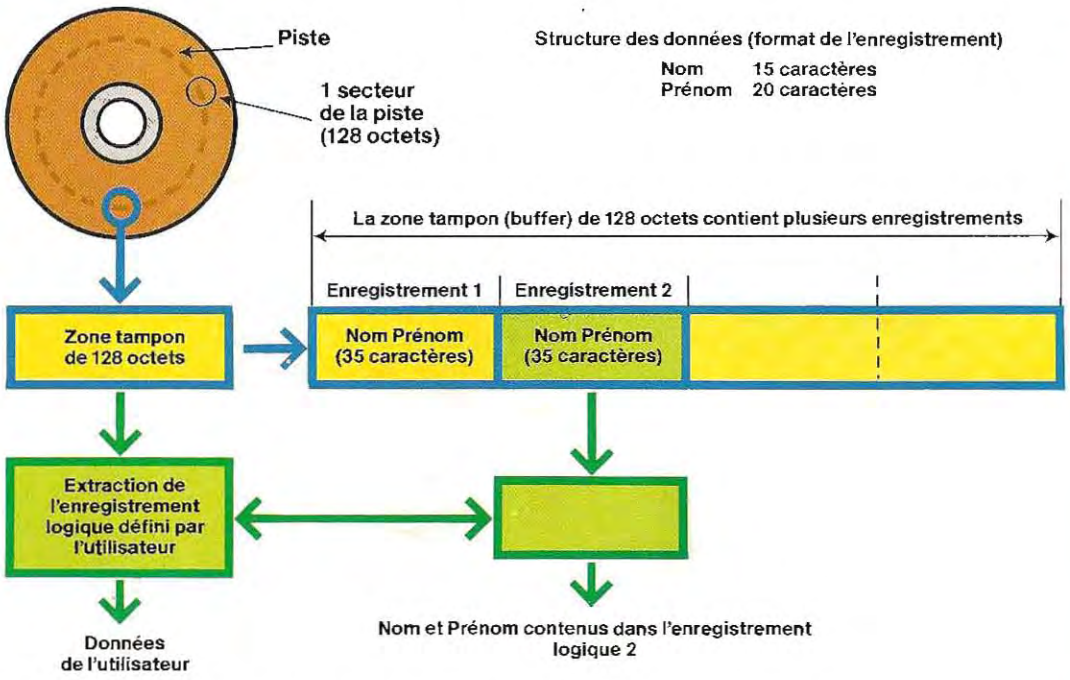
Une encoche de protection est découpée dans cette enveloppe : lorsqu'elle est libre, on peut lire mais non écrire ; si, au contraire, elle est occultée par une étiquette adhésive, l'écriture devient possible (le système de protection fonctionne en sens inverse sur certains matériels).

Les disquettes peuvent être enregistrées sur un ou deux côtés (ce sont alors des disquettes simple ou double face) et leurs zones de magnétisation sont plus ou moins rapprochées, ce qui entraîne la distinction entre disquettes à double densité (souvent désignées par DD) ou disquettes à simple densité (SD). Ce sont, bien sûr, les disquettes double face-double densité qui contiennent le plus d'informations (octets).

Un disque neuf ne peut pas être utilisé tel quel. Il faut d'abord qu'il soit « formaté » par le système d'exploitation. Le **formatage** consiste à diviser le disque en pistes, elles-mêmes divisées en secteurs. On crée ainsi une « map » des espaces disponibles, qui permet d'adresser chaque zone physique de la disquette, en indiquant la piste et le secteur où elle se trouve.

Le secteur est la plus petite unité de mémoire de masse adressable. La lecture et l'écriture s'effectuent donc par blocs dont la longueur est au moins égale à celle d'un secteur.
Un secteur contient, en général, 128 ou 256 octets et les transferts entre la disquette et la mémoire se font toujours par blocs de cette longueur.
C'est seulement ensuite que seront extraits

MECANISME DE LECTURE D'UN ENREGISTREMENT LOGIQUE



Le système prélève sur le disque un secteur entier dont il extrait ensuite les données (enregistrement logique).

les caractères réellement nécessaires au programme d'application (voir le schéma ci-dessus).
La structure d'une disquette, c'est-à-dire le nombre de pistes qu'elle contient, le nombre de secteurs par piste et le nombre d'octets par secteur, dépend à la fois de son type (diamètre, densité, simple ou double face) et du formatage effectué par le système. Voici un exemple de caractéristiques assez courantes :

- 2 faces désignées l'une par 0 et l'autre par 1 (double face) ;
- 35 pistes par face ;

- 16 secteurs par piste ;
- 128 octets par secteur (1D).

La numérotation des secteurs n'est, généralement, pas séquentielle mais intercalée (interleaved). Ainsi, une piste contiendra les secteurs 1, 3, etc., tandis que les secteurs pairs se trouveront sur la piste suivante. Cette technique d'enregistrement est la plus fréquemment utilisée pour réduire les temps d'accès. Cependant, sur certains systèmes, on peut modifier l'ordre de numérotation à volonté, et sauter, par exemple, cinq numéros entre deux enregistrements (secteur 1, secteur 7, etc.). Dans certaines conditions, les temps d'accès s'en trouvent encore réduits.
Après son formatage, la disquette est prête à être utilisée comme mémoire de masse. L'enregistrement des données (ou des programmes) se fait dans des zones contiguës, constituant un fichier identifié par un nom propre. Pour relire des données, le système, connaissant ce nom, se chargera de retrouver l'emplacement physique du fichier et de transférer son contenu en mémoire.
Pour cela, les informations nécessaires, c'est-à-dire le nom, l'adresse et le type de contenu (données, programme) du fichier, ainsi que son emplacement physique (piste et secteur de début, piste et secteur de fin), sont écrites lors de la création des fichiers dans une zone réservée du disque, appelée directory (répertoire).
Le système d'exploitation peut alors positionner la tête de lecture à l'endroit correspondant au début du fichier. Il y trouvera un ensemble d'informations n'appartenant pas aux données proprement dites et constituant le FDB (File Descriptor Block), ou bloc de description de fichier.
En voici le détail pour ce qui concerne le système PCOS* (Olivetti) :

- longueur actuelle du fichier ;
- nombre d'extensions ;
- mot de passe (password) ;
- drapeau (flag) de protection ;
- table des extensions ;
- pointeur vers une extension éventuelle de la table des extensions.

Cette liste diffère légèrement d'un système à l'autre, mais les informations les plus importantes sont toujours présentes. Ainsi, pour le système CP/M*, les principales rubriques sont, dans l'ordre :

- le nom du fichier (8 caractères) ;
- le type de fichier (3 caractères) ;
- le nombre d'extensions (extent) ;
- la taille des extensions ;
- la liste des emplacements sur le disque ;
- le numéro du prochain enregistrement en lecture et en écriture.

Un fichier n'occupe pas constamment le même espace : que ce soit un programme ou une suite de données, sa longueur augmente à chaque nouvelle saisie. Les ajouts ne sont pas placés physiquement à la suite des enregistrements préexistants dans la mesure où d'autres fichiers peuvent se trouver intercalés sur le disque.
Il faut donc mémoriser les **extensions de données** dans une zone spéciale qui contiendra une table de pointeurs (adresses) vers ces zones (ou extents). Cette table redonne une continuité logique au fichier, physiquement « éclaté », dispersé en plusieurs endroits du disque.
La lecture de la table des extensions, le positionnement à l'adresse trouvée et le chargement des données en mémoire sont pris en charge par le système d'exploitation.
Lorsqu'un fichier comporte de nombreuses extensions, sa lecture demande plus d'opérations que celle d'un fichier peu morcelé, et il est donc parfois avantageux de le recopier séquentiellement sur une nouvelle disquette.
Cette opération peut être effectuée lors d'une sauvegarde du fichier sur un disque fraîchement formaté.

On peut conserver sur disque plusieurs types de fichiers dont les principaux sont :

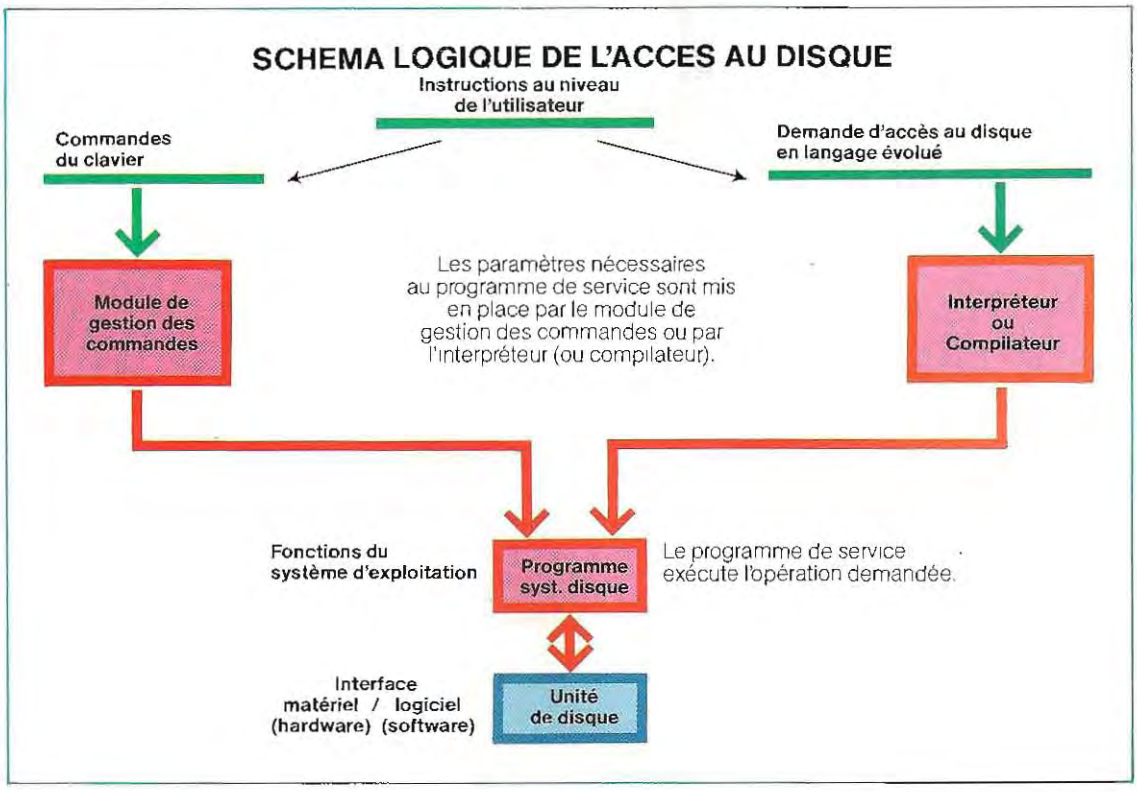
1 / les programmes sources en Assembleur (ASM) ;
2 / les programmes en langage machine avec représentation hexadécimale (HEX) ;
3 / les programmes sources en Basic (BAS), ou Fortran (FOR), entre autres ;

*Les dénominations CP/M et PCOS sont des marques déposées.

4 / les programmes compilés (COM) ou programmes images ;
5 / les fichiers des modules de service ;
6 / les fichiers de données.

Nous avons indiqué pour les fichiers portant les numéros 1 à 4 les sigles de trois caractères servant généralement à les identifier. Les symboles associés aux deux autres types de fichiers varient souvent d'un système d'exploitation à l'autre.
Pour identifier un fichier, il faut indiquer son nom et son type. Par exemple, pour charger un programme en Basic sauvegardé dans un fichier appelé ESSAI, il faut indiquer ESSAI.BAS. La première partie de cette référence donne le nom du fichier (qui, en général, ne doit pas dépasser 8 caractères) et la seconde partie son type (ici programme source en Basic).
Sur certains systèmes d'exploitation, on peut remplacer l'une de ces deux informations par un symbole générique (*, par exemple). La recherche ne se limite plus alors à un nom ou à un type, et tous les fichiers qui répondent à la condition conservée sont pris en considération. Ainsi, la référence *.BAS désigne tous les programmes en Basic quel que soit leur nom. Cela est utile dans de nombreux cas. Par exemple, l'emploi du symbole générique (*.BAS) à la place du nom permet de recopier tous les programmes en Basic, en une seule fois et sans avoir à préciser leur nom. De même, si l'on écrit *.*, tous les fichiers seront concernés, indépendamment de leur nom et de leur type.

Les opérations de gestion des fichiers du système d'exploitation. Le système d'exploitation contient des modules ou programmes utilitaires (ou de service) destinés à l'exécution des principales opérations de gestion des fichiers. Chaque système d'exploitation possède son symbolisme et l'on peut donc retrouver une même opération sous des noms différents. Les dénominations que nous adopterons par la suite sont, soit les plus répandues, soit les plus proches des instructions Basic.
Pendant toute la lecture de ce paragraphe, il faudra garder présent à l'esprit le fait que l'utilisateur n'a pas directement accès aux programmes de service du système (à moins qu'il n'emploie un langage Assembleur), mais



Le dessin animé et l'ordinateur

Il y a peu de temps encore, la réalisation d'un film d'animation à partir de dessins demandait un travail colossal et prenait beaucoup de temps. C'est ainsi qu'il fallut 250 000 dessins et 300 000 heures de travail pour achever «Animal Farm», de Halas et Batchelor, l'un des premiers dessins animés en long métrage.

Aujourd'hui, l'ordinateur permet de diminuer considérablement le temps d'exécution du dessin animé, mais il met également à la disposition des dessinateurs des techniques particulières suggérant la création d'effets nouveaux.
Une caméra prenant 24 images à la seconde, le dessin animé sera obtenu en filmant 24 dessins légèrement différents l'un de l'autre, afin que leur projection à vitesse constante reproduise le mouvement dans son déroulement naturel.

Les techniques d'animation ont connu relativement peu de changements pendant longtemps, et le travail se faisait alors presque complètement à la main. Aujourd'hui, on cherche à rendre moins laborieuse l'élaboration de chaque groupe d'images en perfectionnant des procédés de traitement électronique.
En effet, même si on continue à utiliser la caméra de façon traditionnelle et à photographier des dessins faits à la main et non des images vidéo produites par ordinateur, on peut néanmoins accélérer le processus par des commandes électroniques. On installe alors directement la caméra sur une planche horizontale. On peut déplacer verticalement l'appareil, et horizontalement la table elle-même.
L'opérateur respecte, en général, un enchaînement précis des mouvements de la table ou de l'appareil selon qu'il veut obtenir des effets panoramiques, modifier la taille des personnages ou donner l'impression du mouvement.

On peut ainsi photographier les différentes positions d'un personnage en train de courir, tandis que l'on fait reculer, par à-coups calculés avec précision, le décor du fond, y gagnant ainsi en réalisme.

Lorsque ce déplacement du matériel est commandé par un ordinateur, l'enchaînement des mouvements est beaucoup plus précis, le travail bien moins long et le résultat plus vivant.
Naturellement, chaque image doit tout d'abord être traduite dans le langage de l'ordinateur avant de pouvoir être traitée.

Pour coder une image en langage informatique, il faut la découper en une mosaïque de minuscules carrés de couleur. On attribue à chaque petit carré, ou pixel (élément d'image), un numéro correspondant à sa valeur tonale. Ce numéro varie et donne, par exemple, 0 (0000 en binaire) pour le blanc et 8 (1000 en binaire) pour le noir ; les numéros intermédiaires permettent de traiter les nuances. Pour la mémoire d'un ordinateur, un dessin n'est donc qu'une suite ordonnée de chiffres. Le grand avantage de ce système est qu'on n'a plus besoin de redessiner chaque fois un personnage dans une position donnée : l'ordinateur est programmé pour projeter n'importe quelle image ou ensemble d'images stockés dans sa mémoire. Tous les dessins peuvent donc être réutilisés.

Certains ordinateurs (les systèmes DAO, dessin assisté par l'ordinateur), et notamment le Flair, laissent au dessinateur la liberté de travailler comme il en a l'habitude. Le Flair ne possède pas de clavier et il suffit, pour créer des images, de dessiner avec un stylo spécial sur une table graphique reliée à l'ordinateur. Aucune trace ne reste sur cette table, mais le dessin apparaît sur l'écran de contrôle qui sert de guide au dessinateur.

Un unique instrument remplace ainsi tout un ensemble de pinceaux ou de crayons : il peut tracer des traits fins ou des aplats. Pour changer de style, il suffira d'appuyer sur des touches de sélection spéciales disposées sur le bord de la table.

La table graphique est conçue pour communiquer à l'ordinateur les points touchés par le dessinateur : le stylet émet deux ondes électromagnétiques traversant la surface de la table en largeur et en hauteur. L'ordinateur peut ainsi calculer les coordonnées de chaque point en fonction du temps mis par les ondes pour atteindre les bords de la table.

Pour rendre les nuances de couleur, le dessinateur dispose d'une sorte de pinceau électronique commandé par une touche. Ce pinceau « vaporise » sur l'écran la couleur choisie dans la palette électronique dès que le stylet touche la table graphique. Plus le contact du stylet sur la table se prolonge, plus la couleur est intense. Ces nuances servent généralement à créer l'illusion d'une image à trois dimensions.

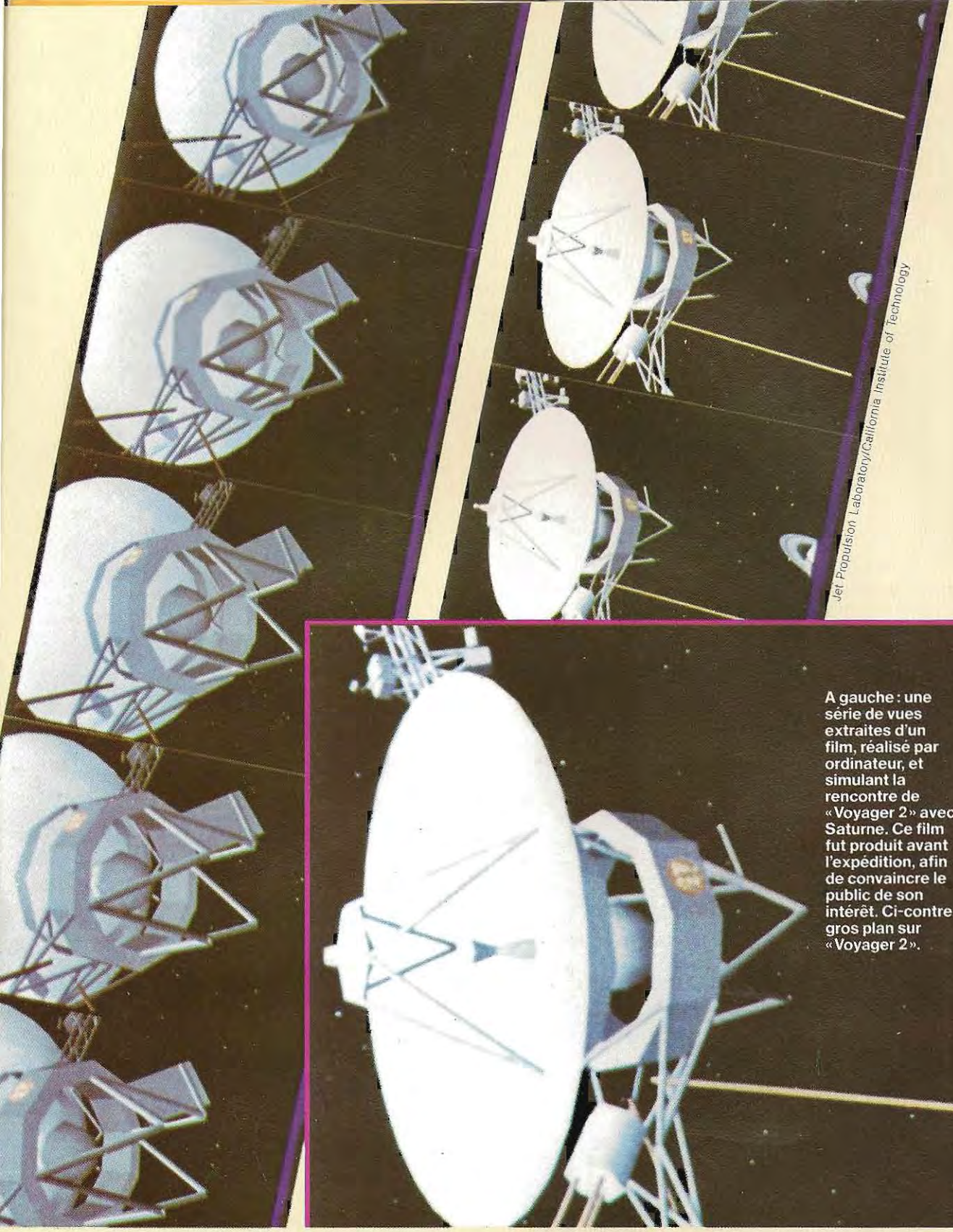
La réalisation de ces effets spéciaux demande au dessinateur une habileté comparable à celle requise par le travail traditionnel sur papier avec des pinceaux et de la peinture. Certaines opérations sont toutefois simplifiées. Pour tracer, par exemple, une ligne droite, il suffit d'indiquer ses extrémités à l'ordinateur et de le laisser relier lui-même les deux points.
Le dessinateur peut également demander à sa machine de faire apparaître sur l'écran des figures géométriques telles que cercles ou ellipses. Il suffit pour cela de fournir les points clés (centre et rayon par exemple) à la machine qui se charge de tout le reste.
Pour choisir sa couleur, le dessinateur demande l'affichage de la palette, tableau composé de carrés de couleur et appuie son stylet sur la couleur désirée pour informer l'ordinateur de son choix.

Les informations visuelles sont enregistrées dans la mémoire de l'ordinateur sous forme numérique. L'écran du Flair comporte 576 lignes de 768 points. L'ordinateur emmagasine les informations caractéristiques de chaque point sous la forme de l'une des 256 combinaisons possibles de signaux de base (rouge, vert et bleu) qui, sur l'écran, sont perçues comme une seule couleur. Lorsque, après avoir choisi une couleur, le dessinateur déplace son stylo, le code couleur correspondant est transmis à chaque point de contact.

Un seul microprocesseur, de type 8085A, contrôle l'entrée des informations et la synthèse d'image. Il est relié à la table graphique et à une unité de mémoire de disque. Ainsi,

Table graphique connectée à un micro-ordinateur. Le dessin, exécuté au crayon électronique, apparaît sur l'écran du poste de télévision.

B. Coleman/Marka



Jet Propulsion Laboratory/California Institute of Technology

A gauche : une série de vues extraites d'un film, réalisé par ordinateur, et simulant la rencontre de « Voyager 2 » avec Saturne. Ce film fut produit avant l'expédition, afin de convaincre le public de son intérêt. Ci-contre : gros plan sur « Voyager 2 ».

une fois la conception achevée, on peut conserver le dessin sur disque.
Conscients du fait que le cinéma est l'un des meilleurs supports de diffusion d'un grand nombre d'informations, la NASA et le Jet Propulsion Laboratory ont produit des films d'animation réalisés par ordinateur, et retraçant les trajectoires des sondes envoyées dans l'es-

pace par des fusées.
Ce travail requiert toute la puissance de calcul d'un gros ordinateur, mais l'exactitude mathématique des résultats permet de simuler la modification des paramètres de vol. Les films sont très nets et la précision des ombres, de la perspective et des structures rend ces films extrêmement réalistes et convaincants.

Pour créer des images plus vivantes, l'équipe d'informaticiens du Jet Propulsion Laboratory s'est attachée à la collaboration d'artistes comme David Miller, lequel s'est rapidement passionné pour les possibilités de création offertes par l'outil informatique. Cet artiste travaille souvent la nuit lorsqu'il peut disposer de l'ordinateur pour son usage personnel. Ses créations insolites sont du plus grand intérêt et présentent des images surprenantes pour qui est habitué aux techniques traditionnelles de l'art graphique.

Les chaînes de télévision anglaise et allemande, BBC et ARD, utilisent le système ICON, comprenant un mini-ordinateur programmé dans un langage appelé RTL (en français LTR, langage temps réel) et déjà à l'œuvre dans de nombreuses autres applications, du télex à la comptabilité. Il reçoit les données en entrée et génère les informations qui servent à produire une image télévisée.

L'ICON permet d'obtenir une définition de 1 024 points pour chacune des 574 lignes de l'image. Chaque point apparaît dans la gamme des 64 couleurs de la palette électronique de base. De plus, l'ordinateur peut faire varier chaque teinte de cette palette. La couleur numéro 6, par exemple, pourra passer, en un éclair, du vert océan au bleu indigo. Cette modification des couleurs est un moyen simple et efficace de faire ressortir les détails flous d'une image par contraste avec le fond.

Dans le système ICON, une image n'est pas enregistrée sous forme de nombres indiquant la valeur de chaque point. La plupart des images comportant de larges zones de couleur unie, il est donc plus simple de les définir en indiquant une couleur et le nombre de points correspondants (X et Y). Cette méthode, appelée codification à longueur de séquence, permet de réduire considérablement le nombre des données nécessaires à la reproduction d'une image.

Il faut, en effet, trente fois moins d'informations pour définir une image de cette façon que pour la définir point par point. L'ordinateur n'a plus qu'à calculer les contours des zones où il y a des changements de couleur pour appliquer la teinte qui convient. Lorsque les images sont lues au rythme de défilement de la télévision, les compteurs numériques électroniques passent en revue les points de couleur et changent les numéros de teinte au moment voulu.

Enfin, la palette traduit les numéros en une série de tensions électriques qui correspondent à la combinaison de rouge, de vert et de bleu dont est composée chaque couleur de l'écran.

Pour les longs métrages ou les films spatiaux de la NASA, qui sont des animations très complexes, les images sont produites une à une, puis enregistrées sur une pellicule ou une bande vidéo.

Les données numériques sont communiquées à une mémoire à accès sélectif, selon un rythme relativement lent, puis lues au rythme très rapide des images télévisées.

L'ordinateur peut donc synthétiser l'image à une vitesse choisie et chaque image terminée peut être transférée sur une pellicule. Les photogrammes sont ensuite projetés à la vitesse de 24 images à la seconde de façon à donner l'illusion du mouvement. Cette technique, qui permet de produire des images très complexes, a été utilisée pour réaliser des films comme « la Guerre des étoiles » et « Star Trek ».

Certains ordinateurs offrent de nombreuses possibilités d'effets spéciaux très variés. Ils permettent ainsi de rendre l'effet de volume (trois dimensions) des personnages, des objets, des constructions. Ils se prêtent au rendu de la texture des matériaux (le lisse et le rugueux).
Enfin, ils sont capables de reproduire les effets de lumière et les différents éléments climatiques comme la pluie, le brouillard ou la neige. C'est dire le grand apport du traitement électronique au dessin d'animation.

doit passer par l'intermédiaire de l'interpréteur (ou du compilateur) ou des commandes du système d'exploitation. La logique de ces deux moyens d'accès est illustrée schématiquement page 299. Voici les principaux modules des systèmes d'exploitation :

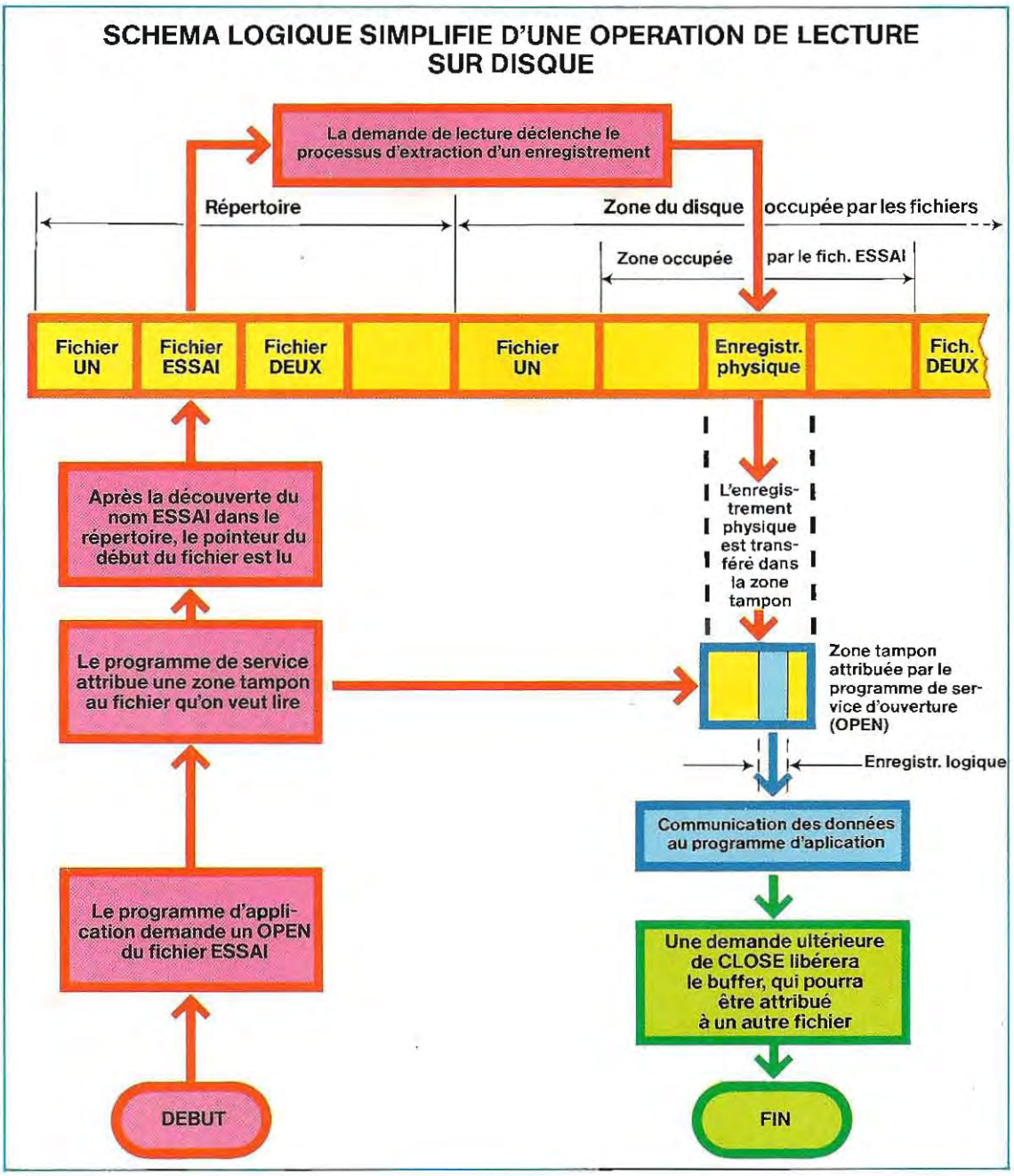
SEARCH : recherche d'un fichier dont on connaît le nom.

OPEN : ouverture d'un fichier. Cette opération permet d'accéder au contenu du fichier. Elle précède presque toujours la lecture ou l'écriture des informations.

CLOSE : fermeture du fichier. Cette opération rend impossible l'accès aux données. Il faudra ouvrir à nouveau le fichier pour manipuler les enregistrements qu'il contient.

RENAME : changement de nom d'un fichier.

SCHEMA LOGIQUE SIMPLIFIE D'UNE OPERATION DE LECTURE SUR DISQUE



READ : lecture d'un enregistrement.
WRITE : écriture d'un enregistrement.
SELECT : sélection de l'une des unités de disques (en anglais, un lecteur de disques s'appelle disk drive)

Les programmes d'application en langage évolué (Basic, par exemple) utilisent des instructions entraînant l'exécution de ces programmes de service chargés de gérer le matériel et d'effectuer les transferts de données demandés par l'application.
L'OPEN et le CLOSE représentent deux fonctions particulièrement importantes du système d'exploitation. Deux instructions Basic portent exactement le même nom.
Comme nous l'avons vu, la lecture et l'écriture impliquent le transfert de secteurs entiers de données, même lorsque l'enregistrement logique est plus petit. Une zone de mémoire tampon (buffer) accueille donc temporairement les données lors des échanges avec le disque ; l'enregistrement logique recherché est ensuite extrait du buffer (enregistrement physique). La fonction OPEN sert essentiellement à attribuer une zone tampon au fichier. Le nombre de buffers, et donc de fichiers utilisables en même temps n'étant pas illimité (il est souvent réduit à 3 ou 4), il faudra, si l'on désire utiliser plus de fichiers, libérer des zones tampons par un CLOSE avant d'ouvrir de nouveaux fichiers. Le schéma de la page 304 illustre le processus de lecture d'un enregistrement ; l'écriture se déroulerait de façon analogue. Les programmes de service sont appelés par des instructions en langage évolué mais aussi par de nombreuses commandes du système d'exploitation. On distingue les commandes temporaires (ou utilisateur) et les commandes résidentes (utilisables seulement lorsqu'on travaille en « mode système »). Dès que l'on a chargé le Basic, on n'y a plus accès directement. Cependant, d'autres instructions permettent d'exécuter les mêmes opérations en Basic ainsi qu'avec tous les autres langages évolués. Le schéma de la page 306 illustre cette répartition des commandes et les opérations correspondantes.
Voici maintenant l'explication des opérations exécutées par ces commandes et des exemples d'utilisation. Si le symbolisme d'écriture de ces commandes dépend du système d'exploitation, les opérations exécutées sont les mêmes sur tous les systèmes. Nous mentionnerons pour chaque commande l'abréviation la plus fréquemment rencontrée.

1 / Destruction d'un fichier (commande = **ERA**)
Le code ERA (abréviation du verbe anglais erase, effacer) est le symbole qui déclenche l'opération de destruction (effaçage) d'un fichier sur le disque.
Il faut fournir les indications suivantes :
- le disque sur lequel se trouve le fichier (soit A ou B, soit 0 ou 1, selon le système) ;
- le nom du fichier ;
- le type du fichier (Basic, compilé, données, etc.).

Ainsi, pour effacer le fichier ESSAI qui se trouve sur le disque A et contient un programme en Basic, l'instruction du système CP/M est la suivante :

ERA ESSAI.BAS

La commande ne provoque ici que la destruction des fichiers situés sur la disquette A (qui est souvent « prise par défaut »). Si le fichier s'était trouvé sur la seconde unité (désignée par B), il aurait fallu préciser :

ERA B:ESSAI.BAS

Avec le système d'exploitation PCOS, l'instruction correspondante est :

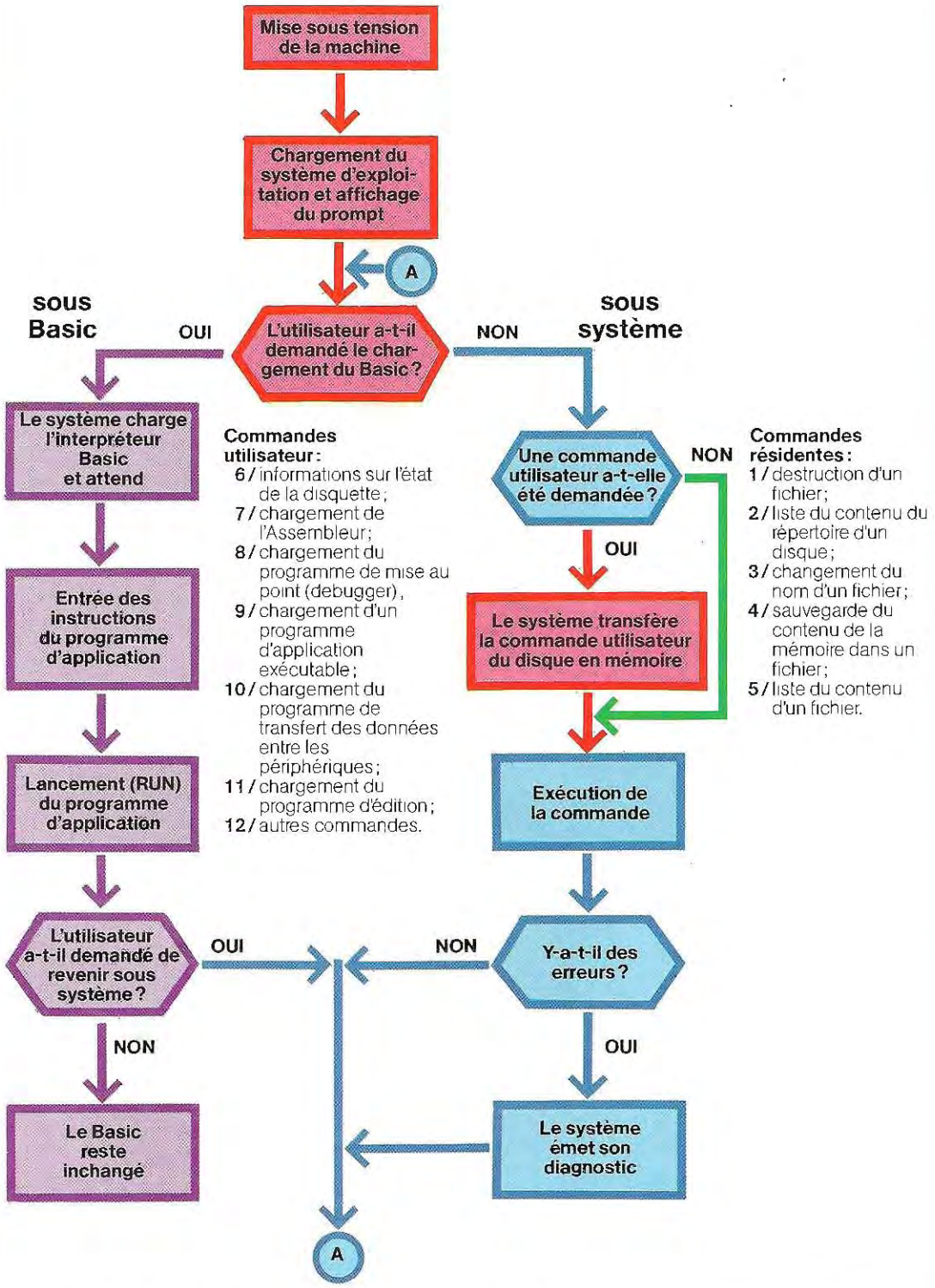
KILL "1:ESSAI"

En effet, sur ce système, les unités de disque sont désignées par 0 et 1 (et non par A et B) et on n'a pas besoin de préciser le type du fichier. La forme générale de ce type d'instruction est donc :

Code opération	Unité de disque	Nom du fichier (et type)
ERA	B:	ESSAI. BAS
KILL	1:	ESSAI
etc.		

2 / Liste du contenu du répertoire d'un disque (commande = **DIR**)
Cette instruction fournit le plus souvent à l'écran la liste des fichiers d'une disquette.

SCHEMA LOGIQUE DES FONCTIONS EXECUTEES PAR UN SYSTEME D'EXPLOITATION



La répartition des commandes résidentes et utilisateur citée en exemple est valable pour le CP/M.

Cette liste indique le type des fichiers, ainsi que, dans certains cas, leur longueur (exprimée en secteurs ou en octets). Le format habituel de cette instruction est :

DIR A : pour lister le répertoire du disque A
DIR B : pour lister le répertoire du disque B

3 / Changement du nom d'un fichier (commande = **REN**)
Cette commande (de l'anglais rename) change le nom du fichier. Son format est :

REN B:NOUVEAU.BAS = ESSAI.BAS

Le fichier ESSAI qui se trouve sur la disquette B, s'appelle désormais NOUVEAU.
Voici la structure générale de cette instruction :

Code commande	Unité de disque	Nouveau nom et type	Ancien nom et type
REN	B:	NOUVEAU.BAS =	ESSAI.BAS

4 / Enregistrement d'un programme sur un disque (commande = **SAVE**)
Cette commande est utilisée en Basic pour mémoriser sur disque un programme enregistré en mémoire centrale. Nous reviendrons plus loin sur les différentes formes de cette instruction dont la formulation est :

SAVE "A:ESSAI"

Par cette commande, le programme qui se trouve en mémoire va être enregistré sur la disquette A, sous le nom ESSAI.

5 / Liste du contenu d'un fichier (commande = **TYPE**)

Le fichier ESSAI.BAS apparaîtra avec la commande suivante, à condition qu'il ait été enregistré sous la forme d'un fichier ASCII (SAVE « A : ESSAI », A).

Code commande	Unité de disque	Nom du fichier (et type)
TYPE	B:	ESSAI.BAS

L'instruction TYPE B:ESSAI.BAS commande l'affichage du contenu du fichier ESSAI.

6 / Etat du disque (commande = **STAT**)
Elle donne des informations sur la disquette de l'unité indiquée : remplissage des fichiers, espace disponible, etc. Il suffit, en général, de préciser le nom ou le numéro de l'unité de disque et l'instruction se présente comme suit :

Code commande	Unité de disque
STAT	B:

Sur certains systèmes, dont le CP/M, cette commande fournit également des informations sur les périphériques.

7 / Chargement de l'Assembleur (commande = **ASM**)
Sur les systèmes où il est possible d'utiliser l'Assembleur, cette commande charge en mémoire un programme écrit en Assembleur et le compile (traduction en langage machine hexadécimal).
La syntaxe de cette instruction est :

Code commande	Unité de disque	Nom du programme
ASM	B:	TEST

Par cette commande, le programme TEST qui se trouve sur la disquette B va être chargé en mémoire (si toutefois il est bien écrit en Assembleur). Après sa compilation, il est enregistré sur une disquette sous le même nom mais son type est désormais HEX (langage machine) : TEST.HEX.

8 / Programme de mise au point (debugger; commande = **DDT**)
C'est une fonction très complexe qui permet de tester et même de corriger des programmes d'application pendant leur exécution.
On peut ainsi ajouter des instructions, en supprimer, examiner et modifier le contenu des zones mémoire, etc. Toutes les instructions doivent être données en langage machine (hexadécimal) puisque cette mise au point a lieu pendant l'exécution et qu'un programme en langage évolué n'est exécutable qu'après sa compilation.
L'utilisation de cette commande demande une grande expérience.

9 / Chargement d'un programme exécutable (commande = **LOAD**)
Un programme n'est exécutable qu'une fois

CONTENU DU REPERTOIRE D'UN DISQUE ET COPIE D'UN FICHIER

Le nom et le type de tous les fichiers présents sur disque sont mémorisés dans le répertoire.

La commande DIR entraîne l'affichage sur l'écran du contenu du répertoire. Ici, on a choisi le disque de l'unité A.

Chaque ligne affiche la référence de l'unité de disque choisie suivie du nom et du type de quatre fichiers.

Cette ligne nous informe que les fichiers F1000 (compilé), GENSIS (compilé), ZSID (compilé) et COB2 (programme source en Fortran) se trouvent sur la disquette A.

Le type du fichier BCLOAD n'est pas indiqué : il s'agit d'un fichier de données.

Les fichiers COMPTE, HEURE et REGR contiennent des programmes source en Basic.

Nous voyons également, sur cet écran, la commande PIP utilisée pour transférer un fichier d'une unité à une autre (unité de disque ou autre périphérique : imprimante, par exemple).

Lorsqu'on tape la commande PIP, le système répond par le symbole *. L'opérateur doit alors fournir des indications supplémentaires.

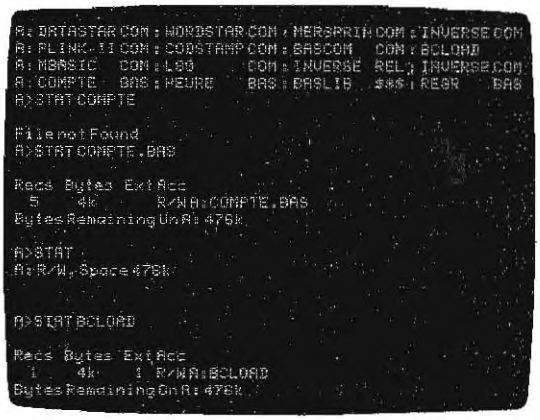
L'opérateur demande la copie du fichier REGR sur le disque de l'unité B, mais il oublie de préciser son type. Le système ne trouve pas le fichier (répertorié en BAS, comme on le voit sur l'écran) et prévient l'opérateur.

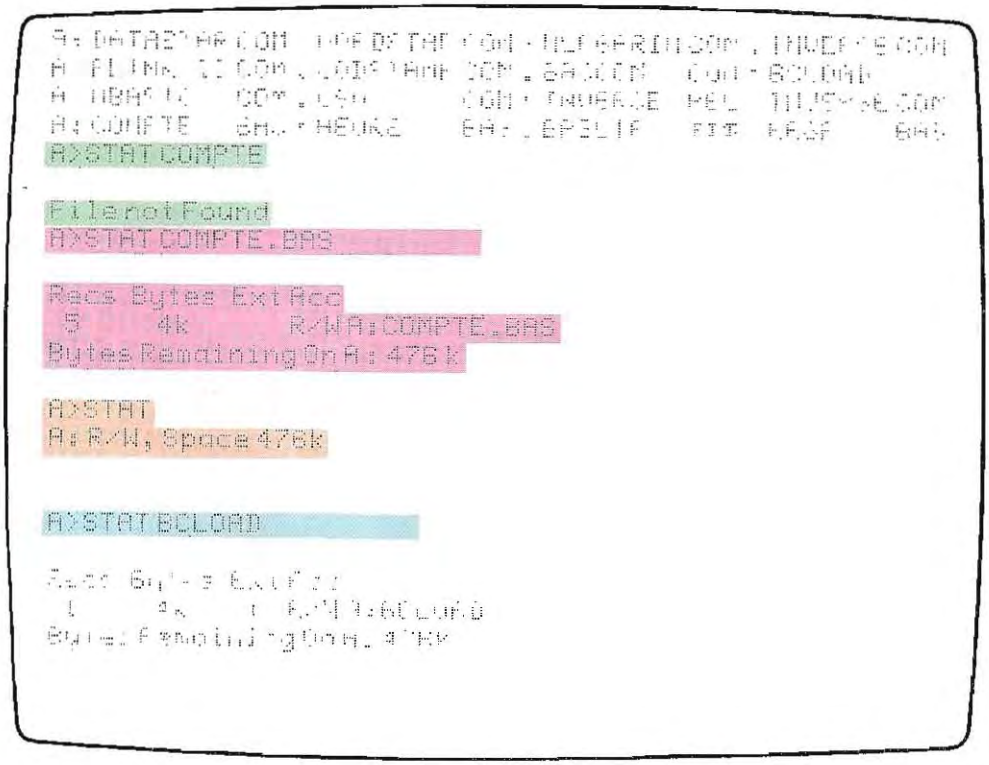
L'opérateur demande alors la copie du fichier REGR. COM. Ce fichier n'existe pas et l'ordinateur repose la question.

La demande de l'opérateur est enfin correcte et la commande est exécutée.

EMPLOI DE LA COMMANDE STAT

La commande STAT provoque l'affichage à l'écran de l'état d'un fichier.





Après l'affichage du répertoire, on a demandé l'état du fichier COMPTE sans préciser son type. Le système le considère par défaut comme un fichier de données, et ne le trouve donc pas.

La commande est reformulée correctement. Le système présente donc l'état du fichier COMPTE : celui-ci contient cinq enregistrements (logiques) ; sa longueur totale est de 4096 octets (4 Ko) ; il ne comporte pas d'extension et peut être lu ou écrit. (R et W sont les initiales de read et write). La dernière ligne indique le nombre de K octets encore disponibles sur le disque.

On aurait également pu obtenir cette information par l'ordre STAT, sans indiquer aucun nom de fichier.

On a demandé l'état du fichier de données BCLOAD. Dans ce cas, on n'a pas besoin de préciser le type du fichier (voir le répertoire), ce fichier comporte une extension

compilé ou interprété : le programme source (écrit, par exemple, en Basic) devient un programme objet en langage machine.
Le résultat de la compilation est un fichier sur disque qui contient la codification hexadécimale des instructions du programme source.
Pour travailler sur le programme compilé, il faut transférer les codes du disque vers la mémoire, par la commande LOAD dont la syntaxe est :

Code commande LOAD	Unité de disque B:	Nom du programme ESSAI

La commande LOAD B:ESSAI extrait la version compilée du programme (de type HEX) et la transfère dans la zone utilisateur de la mémoire. Le programme est alors prêt à être exécuté.
Notons toutefois que, sur les systèmes sans compilateur, l'ordre LOAD effectue le chargement des programmes écrits en Basic.
Si l'on veut charger un programme source en Basic sur un système muni d'un compilateur, il faut d'abord passer sous Basic, puis utiliser l'instruction LOAD. L'opération se déroule alors de la même manière mais c'est le programme en Basic qui est chargé.
La commande LOAD a donc plusieurs rôles :
- sur les systèmes dépourvus de compilateur, elle charge un programme source.
- sur les systèmes munis d'un compilateur, elle charge le programme compilé et le programme source sous Basic.

10 / Echange de données entre les périphériques (commande = **PIP**)
Elle permet le transfert de données ou de programmes d'un périphérique à un autre. On s'en sert essentiellement pour échanger les programmes entre les deux unités de disque.
Voici sa syntaxe :

```
> PIP
* B: = ESSAI.BAS
```

L'instruction PIP sollicite le programme d'échange. Le système répondant par le symbole * attend la suite de l'instruction indiquant entre quels périphériques l'échange doit avoir lieu.
Dans notre exemple, le programme en Basic ESSAI va être transféré sous le même nom de la disquette A à la disquette B.
Cette commande s'applique à tous les types de fichiers prévus par le système d'exploitation, quel que soit leur nom :

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

Cette instruction transfère tous les fichiers de A sur la disquette B (le symbole * qui remplace le nom et le type a une valeur générique : « quel que soit »).

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

Cette fois, seul le type est précisé (BAS). Tous les fichiers en Basic seront donc transférés.

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

Le type est laissé en blanc : il s'agit de fichiers de données de A, à recopier sur la disquette B.

11 / Editeur (commande = **ED,** ou **EDIT**)
Cette fonction permet de lire, de charger en mémoire, puis de modifier le contenu d'un fichier.
Ce programme complexe effectue de nombreuses opérations de correction telles que :
- le remplacement, la suppression et l'insertion de caractères ou de lignes ;
- la recherche de mots dans un texte ou dans un programme ;
- le remplacement de groupes de caractères.

Le travail terminé, un nouveau fichier contenant les données corrigées est créé, ou remplace l'ancien fichier.

12 / Autres commandes
Il existe de nombreuses autres commandes importantes propres à chaque système d'exploitation, tels le formatage des disquettes (pour lequel il suffit d'entrer le code de la commande, FORMAT par exemple) et la copie du système d'exploitation sur une disquette de secours.
Nous allons donner en exemple les principales fonctions du système PCOS (M 20), sans faire de distinction entre les programmes temporaires et les programmes résidents.

Test 8

1 / Enumérez les principaux éléments du système d'exploitation d'un micro-ordinateur.

2 / Laquelle de ces affirmations est vraie ?
a) la mémoire d'un ordinateur est à l'entière disposition de l'utilisateur ;
b) les programmes utilisateurs sont chargés dans la zone système de la mémoire ;
c) le système d'exploitation peut être chargé tout ou partie dans la mémoire utilisateur et tout ou partie dans la zone système.

3 / Que représentent les tables des domaines (extent) des fichiers ?
a) des tables qui contiennent les longueurs des fichiers ;
b) des listes des fichiers présents sur un disque ;
c) des tables qui contiennent les adresses des différentes parties d'un fichier.

4 / A quoi sert le répertoire (directory) ?
a) à donner des informations sur tous les fichiers ;
b) à contenir les différents éléments du système d'exploitation ;
c) à contenir les programmes d'application.

5 / En quoi consiste le formatage des disquettes ?
a) à effacer leur contenu ;
b) à les diviser en pistes et en secteurs ;
c) à vérifier leur bon état.

Les solutions du test se trouvent page 313.

- formatage de disquettes ;
- copie du système d'exploitation ;
- détermination d'un mot de passe (empêchant les personnes non autorisées d'accéder aux données) ;
- création d'un fichier ;
- détermination d'un mot de passe (password) pour un fichier particulier (seuls sont protégés les fichiers possédant un mot de passe ; tout le monde a accès au reste du disque).
- commandes spéciales modifiant le fonctionnement des périphériques (format d'impression par exemple).

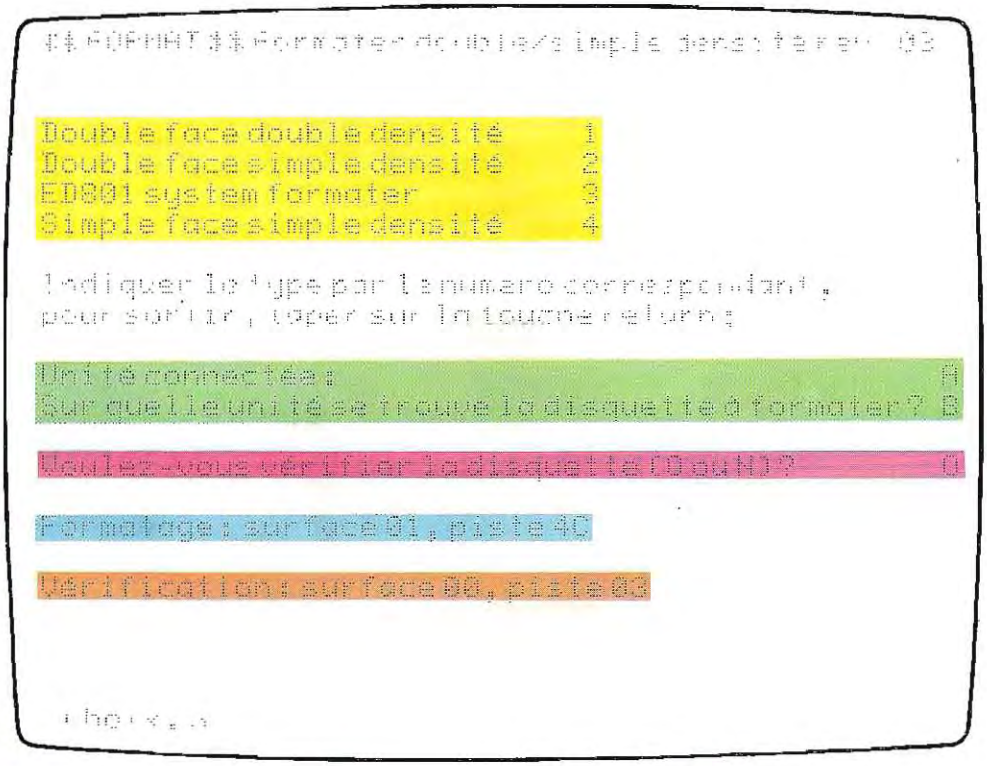
Les opérations de passage sous Basic

Une fois branché, l'ordinateur est prêt à fonctionner dès que le système d'exploitation est chargé. Toutefois, il est sous le contrôle du système et ne peut accepter les instructions en Basic. Il faut donc « passer sous Basic » avant d'entrer un programme.

Cette instruction, qui diffère selon les systèmes d'exploitation (le CP/M utilise l'instruction MBASIC, tandis que celle du PCOS est BA) provoque le chargement de l'interpréteur du Basic et on peut alors commencer la programmation.
Une fois le travail terminé, il suffira, pour revenir sous système, d'entrer une autre instruction, souvent symbolisée par SYSTEM. Toutefois, ces opérations ne sont pas nécessaires sur les systèmes qui ne peuvent travailler qu'en Basic.
Pour les autres langages compilés, comme le Fortran, la procédure est différente. On commence par écrire les instructions comme s'il s'agissait de simples données, en utilisant, par exemple, le programme EDIT, et on demande ensuite sa compilation. Le programme est traduit en langage machine et peut alors être exécuté.
On suit la même procédure pour compiler un programme en Basic si le système d'exploitation possède un compilateur.

EMPLOI DE L'ORDRE FORMAT EN CP/M

L'ordre FORMAT, qui peut prendre des noms différents selon les machines, INIT par exemple, permet de préparer un disque neuf à son utilisation ultérieure. Sur un disque déjà formaté cette commande entraîne la perte des données qu'il contenait éventuellement. Le système d'exploitation dirige, pas à pas, la procédure de formatage afin de réduire les risques d'erreurs. Voici ce qui apparaît à l'écran lorsqu'on transmet l'ordre FORMAT au système CP/M d'une machine ayant de nombreuses possibilités.



Le système indique à l'opérateur le choix entre quatre possibilités.

Le système prévient qu'il travaille pour l'instant sur l'unité A et demande à l'opérateur sur quelle unité est montée la disquette à formater. La réponse est B.

Le système demande si l'on désire vérifier la disquette. Si la réponse est oui (O), le système contrôlera le bon état de tous les secteurs de la disquette après son formatage.

Cette ligne rend compte de la progression du formatage. (sur face 01, piste 4C).

Cette dernière ligne apparaît une fois le formatage terminé, lorsque la vérification de la disquette a été demandée. Elle indique où en est la vérification. A ce stade, le système est en train de vérifier la piste 3 de la face 00. Le travail s'achèvera à la piste 4C de la face 01.

Solutions du test 8

1 / Les principales fonctions du système d'exploitation sont :
– la gestion des opérations E/S. Tous les programmes qui « interprètent » les instructions d'entrée/sortie et activent les contrôles et les fonctions nécessaires pour piloter les périphériques (disques, imprimantes, etc.) font partie du système ; un module peut être préposé à chaque périphérique ;
– l'interprétation des commandes. Ce module analyse les commandes entrées au clavier (inhérent au système d'exploitation) et contrôle leur exécution.
Ces deux composants du système d'exploitation assument les fonctions de base et doivent toujours être présents (espace mémoire résident, ou noyau résident).
Il existe de nombreuses autres fonctions (liste de l'index, échange de données entre périphériques, etc.) qui ne sont pas sujettes à un usage permanent. Les programmes associés sont stockés sur disque, et chargés en mémoire sur demande.

2 / c) l'explication est inclue dans la réponse à la question 1. La mémoire de la machine est divisée entre la zone système (résidente) et la zone utilisateur. En outre, dans le cas du Basic interprété, l'interpréteur aussi doit être chargé en mémoire. Au contraire, dans le Basic compilé, seul le programme utilisateur est chargé, car, étant déjà traduit (en phase de compilation) en langage machine, il ne nécessite pas d'interpréteur. On voit ici un autre avantage de la compilation : comme l'interpréteur n'est pas chargé, il reste à la disposition de l'utilisateur un espace mémoire plus grand, ce qui n'est pas insignifiant si l'on considère que l'interpréteur peut occuper 12 000 octets (12 Ko).
Considérons, par exemple, un ordinateur ayant 64 Ko de mémoire centrale, les répartitions sont :

Système d'exploitation	16 Ko environ	(Cela dépend de la complexité du système d'exploitation)
Interpréteur	12 Ko	
Zone utilisateur	36 Ko	

En l'absence d'interpréteur, pour le Basic compilé, la zone utilisateur est de 48 Ko.

3 / c) un fichier, qu'il contienne des programmes ou des données, ne conserve pas toujours sa longueur initiale.
Les mises à jour successives du contenu peuvent demander un nombre d'enregistrements supérieur à celui dont le fichier dispose déjà. En ce cas, les nouveaux enregistrements sont écrits sur des emplacements du disque éloignés du fichier auquel ils appartiennent. Pour réunir logiquement les différents morceaux, on utilise la table des extensions (table des domaines) qui contient, pour chaque fichier, la position physique des différentes extensions rajoutées.

4 / a) Le répertoire est une zone du disque gérée par le système d'exploitation, et dans laquelle sont mémorisées les informations du contenu du disque (emplacement, nom et type de fichier, mots de passe, etc.).

5 / Les trois réponses (a, b et c) sont globalement bonnes. Le but principal du formatage est de partager le disque en pistes et secteurs, pour créer une méthode d'accès à l'espace physique. Pendant cette opération, tout le contenu du disque est effacé. En outre, dans beaucoup de systèmes d'exploitation, une fonction de vérification lit et écrit tout le disque, de façon à déceler les éventuels espaces défectueux.

Mais on a l'avantage de pouvoir tester le programme de manière interactive. Les erreurs éventuelles peuvent être corrigées avant compilation. Au contraire, dans les langages sans interpréteur, il faut compiler le programme source après chaque modification.

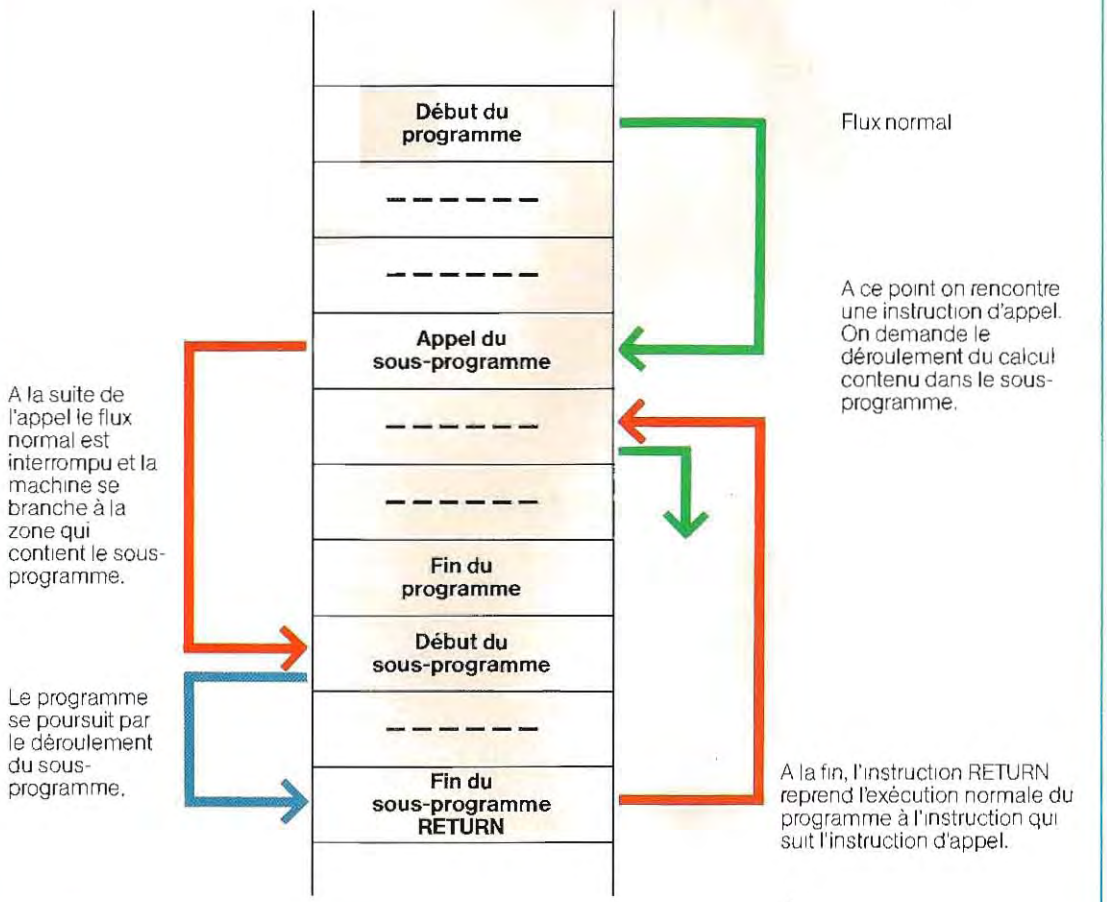
Critères d'évaluation d'un système d'exploitation

Les fonctions principales d'un système d'exploitation constituent les critères auxquels on se réfère pour juger le système.
En règle générale, un système d'exploitation est très complexe et il faut une solide expérience et des connaissances approfondies pour savoir en apprécier la valeur. En effet, un logiciel peut contenir des erreurs.
La phase d'essai est une opération délicate, et même si l'on a recours à des méthodes de contrôle très fiables, on n'est jamais assuré de mettre en évidence tous les défauts éventuels d'un programme. C'est pourquoi on se limite le plus souvent à formuler une estimation globale des fonctions du système d'exploitation. Généralement, le compilateur Basic est prioritaire, le reste étant à évaluer cas par cas, selon les besoins de l'utilisateur.

Subdivision des programmes

Lors de l'écriture de programmes complexes, il faut tenir compte de deux conditions fondamentales : éviter d'écrire plusieurs fois les mêmes séquences d'instructions, et faire tenir le programme entier dans l'étendue maximale de l'espace utilisateur (qui fait partie de la mémoire transitoire ; si le programme est en Basic interprété, une partie considérable de la mémoire transitoire sera occupée par l'interpréteur).
Un programme donné peut, en divers points de son développement, recourir à des calculs identiques, même s'ils s'appliquent à des valeurs numériques différentes. Les écrire à nouveau toutes les fois qu'ils servent constituerait un gaspillage inutile de la mémoire ; il vaudra mieux réutiliser les mêmes calculs, mais il faudra pour cela commencer par préparer d'abord une version unique de chacune des procédures en l'écrivant de façon généralisée. Une partie de programme ainsi structurée prend le nom de **sous-programme** (ou routine). Pour obtenir le déroulement des calculs contenus dans celui-ci, il suffit de l'« appeler » au cours du programme avec les instructions appropriées.
Pour la machine, cet appel détermine un saut, du point (point d'appel) du programme voulant utiliser le sous-programme, au point par lequel commence le sous-programme appelé. A la fin du sous-programme, il faut écrire l'instruction RETURN (retour). En la reconnaissant, le contrôle est redonné à l'instruction qui suit l'appel et se poursuit normalement jusqu'à la fin du programme. D'un point de vue logique, c'est comme si un bloc à part (routine) était inséré aux emplacements nécessaires chaque fois que cela s'impose. Dans le schéma de la page 315, on a reproduit le déroulement logique d'un appel de sous-programme. Dans les organigrammes, l'appel d'un sous-programme s'indique le plus souvent avec le symbole d'action générale (rectangle), dans lequel est spécifié le sous-programme appelé. Dans les systèmes d'exploitation plus complexes, les sous-programmes peuvent ne pas faire partie du programme.
Ils résident généralement sur disque et sont chargés en mémoire quand il le faut. Pendant le chargement, ils peuvent prendre la place d'un autre sous-programme qui n'est pas utilisé à ce moment. Il se produit alors un recouvrement (en des moments successifs) de différents sous-programmes dans le même espace mémoire, avec une occupation très inférieure à celle qui serait nécessaire si tous les sous-programmes étaient simultanément en mémoire.
On peut trouver des exemples de sous-programmes dans les fonctions mathématiques de tous les langages de haut niveau. En Basic, comme dans les autres langages, on peut développer des calculs complexes (exemple : la racine, les logarithmes, etc.) grâce aux sous-programmes du système, qu'on appelle plus précisément fonctions.
La symbolique des sous-programmes système est différente de celle employée pour les sous-programmes écrits par l'utilisateur, mais le mécanisme est le même. Généralement, pour appeler une fonction système il suffit d'en indiquer le nom (ainsi la racine est appelée SQR), alors que, pour les programmes de l'utilisateur, il faut une instruction spécifique prenant en charge le mécanisme de saut et de retour.

MECANISME D'APPEL D'UN SOUS-PROGRAMME (CONTENU DANS LE PROGRAMME)



Malgré l'analyse correcte du problème en phase de conception et l'utilisation des sous-programmes, on arrive fréquemment à une occupation mémoire excessive. L'obstacle peut être contourné en utilisant deux techniques similaires : l'enchaînement et le lancement d'un programme à partir d'un autre programme.

L'**enchaînement** (CHAIN) a des aspects voisins de la gestion des sous-programmes non résidents et il consiste à charger en mémoire certaines parties du programme seulement quand elles sont nécessaires, en les remplaçant de temps à autre. A la différence des systèmes plus évolués, dans les ordinateurs personnels et les micro-ordinateurs, l'utilisateur doit lui-même gérer tous les transferts.

Le schéma de la page 316 illustre l'exécution d'un enchaînement. Par exemple, le programme utilisateur est divisé en trois parties, respectivement PROG-1, PROG-2, PROG-3. La première partie (PROG-1) constitue le « corps » principal, et réside en permanence en mémoire ; les deux autres parties sont chargées par l'opération d'enchaînement seulement quand elles servent.

La seconde méthode consiste à exécuter un autre programme au moyen d'une instruction de **lancement** (RUN) contenue dans le programme d'appel. La mémoire ne garde pas

trace du programme d'appel, et le nouveau bloc remplace le précédent.

Transfert des paramètres

Pour qu'un programme puisse fonctionner, tous les points du programme doivent « connaître » les variables utilisées. Par exemple, pour écrire un programme qui calcule la superficie d'un cercle, il faut entrer la valeur du rayon. Si ce programme est écrasé par un autre module, la valeur du rayon est perdue, à moins que le programmeur ait transmis la valeur au nouveau module, c'est-à-dire, ait opéré la transfert du paramètre. Ce transfert se déroule de façon différente selon la technique utilisée pour écrire les différentes parties du programme.

Dans les sous-programmes (à l'exception des cas qui seront mentionnés dans l'explication du langage Basic), il ne faut aucune instruction particulière.

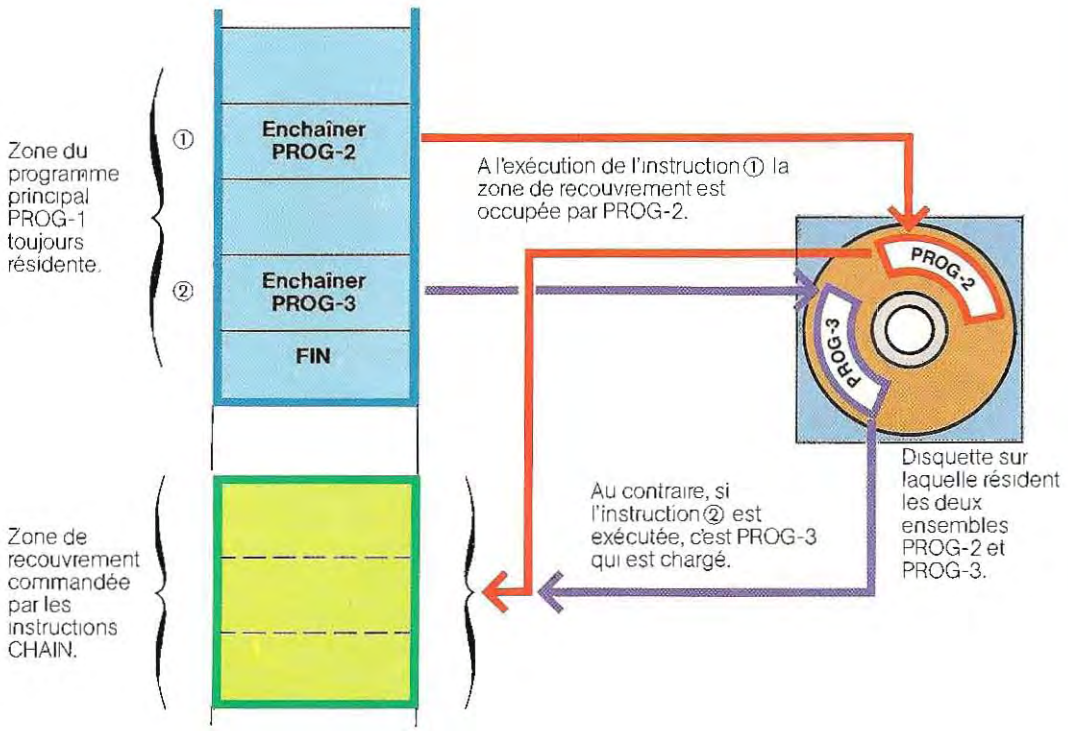
Les sous-programmes (dans la forme standard du Basic) sont parties intégrantes du programme principal, et connaissent donc toutes les valeurs (paramètres).

L'opération CHAIN nécessite une instruction particulière (COMMON) déclarant quelles sont les valeurs utilisées en commun par tous les modules enchaînés. Avec l'instruction RUN, on ne peut passer aucun paramètre en utilisant le disque comme mémoire temporaire. Les valeurs à transférer sont écrites sur disque par le programme appelant (qui contient l'instruction RUN) et sont lues, après chargement, par le programme appelé.

L'étude des instructions Basic de segmentation des programmes sera approfondie dans le chapitre consacré à la syntaxe du langage.

Dans les autres langages, même ceux très semblables au Basic, comme le Fortran, les méthodes diffèrent par la forme. Les variantes principales seront exposées dans les chapitres sur les différents langages.

SCHEMA D'EXECUTION D'UN ENCHAINEMENT



La zone de recouvrement est occupée tour à tour par l'un des deux segments (PROG-2 ou PROG-3) selon qu'est exécutée l'instruction ① ou l'instruction ②

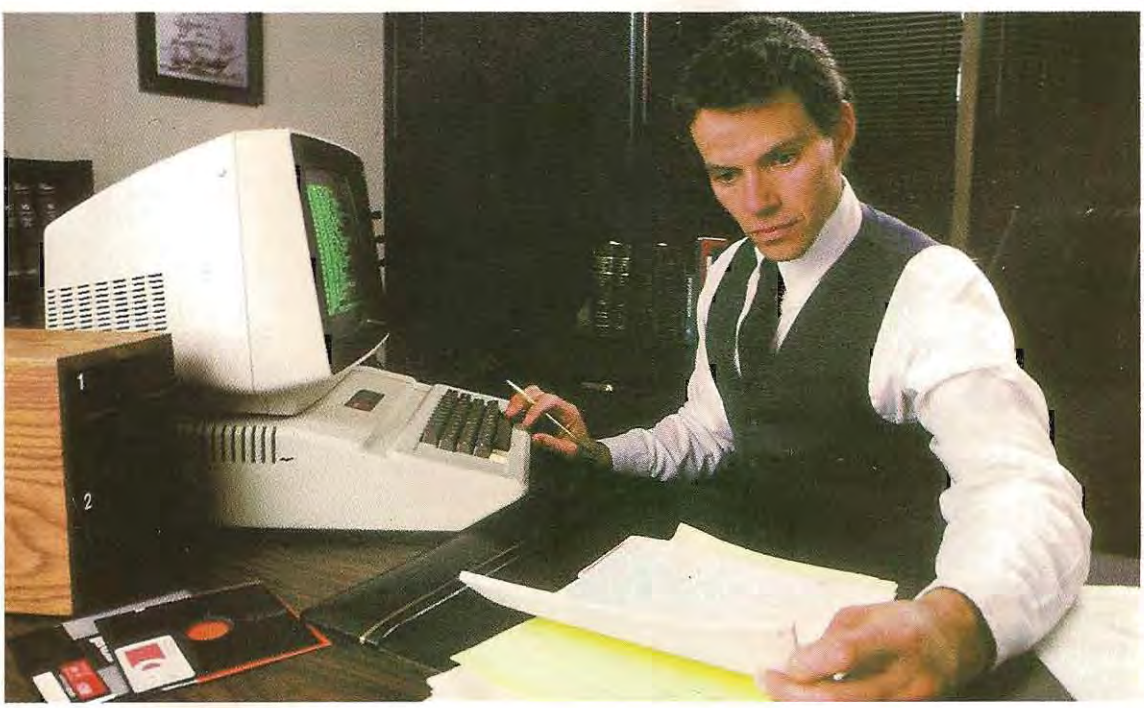
Le langage de programmation Basic

Les seuls moyens pour dialoguer avec les micro-ordinateurs et les ordinateurs personnels sont le clavier, comme dispositif d'entrée, et l'écran (ou l'imprimante), comme dispositif de sortie. Toutes les commandes et les instructions sont entrées au clavier ; la réponse éventuelle de la machine est affichée sur l'écran vidéo ou écrite sur l'imprimante.

L'entrée au clavier se fait en tapant la commande ou l'instruction comme s'il s'agissait d'une simple machine à écrire. Elle transforme chaque symbole en un signal numérique. A la fin de l'entrée, il faut presser la touche CR (sur certains claviers elle est indiquée par RETURN, ou par le symbole ↵). Supposons que nous sommes en système d'exploitation et que nous voulons obtenir la liste des fichiers contenus sur le disque. L'instruction à entrer est DIR ; dont il faudra frapper (entrer au clavier) les trois lettres DIR et à la fin presser la touche CR (ou RETURN, ou ↵, selon le type de clavier). La touche CR permet d'avertir la machine que l'on a terminé de composer l'instruction et que l'on peut procéder à son exécution.

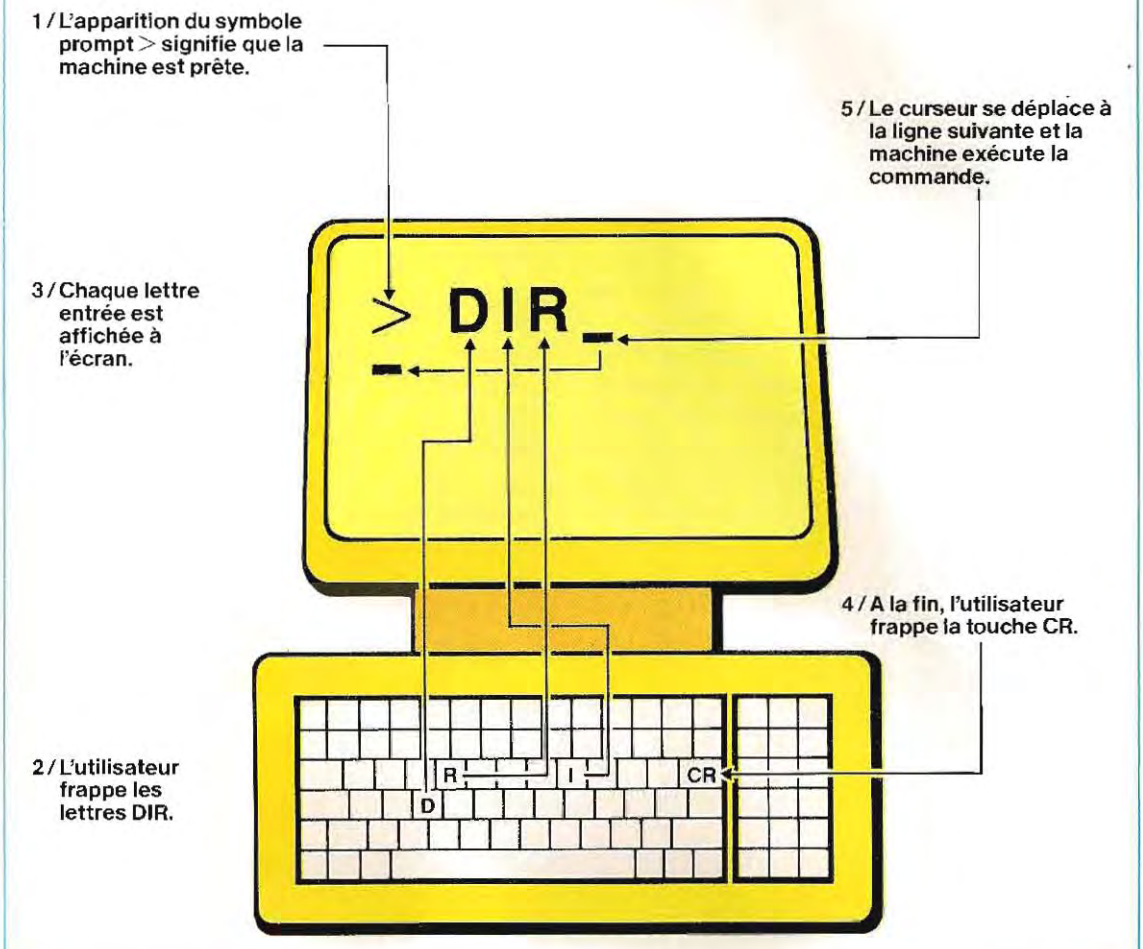
Toute commande entrée au clavier apparaît à l'écran (ainsi les lettres DIR) : de cette façon, il est possible de vérifier l'exactitude de ce que l'on écrit et d'apporter d'éventuelles corrections. A la fin de la phrase, le curseur s'arrête à côté de la dernière lettre. En pressant la touche CR le curseur se positionne au début de la ligne suivante et l'ordre frappé est exécuté. Cette séquence est illustrée par le schéma de la page 318. Le fonctionnement ainsi exposé se nomme **écho** (l'affichage à l'écran est l'écho de la frappe au clavier), et c'est la façon normale d'opérer. En certaines circonstances (par exemple, à l'entrée d'un mot de passe), on peut supprimer l'écho. Tout ce qui a été frappé au clavier n'est pas restitué à l'écran, et un observateur éventuel ne pour-

L'ordinateur est devenu le matériel de pointe du secteur tertiaire, au niveau le plus élevé.



Marka

ENTREE DE LA COMMANDE DIR



rait pas lire ce que l'on écrit. Deux formats type de clavier sont représentés page 319. L'emplacement des lettres diffère selon les claviers, qui sont « AZERTY » en France, ou « QWERTY » (clavier international de langue anglaise). Certains claviers présentent en outre des touches (de fonction) programmées pour exécuter des fonctions particulières. On peut ainsi éviter parfois de frapper toutes les lettres d'une commande donnée.

Sur certains claviers, il existe des touches non utilisées, qui restent libres de toute inscription et sont réservées pour d'autres applications (par exemple PRINT, REPT, CLEAR, EOL, BREAK).

Sur tout clavier, une barre d'espacement (SPACE) a la même fonction que celle d'une machine à écrire, c'est-à-dire l'insertion d'un blanc (dans un ordinateur, un blanc équivaut à un caractère, et, comme tout caractère, occupe une position mémoire). Les lettres des touches sont minuscules. Or les commandes et les instructions d'un programme se composent normalement de lettres majuscules ; c'est pourquoi le clavier prévoit aussi des majuscules ; pour les obtenir, il faut simplement tenir pressée la touche SHIFT lorsqu'on tape une lettre. Pour écrire une longue suite de lettres majuscules, on bloquera le sélecteur des lettres majuscules en poussant la touche LOCK (on débloquera LOCK par simple pression). Certaines touches portent un double symbole : le second symbole sera obtenu par pression préalable de la touche

Daniel Bouquignaud

Le clavier d'un ordinateur ressemble beaucoup à celui d'une simple machine à écrire, mais il comporte plus de touches, certaines d'entre elles étant programmables. Ces touches spéciales sont souvent désignées par la lettre F (Fonction) suivie d'un chiffre. En haut : clavier du M20 (Olivetti). En bas : celui du HP150 (Hewlett-Packard).

SHIFT qui prend donc aussi en charge la sélection des caractères des touches doubles Il existe enfin deux touches particulières, ESC (escape) et CTRL (contrôle), dont la mise en œuvre provoque l'envoi de caractères particuliers ayant un rôle que nous étudierons par la suite.
En résumé, les touches du clavier d'un micro-ordinateur ou d'un ordinateur personnel sont :

- **alphabétiques et numériques :** pour l'entrée des données ;
- **fonctionnelles :** il s'agit de touches dont la fonction est déterminée par programme ;
- **spéciales** (ESC, CTRL) : elles ont des fonctions particulières.
- Certaines touches **non employées** sont laissées libres pour d'autres applications au clavier ;
- la touche **CR** permet de replacer le curseur au début de la ligne suivante ;
- la touche **SHIFT** sert à obtenir les majuscules, ou les caractères du haut des touches ayant deux symboles ;
- la touche **LOCK** bloque l'état SHIFT.

L'utilisation d'une touche quelconque génère le code correspondant ASCII. A la fin de l'instruction, avec l'entrée du code CR (touche CR), les caractères (codifiés ASCII), sont envoyés à l'unité centrale et traités par le programme résident en cours.
Certains symboles spéciaux (flèches, barres, etc.) sont typiques du Basic alors que dans d'autres systèmes d'exploitation ils ne sont pas reconnus. En ce cas, la machine rend un diagnostic mettant l'erreur en évidence et demandant une nouvelle saisie.

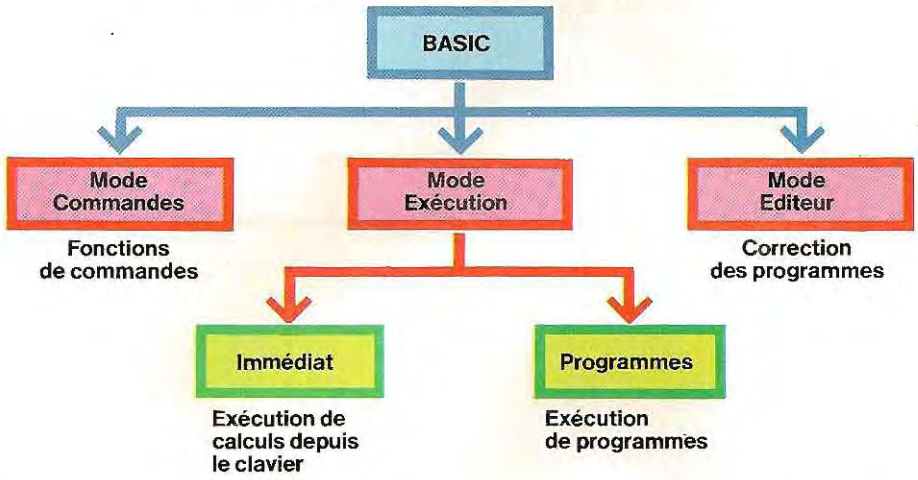
Généralités

Le langage Basic (Beginner's All purpose Symbolic Instruction Code) est né en 1963, développé par Kemmeny et Kurts sous forme interprétée, et doté d'instructions très restreintes. Il était destiné à des utilisateurs inexpérimentés en programmation, et était limité à un usage spécifique.
En peu d'années, il devait pourtant jouir d'un essor considérable, consécutif au développement toujours plus étendu des micro-ordinateurs et des ordinateurs individuels jusqu'à devenir un langage complet, compilable, et constituer une alternative valable aux langages déjà existants (Fortran, Cobol, RPG, etc.). Il a toutefois conservé ses caractéristiques initiales de simplicité et de compréhension immédiate qui en font le langage le plus adapté à ceux qui débutent en informatique. Sa structure particulière et la grande variété des instructions qu'il rassemble permettent aussi au Basic moderne d'être un langage très utilisé, même par des professionnels de l'informatique. L'étude du Basic n'est pas seulement une forme d'introduction à la programmation, elle donne aussi les moyens de développer toutes sortes d'applications, qu'elles soient de type scientifique ou de gestion. En outre, cette étude n'est pas limitée aux ordinateurs individuels et aux micro-ordinateurs, car la majorité des machines peuvent travailler en Basic.
Comme tout langage, le Basic doit aussi correspondre aux normes qui en décrivent les fonctions et les modalités d'exploitation. Les fabricants de logiciel et de matériel doivent au moins respecter les normes correspondant à chaque type de système.
La version du Basic que nous allons étudier est la version standard définie dans le document officiel ANSI - BSRX 3.60 - 1978, émis par l'office américain pour la définition des standards, et augmentée des fonctions propres au Basic 80.
Nous présenterons également quelques particularités adoptées par les fabricants les plus connus, et les caractéristiques du Basic compilé.

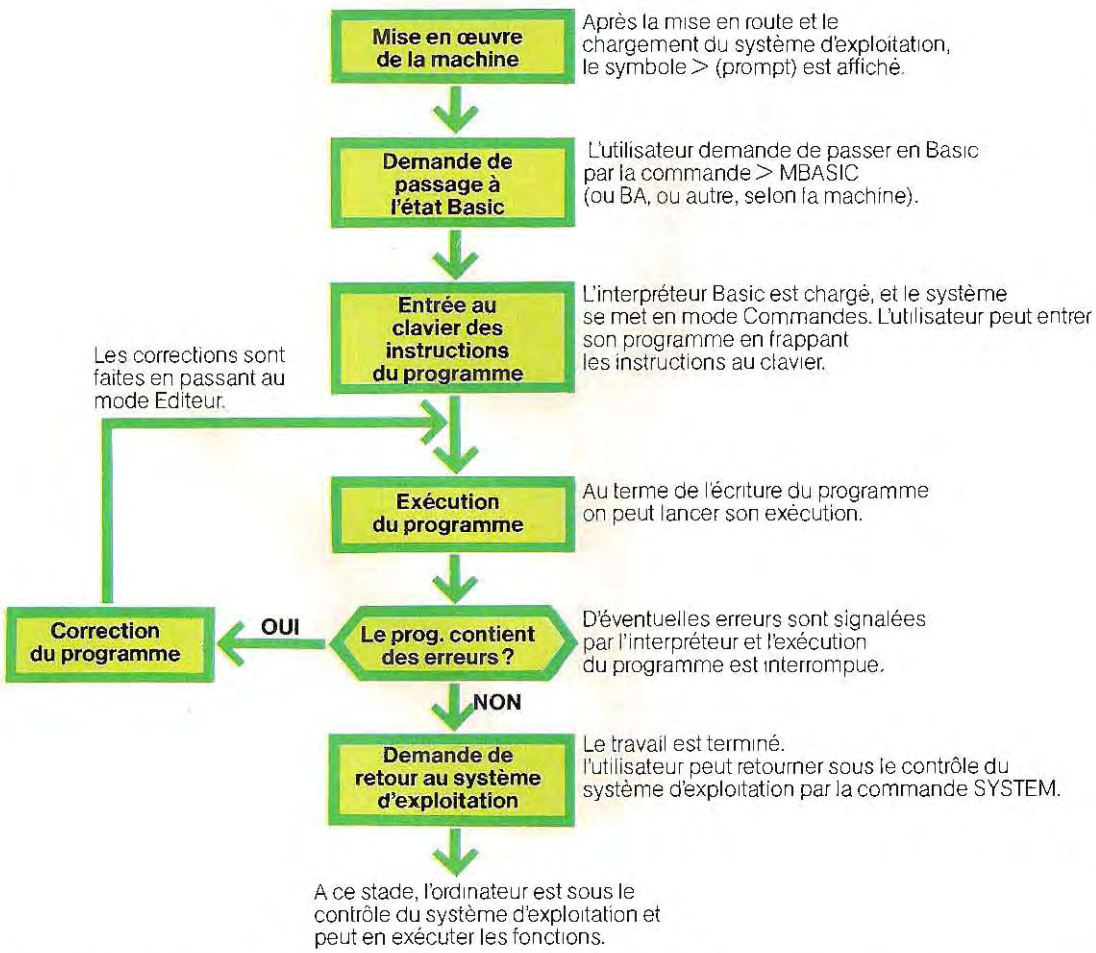
Les modes opératoires

Le Basic interprété possède trois modes opératoires (voir schéma en haut de la page 321) : un mode Commandes, un mode Exécution, et un mode Editeur.
Le mode Commandes permet de saisir toutes les fonctions de commande, telles la demande d'édition du listing d'un programme, les fonctions de gestion des fichiers, l'entrée d'un programme, etc. On a recours au mode Exécution pour lancer les programmes d'application ou exécuter des calculs immédiats. Le mode Editeur sert à la correction des erreurs dans les programmes d'application. Le schéma en bas de la page 321 illustre la procédure d'écriture et de lancement d'un programme d'application.

MODES OPERATOIRES DU BASIC



PROCEDURE COMPLETE D'ECRITURE ET DE LANCEMENT D'UN PROGRAMME D'APPLICATION



Structure des instructions Basic

Un programme Basic est constitué d'une succession de lignes (instructions) qui spécifient la suite logique des actions à accomplir. L'entrée de chaque ligne survient au terme de son écriture en poussant la touche CR, comme pour toute saisie.

Chaque ligne doit être précédée d'un numéro qui la distingue des autres. Ce numéro est l'adresse de l'instruction, et l'exécution du programme (mis à part l'intervention d'instructions de saut) va du plus petit numéro de ligne jusqu'au plus grand.

La numérotation peut partir d'une valeur quelconque et se poursuivre avec un pas quelconque, même variable. Les seules limitations concernent la valeur maximale (normalement elle est de 32 767, donc la ligne numérotée 40 000 ne peut exister) et la séquence de progression. Si l'on rentre les instructions numéros 10, 35, 160, 2 000, puis l'instruction numéro 20, l'interpréteur Basic place cette dernière ligne à sa place dans l'ordre croissant (qui n'est pas celui d'entrée). Le programme sera constitué de la séquence : 10, 20, 35, 160, 2 000, même si la ligne 20 a été la dernière entrée. Le format des instructions est celui-ci :

nnnn	Instruction
1 260	PRINT 6 + 4 (CR)

Le symbole nnnn est le numéro de ligne ; le signe **CR** indique que, pour terminer et saisir la ligne, il faut frapper la touche CR.

On peut écrire différentes instructions de programme (le maximum de caractères qu'il est possible d'entrer sur chaque ligne est 72 ou 255, selon la version du Basic) sur la même ligne physique, les différentes instructions étant séparées par un symbole déterminé. Le plus employé est : , mais certaines machines font usage du symbole backslash, \ . Par exemple le programme

```
10 PRINT  6 + 4
20 PRINT 10 - 2
30 PRINT  8 + 3
```

peut être écrit de la manière suivante :

```
10 PRINT 6 + 4 : PRINT 10 - 2 : PRINT 8 + 3
```

Si la machine adopte l'autre symbole \ , on aura :

```
10 PRINT 6 + 4 \ PRINT 10 - 2 \
PRINT 8 + 3
```

Les fonctions de la touche (CTRL)

Dans la présentation du clavier, on a mentionné la présence de certaines touches spéciales qui ont des fonctions particulières.

Une d'entre elles est la touche CONTROL,

Codes commandes du Basic 80

CTRL + A	La machine se met en mode éditeur (correction des programmes).
CTRL + C	Interrompt l'exécution d'un programme en cours (BREAK) et met la machine en mode commande.
CTRL + G	Sert à activer le signal acoustique habituellement présent sur l'unité vidéo. Cette fonction est également utilisée pour attirer l'attention du programmeur quand son intervention est requise. En ce cas, c'est le programme d'application qui doit émettre, au terminal, le même code que celui généré par la saisie simultanée des touches CTRL + G.
CTRL + H	Déplace le curseur (vers la gauche) et supprime le dernier caractère. Si, par exemple, apparaît sur l'écran le mot PARIS, la commande CTRL + H efface le dernier caractère ; on a PARI. En phase de correction, on effacera autant de caractères que l'on voudra en répétant l'entrée de cette instruction.
CTRL + I	Déclenche la tabulation. Le curseur se déplace de 8 colonnes à chaque commande CTRL + I. L'écran comportant généralement 80 colonnes, on peut effectuer 10 tabulations au maximum.
CTRL + O	Ferme momentanément la sortie d'un programme d'application qui tourne (par exemple, l'impression) ; une seconde entrée de cette même commande réactive la sortie du programme interrompue précédemment.
CTRL + R	Réécrit une ligne (on l'emploie pour dupliquer une ligne sans avoir à la retaper).
CTRL + S	Suspend l'exécution d'un programme (ne pas confondre avec CTRL + C, qui interrompt définitivement le programme).
CTRL + Q	Réactive le programme suspendu par CTRL + S.
CTRL + U	Efface une ligne.

habituellement indiquée par le sigle CTRL.
Comme on peut voir sur le tableau de la page 322, les touches dites spéciales ne sont pas des touches uniques, mais une combinaison de la touche CTRL, et de l'une des lettres (A, C, G, H, I, O, R, S, Q ou U). Utilisées conjointement, elles deviennent une série de commandes interprétées et exécutées par le Basic. La symbolisation CTRL + une lettre indique que, pour activer la fonction, il faut presser simultanément la touche CTRL, et celle de la lettre relative souhaitée.
Toutes les commandes possibles figurent sur le tableau à côté de chaque code.

Utilisation interactive du Basic (mode direct ou immédiat)

Le langage Basic, grâce à son interpréteur, permet l'exécution immédiate de calculs, sans d'autre instruction que la saisie au clavier du calcul à exécuter. Cette façon de procéder n'est pas possible dans d'autres langages, car il faut d'abord saisir la totalité de la formule du calcul à exécuter, comme s'il s'agissait d'une ligne source (c'est-à-dire écrite en symbolique de haut niveau, Fortran, Cobol, etc.), puis compiler la ligne, comme s'il s'agissait d'un programme ordinaire et l'exécuter ensuite. La caractéristique fondamentale du Basic réside dans son programme d'interprétation, qui analyse, et traduit en langage machine chaque ligne au moment de la saisie. Cela constitue un avantage appréciable surtout pour la recherche des erreurs.
Pour vérifier l'emplacement d'une erreur, on peut développer les calculs un par un (mode immédiat). L'utilisateur suivra le déroulement du programme en simulant les fonctions au clavier et en procédant instruction par instruction ; ainsi, il vérifiera les résultats et trouvera l'instruction erronée.
Les opérations qui peuvent se dérouler de façon interactive sont celles prévues dans le fonctionnement normal (programmation), et on peut utiliser les parenthèses. Le calcul à exécuter doit être précédé d'une commande spécifiant la fonction demandée.
Si on veut voir apparaître le résultat sur l'écran, il faut donner l'instruction :

PRINT formule du calcul (CR)

alors que sur l'imprimante, elle devient :

L PRINT formule du calcul (CR)

Rappelons que le symbole CR indique que l'exécution d'un calcul s'obtient en pressant la touche CR. Par exemple l'instruction :

PRINT (7 + 3) / 2 (CR)

donne sur l'écan le résultat : 5.
En mode interactif, on peut aussi utiliser les opérateurs logiques (AND, NOT, OR, etc.).
Par exemple, PRINT 5 OR 2 donne 7 (en binaire 5 = 101 et 2 = 10, dont 101 OR 10 = 111 = 7 en décimal).
Le langage Basic (comme tout langage) dipose d'une « bibliothèque » de fonctions, c'est-à-dire d'un groupe de programmes exécutant des fonctions mathématiques spéciales et pouvant être employées de façon immédiate.
Par exemple, la racine carrée d'un nombre est calculée par un de ces programmes, appelé **algorithme.**

Symbolique, signification et priorité des opérateurs

Les opérateurs reconnus par le Basic sont de quatre types :

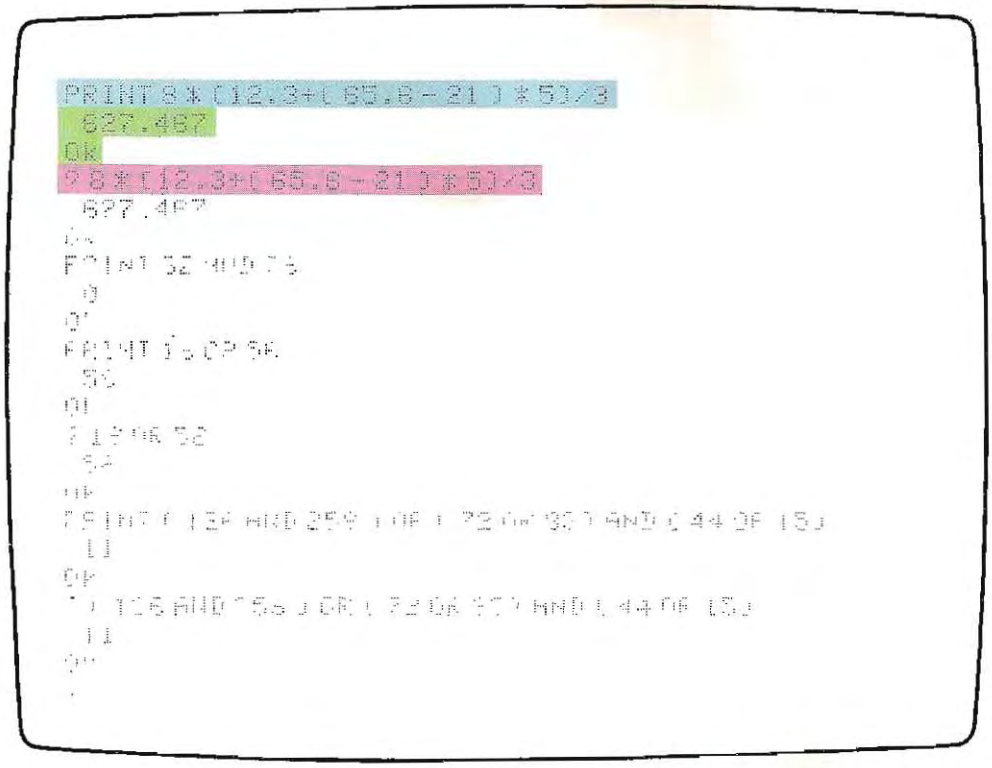
- Arithmétiques
- Relationnels
- Logiques
- Fonctionnels

Chaque groupe possède sa propre hiérarchie, c'est-à-dire un ordre de priorité des calculs. L'ordre hiérarchique peut être modifié par l'emploi de parenthèses. Le calcul s'effectue alors en commençant par l'expression entre parenthèses comme dans une expression algébrique classique.
Dans un calcul mixte, c'est-à-dire contenant différents types d'opérateurs, la priorité suit l'ordre des opérateurs mentionnés ci-dessus. Par exemple, si un calcul contient des opérateurs logiques et des opérateurs arithmétiques, les opérateurs arithmétiques seront considérés en premier.

Opérateurs arithmétiques

Les opérateurs arithmétiques prévus en Basic, par ordre de priorité, sont les suivants :

UTILISATION DU BASIC EN INTERACTIF



On peut exécuter des opérations arithmétiques et logiques en mode interactif au moyen de la seule commande PRINT suivie de l'expression à calculer. Une fois l'expression écrite on doit frapper la touche RETURN.

Le système répond par le résultat de l'opération qu'on lui a soumise et se prépare à recevoir une nouvelle entrée.

L'instruction PRINT peut être remplacée par le symbole « ? » qui a la même signification.
En mode interactif toutes les fonctions du Basic peuvent être utilisées.

Opérateur	Symbole	Exemple
Elévation à une puissance	∧	3∧2
Changement de signe	–	– 3
Multiplication	*	5*2
Division	/	8/3
Addition	+	2+7
Soustraction	–	6–4

Dans certaines machines, l'exposant peut aussi être indiqué par le symbole ↑, ou par deux symboles de multiplication **. Par exemple, le calcul 3^2 peut être symbolisé : 3∧2, 3↑2, ou 3**2.

Dans l'écriture des expressions à calculer, soit en mode interactif, soit en programmation, il faut prêter une attention particulière à la priorité des calculs. Les opérations se déroulent selon l'ordre indiqué par le tableau ; le premier calcul à faire est l'élévation à la puissance. Viennent ensuite le changement de signe, la multiplication, etc. L'ordre peut toutefois être changé par l'usage des parenthèses.
Par exemple l'instruction :

PRINT 5∧2 + 3*7 / 2

donne 35.5*. Sur le schéma de la page 325

EXEMPLES DU DEROULEMENT DE CALCULS EN BASIC

5∧2+3*7/2

Etape 1 25+3*7/2

Etape 2 25+21/2

Etape 3 25+10.5

Etape 4 35.5

5∧2+3*7/2=35.5

Le premier calcul exécuté est 5∧2 = 5^2 = 25, car l'exposant a la priorité la plus haute ; il est suivi de l'opérateur * (multiplication), puis de l'opérateur / (division) et enfin de l'opérateur + (addition).

5∧(2+3)*7/2

Etape 1 5∧5*7/2

Etape 2 3125*7/2

Etape 3 21875/2

Etape 4 10937.5

5∧(2+3)*7/2=10937.5

Dans ce cas le calcul entre parenthèses (5∧5 = 5^5 = 3125) est exécuté en premier.

Expression algébrique	Expression en Basic	
(5x3)–7:2	(5*3)–7/2	
$(9^2+2x6):4^3$	(9∧2+2*6)/4∧3	(9^2=9∧2; 4^3=4∧3)
$\frac{3^2+4}{6}$	(3∧2+4)/6	
$7^{2/3}+9$	7∧(2/3)+9	
$\frac{5x3+2-1}{8^2}$	(5*3+2–1)/8∧2	Le calcul peut être indiqué par des symboles (A, B, C, etc.) auxquels on assignera des valeurs numériques.
$\frac{AxB}{C}+C^2$	(A*B)/C+C∧2	

est reproduite la succession des calculs.
En modifiant la priorité par l'usage des parenthèses, on aura :

```
PRINT 5∧(2+3)*7/2
```

dont le résultat est 10937.5. Pour le développement d'un calcul (expression algébrique), il suffit de remplacer les opérateurs arithmétiques (somme, produit, etc.) par les symboles correspondants. Le schéma de la page 325 en fournit quelques exemples.

Opérateurs relationnels

Ils sont utilisés pour comparer deux grandeurs. Le résultat peut avoir la valeur 0, si la condition est fausse, la valeur −1, si la condition est vraie.
Les opérateurs relationnels sont habituellement employés dans les instructions de « décision »

et le résultat de l'opération (vrai ou faux) est évalué par l'instruction elle-même ; l'utilisateur l'emploie implicitement. On trouvera ci-dessous, classés par ordre de priorité, les opérateurs relationnels disponibles en Basic.

Opérateur	Symbole	Exemple	
Egalité	=	A = B	met la valeur de A égale à celle de B, ou bien la condition est vraie si A est égal à B.
Inégalité	<>	A <> B	vrai si A est différent de B.
Inférieur	<	A < B	vrai si A est inférieur à B.
Supérieur	>	A > B	vrai si A est supérieur à B.
Inférieur ou égal	<=	A <= B	vrai dans les deux cas, A inférieur à B ou A égal à B.
Supérieur ou égal	>=	A >= B	vrai dans les deux cas, A supérieur à B ou A égal à B.

Les programmes de recherche de données emploient ces opérateurs. Pour déterminer si la valeur cherchée se situe à gauche ou à droite de celle lue, la recherche dichotomique sur des archives triées implique une série de comparaisons effectuées au moyen des opérateurs relationnels > (supérieur), < (inférieur), = (égal). Le schéma en haut de la page 327 représente l'organigramme d'une recherche binaire (dichotomique) de données.
Les opérateurs relationnels peuvent être inclus dans des expressions qui contiennent aussi des opérateurs arithmétiques (∧, *, /, +, −), mais ces derniers sont prioritaires. Par exemple, dans l'expression 3 + 5 < 7 * 2, on exécute d'abord les calculs arithmétiques (3 + 5 et 7 * 2). L'expression équivaut à 8 < 14.

Opérateurs logiques

Les opérateurs logiques du Basic sont ceux déjà énoncés page 73. Leur priorité d'application est la suivante :

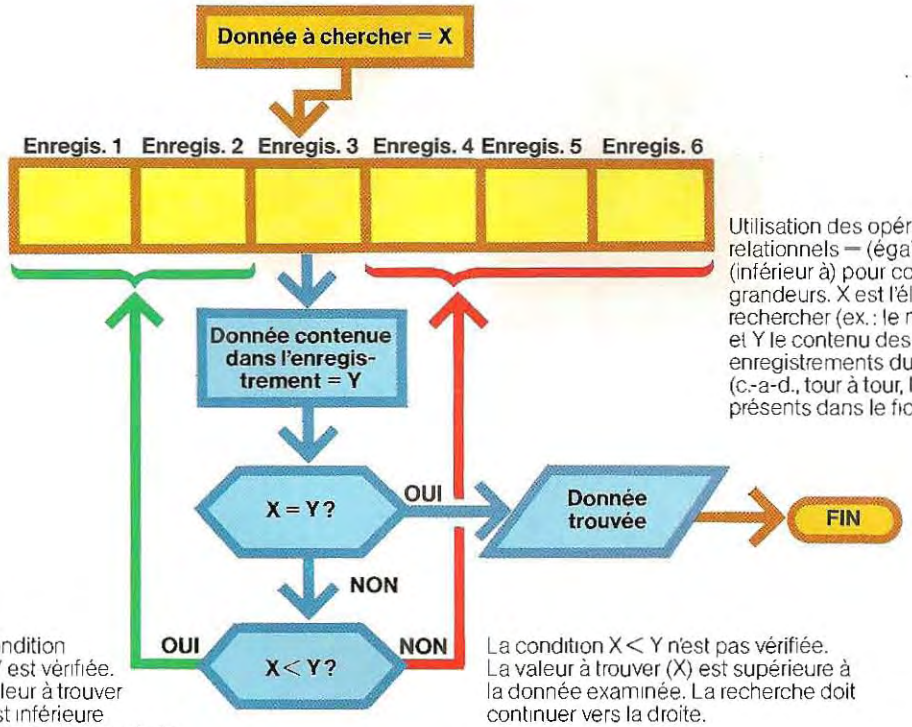
Opérateur	Données	Résultat
NOT	A = 1	NOT A = 0
AND	A = 1, B = 0	A AND B = 0
OR	A = 1, B = 0	A OR B = 1
XOR	A = 1, B = 0	A XOR B = 1

Outre ces quatre opérateurs d'usage fréquent, le Basic 80 en prévoit deux autres pas encore traités : IMP et EQU. Les tables de vérité correspondantes sont reproduites au bas de la page 327. Ces deux opérateurs ne figurent pas dans toutes les versions du Basic et ils sont rarement utilisés dans les programmes d'application.
Les opérateurs logiques peuvent s'appliquer à des nombres entiers (ceux que la machine peut contenir en mémoire sont compris entre − 32 768 et 32 767) et opèrent sur les bits dont sont composés les nombres. Les opérateurs arithmétiques, relationnels et logiques peuvent être employés dans la même expression. Par exemple, la condition :

```
(A+B)>C AND F=B
```

*Pour la représentation des nombres réels, on utilisera désormais le point décimal de préférence à la virgule. C'est la représentation standard en calcul informatique (par ex., 35.5 plutôt que 35,5).

EXEMPLE D'APPLICATION DES OPERATEURS RELATIONNELS A LA RECHERCHE DE DONNEES



Utilisation des opérateurs relationnels = (égal) et < (inférieur à) pour comparer deux grandeurs. X est l'élément à rechercher (ex.: le nom MARTIN) et Y le contenu des différents enregistrements du fichier (c.-a-d., tour à tour, les noms présents dans le fichier).

La condition X < Y est vérifiée. La valeur à trouver (X) est inférieure à la demande examinée. La recherche doit continuer vers la gauche.

La condition X < Y n'est pas vérifiée. La valeur à trouver (X) est supérieure à la donnée examinée. La recherche doit continuer vers la droite.

TABLES DE VERITE DES OPERATIONS IMP ET EQU

Opérateur IMP ⇒

A	B	A IMP B
1	1	1
1	0	0
0	1	1
0	0	1

L'opérateur IMP reconnaît la séquence 1 0 et donne 0 en sortie. Dans tous les autres cas, la sortie (résultat) est 1.

Opérateur EQU ⇔

A	B	A EQU B
1	1	1
1	0	0
0	1	0
0	0	1

L'opérateur EQU reconnaît le cas d'égalité des deux entrées (toutes deux 1 ou 0); dans ce cas, le résultat est 1, sinon 0.

La modélisation sur ordinateur

La mise en service d'un nouvel appareil relevant de technologies avancées exige de nombreux contrôles (parfois très poussés), nécessaires pour vérifier que la machine présente bien la fiabilité souhaitée.

Ainsi, les fabricants de plate-formes pétrolières, de moteurs d'automobiles, de réacteurs, ou de circuits électroniques sont tenus de prévoir avec la plus haute précision possible quel sera le comportement de leurs produits après leur sortie d'usine.

Ces appareils sont soumis à de véritables traitements de choc dans des conditions particulières destinées à mettre à l'épreuve leur résistance. Ces procédés coûtent d'ailleurs fort cher mais ils sont indispensables.

De nos jours, ces expériences peuvent être menées, non plus « sur le terrain », mais simulées par ordinateur. C'est là que le calculateur électronique se révèle d'une aide précieuse dans l'application de ces méthodes pouvant être mises en œuvre grâce à des modèles mathématiques reproduisant le comportement de l'appareil à l'essai.

Si les calculs révèlent l'existence de points faibles dans la structure de l'appareil, on modifie le modèle directement sur la table à dessin, et on recommence ainsi jusqu'à obtention des résultats qui correspondent au fonctionnement désiré.

Lorsqu'il s'agit d'équipements particulièrement complexes, comme une tour de forage pétrolier ou un avion, les modèles mathématiques de simulation s'avèrent eux aussi très compliqués, et les calculs deviennent tellement longs et fastidieux qu'ils sont confiés directement à l'ordinateur.

Le travail des concepteurs s'en trouve considérablement allégé et ces derniers peuvent ainsi passer plus de temps à concevoir de nouveaux modèles et à élaborer les méthodes appropriées pour les tester.

Cette utilisation de l'ordinateur qui permet de prévoir le comportement d'une structure se nomme ***simulation par ordinateur.***

Dans un plus vaste domaine d'application de l'électronique, c'est l'ordinateur lui-même qui dessine et modifie les détails d'un projet ; il le représente en deux ou en trois dimensions de façon à n'ignorer aucun de ses aspects.

Ces méthodes de « prototypage » se divisent en deux étapes principales : le traitement numérique des équations mathématiques qui décriront le comportement de la structure, et la visualisation des résultats du calcul, permettant au concepteur de pourvoir à la correction des erreurs.

Les résultats du traitement numérique sont communiqués au technicien, non pas en chiffres, mais sous forme de dessins quelquefois très élaborés.

Les systèmes de modélisation informatisée font une large utilisation de la ***table traçante*** *(plotter). Cet appareil électronique exécute le dessin à partir des instructions fournies par l'ordinateur ; les tracés apparaissent à l'écran. Quand une table traçante fait office de terminal de sortie d'un ordinateur, on peut disposer d'un enregistrement permanent du dessin.*

Lorsque les schémas sont très complexes, il est nécessaire de les faire apparaître sur la table traçante, le résultat visible sur l'écran étant trop peu précis pour être exploité. Notons toutefois qu'une image se dessine beaucoup plus rapidement sur l'écran que sur la table traçante, et qu'elle peut en outre être modifiée assez vite si l'on veut représenter l'objet sous d'autres angles. Ainsi, le déroulement du processus graphique peut être contrôlé à chaque étape.

La conception des circuits électroniques est un des domaines où l'ordinateur apporte à l'homme la contribution la plus appréciable. Les circuits de type numérique sont formés de milliers de ***portes logiques*** *qui contrôlent de faibles impulsions électriques permettant (ou non) au courant de traverser les différentes parties du circuit, suivant la présence (ou l'absence) d'autres impulsions. En fait, les circuits peuvent devenir extrêmement compliqués et, si le concepteur néglige le fonctionnement d'une seule de ces portes, le fruit de son travail sera anéanti par cette unique erreur. L'ordinateur peut, en revanche, manipuler avec aisance les milliers d'éléments sans commettre ni erreur ni oubli.*

Les modèles simulent aussi le fonctionnement des structures mécaniques.

Il serait en effet inimaginable de construire (par exemple) de véritables tours de forage à seule fin de vérifier leur fiabilité dans les vraies conditions de leur implantation future (pleine mer, tempête, etc.). D'autant plus que, dans certains cas, il faudrait peut-être attendre

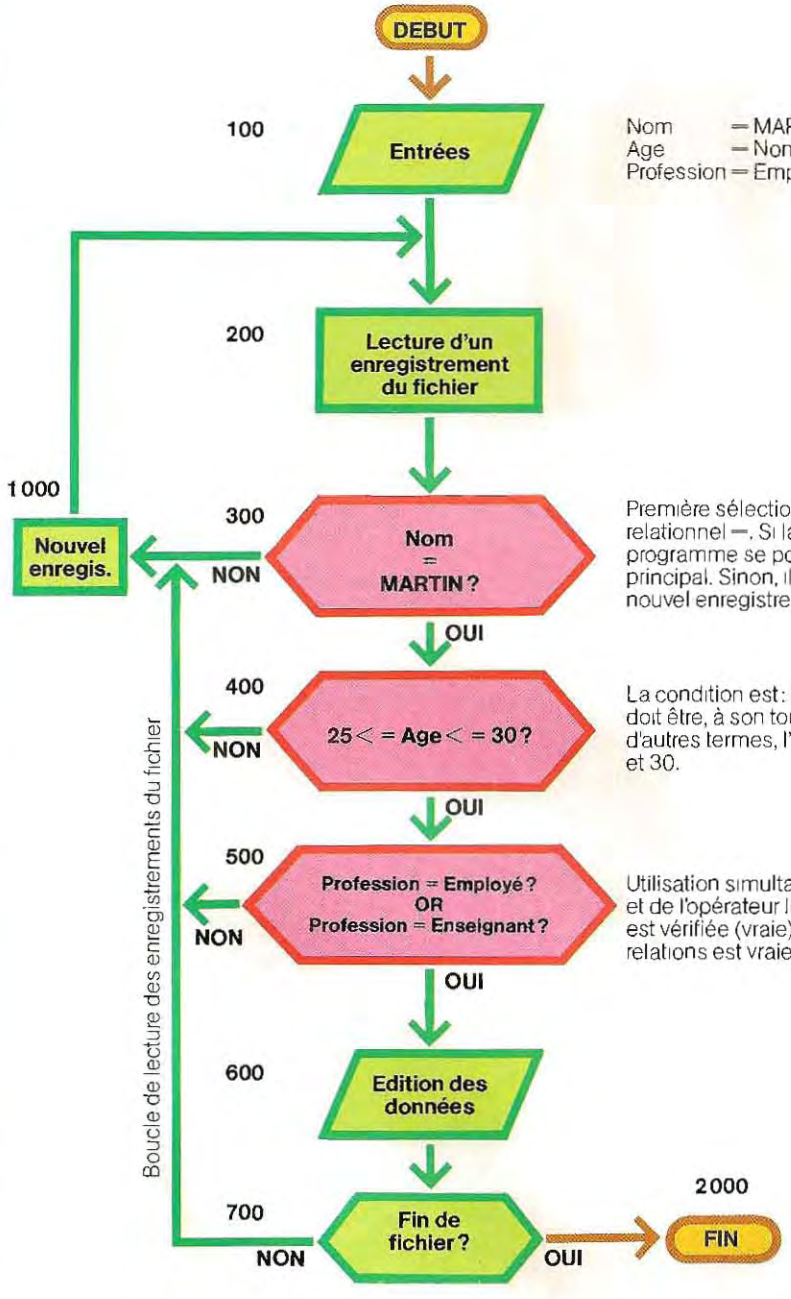
longtemps pour que soient effectivement rassemblées les conditions souhaitées.
La méthode par « éléments finis » est utilisée comme outil mathématique pour l'analyse des tensions auxquelles peuvent être soumis les appareils, qu'il s'agisse de l'ossature d'un avion ou de la simple dent d'un engrenage.
Les éléments finis permettent de schématiser les pièces par des formes élémentaires, triangles, rectangles ou lignes.
L'ingénieur introduit dans l'ordinateur les données décrivant le modèle de la structure à concevoir afin que la machine en construise les éléments finis. Ces éléments apparaissent alors à l'écran comme des sortes de toiles d'araignées dont l'entrecroisement de fils reproduit l'image de la structure réelle.
L'utilisateur peut demander à l'ordinateur une rotation du modèle afin de l'examiner sous un autre angle, ou un agrandissement de tel ou tel élément qui doit être étudié plus en détail.
Pour simuler la situation dans laquelle une charge est exercée sur l'un ou l'autre point de la structure à l'étude, le concepteur communiquera à la machine la valeur des pressions envisagées et leur point d'impact. La machine fera apparaître à l'écran les déformations consécutives de la structure, superposant éventuellement cette image de l'élément déformé à celle de l'élément intact.
L'ordinateur sait aussi analyser les effets produits sur la structure par des anomalies de fonctionnement ou par des situations accidentelles : on peut ainsi étudier ce qui se passerait, par exemple, si la plate-forme pétrolière perdait un de ses piliers de soutien.
Naturellement, les résultats ne sont intéressants que si les paramètres de base communiqués à la machine sont précisément ceux de la réalité.
La précision dans ce domaine sera d'autant améliorée qu'on aura recours à de petits éléments, produisant à l'écran des réseaux très serrés. Toutefois, l'emploi de ces réseaux revient très cher, leur mise au point étant longue et nécessitant aussi un long traitement de l'ordinateur. Ils ne seront donc pas toujours recherchés, d'autant plus que des réseaux à mailles larges procurent la plupart du temps

SDRC Engineering Services Ltd

Sur les deux premières images : visualisation à l'écran d'un bras articulé et du déroulement de son mouvement. Sur les autres : une pièce et différents réseaux représentant ses éléments finis.

des résultats suffisamment précis.
Les modèles informatisés se révèlent fort utiles dans l'étude des appareils en mouvement. Une application particulièrement spectaculaire en est donnée avec notre exemple de la plate-forme pétrolière. Dans la réalité, la tour est traînée (couchée sur le flanc) par un remorqueur jusqu'au lieu de son implantation. On doit ensuite la redresser en lui faisant accomplir un mouvement de 90° pour l'arrimer à l'endroit voulu. La traduction graphique de cette simulation est un ensemble de dessins reproduisant les positions successives de la tour lors de son installation. Les dessins sont tridimensionnels et on peut, à volonté, examiner la tour sous tous ses angles quand on le désire.
De la même façon, on utilise ces modèles pour des programmes d'étude des mouvements effectués par le corps humain dans différentes situations, et tout particulièrement, lors de la manœuvre d'une machine. Cela permet de construire des machines adaptées à leur emploi par l'homme. SAMMIE (sigle de System for aiding man-machine interaction evaluation, soit, en français, Système d'aide à l'évaluation de l'interaction homme-machine) est l'un de ces programmes. Il permet de dessiner, en trois dimensions, des personnages en situation, superposés à des machines, des appareils divers, ou des édifices. Les personnages peuvent se déplacer sur l'écran, s'asseoir ou se lever, et l'on peut ainsi vérifier de quelle manière ils réussissent à atteindre les diverses commandes d'une machine.
Le modèle (ayant cette fois la silhouette humaine) est réalisé de façon à se mouvoir exactement comme un être humain, c'est-à-dire disposant des mêmes articulations corporelles. Si, par erreur, un mouvement impossible pour l'homme était imposé au modèle (par exemple, la torsion excessive d'un bras, ou la rotation de la tête à 100°), la machine le signalerait.
De même que la stature humaine varie d'une personne à l'autre, celle du mannequin sera modifiée dans certains cas. Cependant, il est plus facile de concevoir un personnage de taille moyenne et de mener la plupart des essais en fonctions de ses mensurations.
L'étape suivante consiste donc à dessiner le milieu dans lequel le mannequin doit évoluer, que ce soit un bureau à cloisons mobiles, la cabine d'un tracteur agricole, l'intérieur d'une voiture, ou simplement une cuisine équipée de ses différents appareils électroménagers et de ses éléments de rangement. Une fois le décor en place, le concepteur peut, par le jeu de commandes simples, indiquer à l'ordinateur les déplacements ou les mouvements du mannequin, selon ce qu'il a à accomplir.
Les dessins réalisés par SAMMIE sont en trois dimensions, et le programmeur a toute possibilité de demander à l'ordinateur de lui montrer d'autres aspects de la scène. Il peut également obtenir des vues rapprochées de certaines parties du dessin, autorisant une étude plus en détail de certaines situations. Là aussi, l'ordinateur transmettra un message si une situation ne correspond pas à ce qu'un être humain pourrait effectivement faire. Le concepteur devra répéter son essai.
Tous les éléments du décor ou de l'habitacle pourront être déplacés, dessinés avec de nouvelles dimensions, ou conçus différemment selon les problèmes précis qui auront été mis en évidence. Ainsi, on changera le sens d'ouverture d'une porte de placard, on déplacera l'emplacement des commandes du tracteur pour que les manœuvres soient facilitées, on modifiera la ligne des sièges d'une automobile pour garantir une plus grande sécurité au conducteur.
Dans toutes ces applications, l'ordinateur, en assumant vite et bien tous les innombrables calculs de base et de corrections, offre la possibilité de répéter les essais autant de fois qu'il le faut tant qu'on n'a pas obtenu toute la fiabilité nécessaire à l'appareil étudié, et permet de réaliser ensuite des produits d'une très haute précision, et réunissant tous les critères souhaitables de sécurité.
En outre, il laisse à l'ingénieur tout le temps utile à sa recherche de création. Le fait que l'ordinateur puisse ensuite donner l'analyse de situations fictives vient encore encourager l'imagination du concepteur qui se consacre à l'étude de matériels de plus en plus performants. La généralisation de la modélisation informatisée conduira à l'élaboration de prototypes qui, bénéficiant toujours davantage des progrès de l'informatique, apporteront à leur tour des données de plus en plus sophistiquées au travail de l'ordinateur, accélérant ainsi le progrès technique dans son ensemble.

UTILISATION SIMULTANEE D'OPERATEURS RELATIONNELS ET LOGIQUES



est vraie (elle donne comme résultat – 1), si la somme est A + B supérieure à C et si, en même temps, la valeur de F égale celle de B. Le schéma ci-dessus illustre un exemple d'utilisation simultanée de différents types d'opérateurs. Le problème examiné dans l'exemple traite de l'impression statistique du contenu d'un fichier. On veut imprimer les données de tous les individus dont le nom est Martin, dont l'âge est compris entre 25 et 30 ans inclus et qui sont employés ou enseignants.

Opérateurs fonctionnels

On peut transformer une séquence de calculs en **fonction**, représentée par la suite des opérations à effectuer sur une donnée opérande. Ainsi, ce calcul du montant correspondant à 18 % d'un coût donné :

$$\text{montant} = \frac{\text{coût} \times 18}{100} = \text{coût} \times 0.18$$

peut être transformé en une fonction qui consiste à multiplier par 0.18, le coût donné. Mathématiquement, la fonction est représentée ainsi :

$$\text{montant} = f(\text{coût})$$

Le symbole f (coût), signifie, en ce cas, la multiplication : coût × 0.18. Un symbolisme très voisin est adopté en Basic. Le symbole d'une fonction est formé de deux lettres de la fonction. Ainsi le symbole FNA représente une fonction de nom A. Dans l'exemple précédent l'opérande est coût et la formulation complète de la fonction est :

FNA (coût)

où FNA (coût) n'est autre que le produit coût × 0.18. Une fois définie, la fonction peut être utilisée dans un calcul quelconque. Ainsi, si les 18 % représentent une remise, pour calculer le prix on devra d'abord calculer le montant de la remise et le déduire du coût (prix sans remise) :

$$\text{remise} = \frac{\text{coût} \times 18}{100} = \text{coût} \times 0.18$$

prix = coût − remise = coût − coût × 0.18.

Ayant défini FNA (coût) comme montant de la remise, on a :

Définition (DEF) :
remise = FNA (coût) (soit coût × 0.18)

Calcul
prix = coût − FNA (coût)

Ce genre de fonction est dit **fonction utilisateur** car elle n'est pas définie à priori dans le Basic et doit donc être définie par le programmeur.

L'instruction de définition d'une fonction doit être établie avant l'utilisation de cette fonction, sinon le système émet un diagnostic de fonction inconnue et s'arrête dans l'attente d'une correction. Pour indiquer au système l'existence de cette nouvelle fonction FNA (coût), on a besoin de l'instruction de définition DEF FNA (soit définition de la fonction A). Comme dans presque toutes les instructions Basic, le symbole indique le nom même de l'opération à exécuter. L'opération étant la **définition**, le code Basic correspondant est DEF.
Pour définir la fonction de l'exemple précédent, l'instruction est :

DEF FNA (coût) = coût * 0.18

soit
- codage de l'instruction de définition
- symbole indiquant une fonction à la machine
- nom de la fonction
- paramètre
- calcul effectué par la fonction

Les fonctions définies par l'utilisateur peuvent être de trois types :

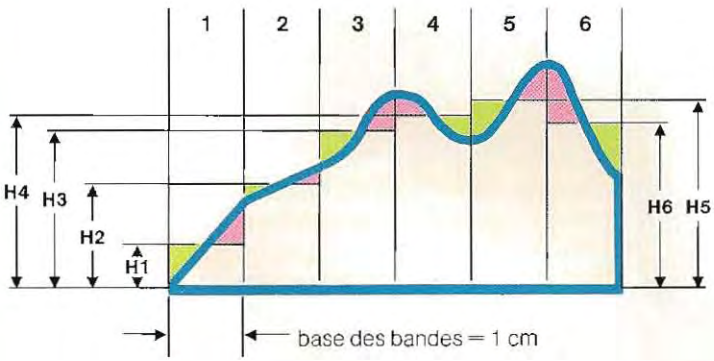
- **réelles :** toutes celles qui opèrent sur des nombres réels, c'est-à-dire formés d'une partie entière et d'une partie décimale.
- **entières :** toutes celles qui n'opèrent que sur des nombres entiers.
- **de chaîne :** les fonctions opérant sur des caractères (lettres ou chiffres).

Les caractéristiques de ces trois types et la façon de définir les fonctions dans les trois cas énumérés seront exposées par la suite. En plus des fonctions définies par l'utilisateur, il existe des fonctions de bibliothèque, déjà prêtes, qui exécutent les opérations mathématiques les plus courantes. Ces fonctions définies au niveau système, sont « appelées » (c'est-à-dire utilisées) par un nom symbolique (défini dans le Basic), sans la contrainte du sigle FNX (X = nom de la fonction).
Par exemple, la fonction qui détermine la racine carrée d'un nombre a pour nom le sigle SQR (abréviation de square root = racine carrée) ; ainsi pour extraire la racine du nombre 1 673 on devra écrire :

SQR (1673)
code paramètre

Si l'on veut comprendre pourquoi un ordinateur requiert une fonction (et non un calcul direct) pour le calcul d'une racine, il faut analyser la structure propre à la machine. Le système exécute les calculs en utilisant les microprogrammes (microcodes) contenus dans l'UAL (unité arithmétique et logique). Les signaux qui représentent les opérandes (les nombres) et les opérateurs (les opérations à exécuter) y sont interprétés, puis transmis à des circuits spéciaux qui fournissent les résultats en sortie.
Toutefois ces circuits ne sont en mesure d'exécuter que quelques types de calculs par

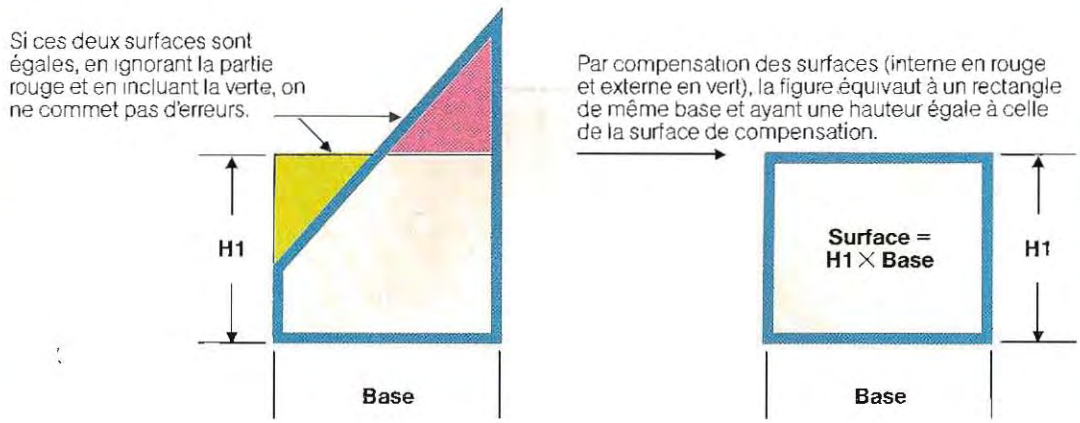
PROCEDE GEOMETRIQUE DE CALCUL DES SURFACES



La figure plane (ligne bleutée) est divisée en un certain nombre (6) de bandes verticales. La surface totale est la somme des surfaces élémentaires, de chaque bande ; cette surface élémentaire s'obtient en multipliant la base par la hauteur correspondante (H1, H2, etc).

complément, essentiellement l'addition et la soustraction. Les opérations de multiplication et de division sont exécutées par des microprogrammes habituellement extérieurs à l'UAL. On dit que ces opérations sont exécutées par logiciel, car elles sont indépendantes des circuits. Tous les autres calculs, par exemple la racine carrée, sont exécutés au moyen des quatre opérations fondamentales.
A titre d'exemple, appliquons le concept de fonction à un programme de calcul de surfaces. Le problème est le suivant : étant donné une figure plane délimitée par une droite horizontale, deux droites verticales et une courbe, calculer la surface de la figure.
La première étape consiste à diviser la figure en un certain nombre de bandes verticales

METHODE DE COMPENSATION POUR LE CALCUL DES SURFACES



(voir le schéma de la page 333) ayant toutes une base égale (par ex., 1 cm). Chaque bande peut être considérée comme un rectangle. On choisit la hauteur du rectangle (H1 sur le schéma ci-dessus), de façon que la zone de la figure exclue du calcul (superficie rouge) soit équivalente à une nouvelle surface supplémentaire (zone verte). De cette façon, tout en excluant une partie de la figure (rouge), on en inclut une autre (verte), de même surface, mais qui ne faisait pas partie de la figure auparavant. En d'autres termes, la surface de chacune des bandes verticales est obtenue comme s'il s'agissait d'un rectangle ayant comme hauteur H1, H2, etc. La subdivision en un certain nombre de bandes s'avère nécessaire pour minimiser l'erreur qui serait commise si l'on dessinait arbitrairement la droite de compensation (hauteur moyenne) sur la totalité de la base. Après avoir préparé le dessin comportant la mention des différentes hauteurs, on peut dessiner l'organigramme du programme. Les fonctions à développer sont les suivantes :

1 / demande du nombre de divisions verticales (6 divisions sont un minimum).
2 / demande de la dimension de la base (dans l'exemple 1 cm).
3 / boucle de recherche des hauteurs (en nombre égal au nombre des bandes), calcul des surfaces de chacun des rectangles et de leur somme.
4 / présentation du résultat (somme des surfaces élémentaires).

Le calcul le plus souvent employé est celui de la surface d'un rectangle ; pour le développer rapidement, on peut définir une fonction opérant sur les deux paramètres base et hauteur.

surface = base × hauteur

La fonction correspondante (par exemple nommée A) est :

DEF FNA (base, hauteur) =
= base × hauteur

soit
- instruction de définition
- symbole de fonction
- nom de la fonction
- paramètres
- calcul.

Le schéma de la page 336 présente l'organigramme de ce calcul. Cette méthode est connue comme « intégration graphique d'une fonction ». La courbe qui sous-tend la surface que l'on cherche à connaître est une fonction (au sens mathématique du terme) et le calcul de la surface est une méthode de calcul de l'intégrale de la fonction, étendu à l'intervalle de base. L'intégration représente un des aspects les plus complexes des mathématiques, au point que le calcul d'une intégrale par des méthodes analytiques peut être impossible. Dans les calculs scientifiques, la méthode que nous venons d'exposer est employée couramment, même quand il s'agit d'une forme qui permet d'éliminer la partie graphique à la charge de l'utilisateur.

Test 9

1 / Ecrire l'instruction qui, en mode interactif, permet de calculer la moyenne des trois nombres 5.2, 15.21, 7.43, de façon que le résultat apparaisse à l'écran et sur imprimante.

2 / Ecrire le résultat des expressions suivantes (attention à la priorité des opérateurs) :
3 + 2 AND 3 × 4 ; 5 + (2 AND 3) × 4

3 / Imaginer un fichier contenant les archives d'une bibliothèque de 1 000 volumes (1 000 enregistrements). Les données sont distribuées dans les champs (ou zones) suivants :

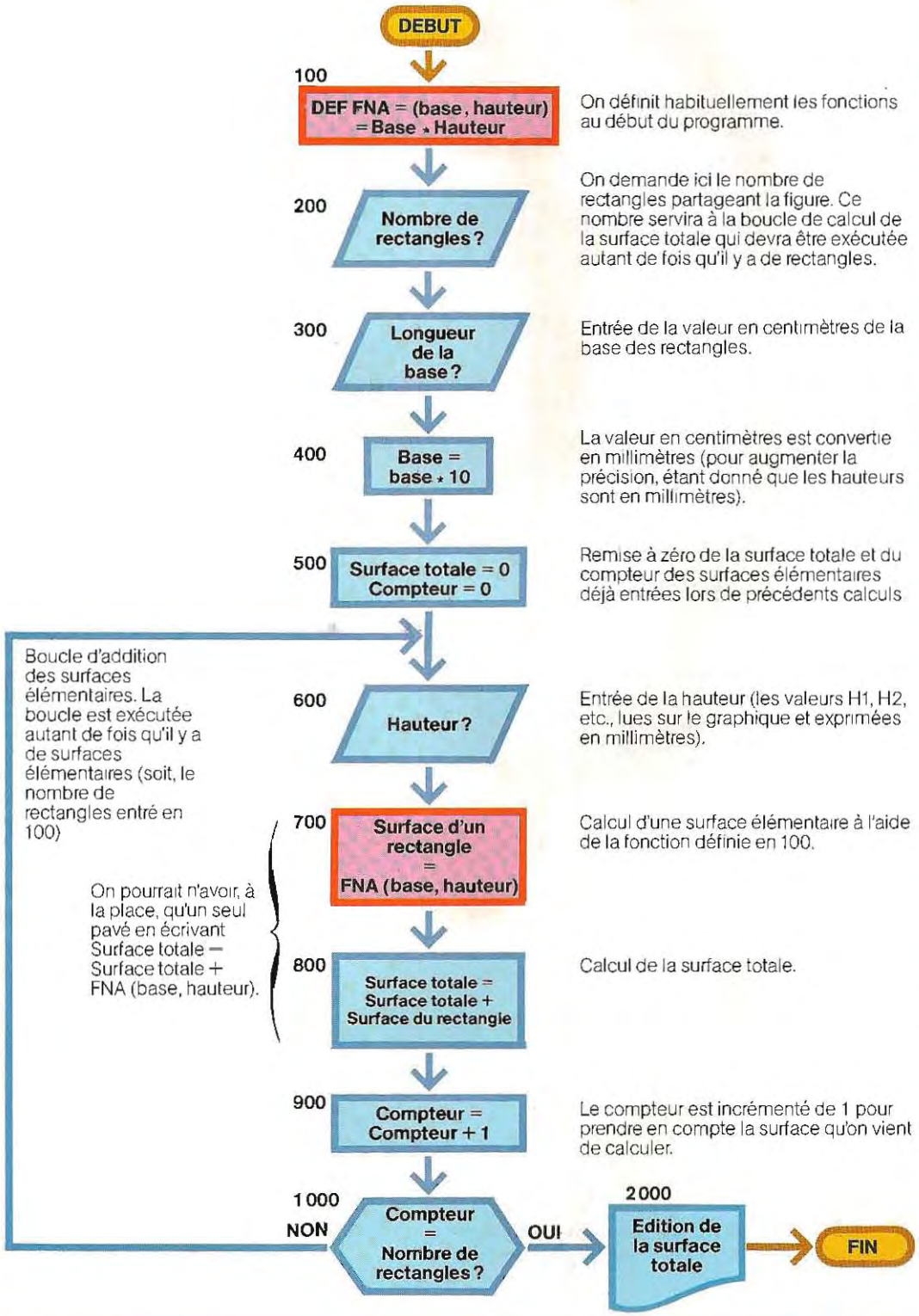
1 / Titre du livre	30 caractères
2 / Sujet	10 caractères
3 / Spécialité	10 caractères
4 / Langue (code)	3 caractères
5 / Disponibilité	1 caractère

Les rubriques (champ n° 2) représentent une subdivision à un niveau général (par exemple histoire, géographie, etc.), la spécialité (champ n° 3) est une subdivision du sujet (pour l'histoire, ce sera, par exemple, la préhistoire, l'époque romaine, etc.). Le champ n° 4 est relatif à la langue dans laquelle l'ouvrage est écrit (FRA = Français, ITA = Italien, etc.).
La disponibilité (champ n° 5) est un indicateur qui peut valoir 0 si le volume a été prêté, ou 1 s'il est présent dans la bibliothèque.
Dessiner l'organigramme d'un programme recherchant tous les livres présents (non prêtés) écrits en français (FRA) ou en anglais (ANG), et traitant de poésie. Le sujet est Art, la spécialité Poésie.

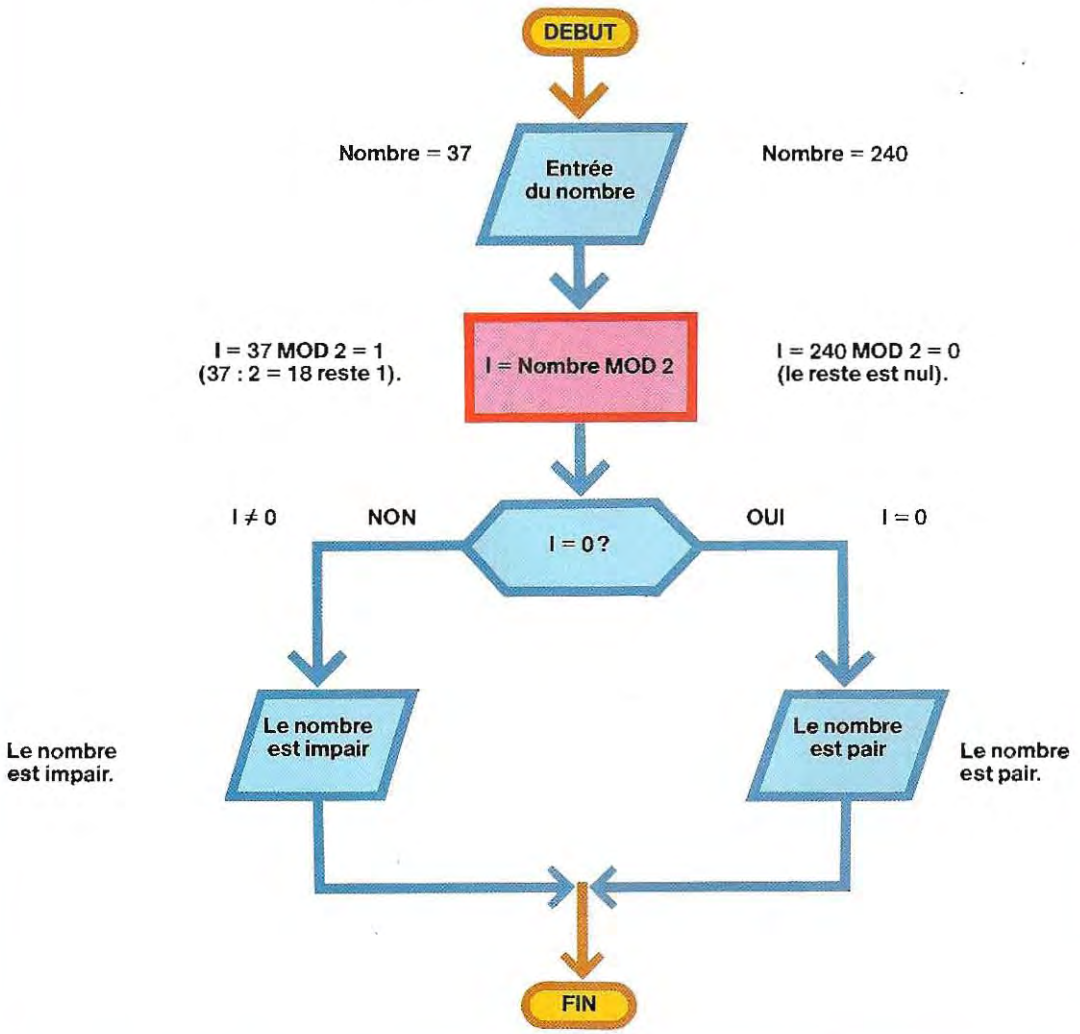
4 / Combien parmi les volumes contenus dans les archives (question n° 3) peuvent être stockés sur une disquette de 1 Moctet ?

Les solutions du test se trouvent pages 350 et 351.

ORGANIGRAMME DE CALCUL DES SURFACES



UTILISATION DE L'OPERATEUR MOD POUR DETERMINER LA PARITE D'UN NOMBRE



Opérateurs de division entière

Dans la division entière indiquée en Basic par le symbole \, les opérandes sont arrondis à une valeur entière avant l'exécution de la division, dont le résultat est lui-même un nombre entier. Ainsi :

35.7 \ 4.7 est évalué à 36 / 5 = 7

Si la conversion des opérandes conduit à un résultat dépassant les limites d'un entier (— 32 768, + 32 767), une erreur de débordement est diagnostiquée. Par exemple, (75.4 * 56 621) \ 5.6 ne peut être calculé car le numérateur excède la valeur 32 767.

Opérateur modulo

L'opérateur modulo (MOD) fournit une valeur entière égale au reste de la division de 2 nombres.
Ainsi :

7 MOD 2 = 1 car 7 : 2 = 3 reste 1
9 MOD 3 = 0 car 9 : 3 = 3 reste 0

L'opérateur MOD permet de s'assurer qu'un nombre est divisible par un autre : dans ce cas le résultat de MOD est 0.
Le schéma ci-dessus présente l'organigramme d'un programme déterminant la parité d'un nombre.

En résumé :
Le Basic possède trois modes opératoires :
– mode Commandes : pour l'entrée des commandes.
– mode Exécution : exécution interactive de programmes et de calculs.
– mode Editeur : pour d'éventuelles modifications de programmes.
Les opérateurs du Basic (par ordre de priorité) sont :
– arithmétiques : les opérations mathématiques habituelles (∧, changement de signe, *, /, +, −) ;
– relationnels : égal, supérieur, inférieur et leurs combinaisons ;
– logiques : les opérateurs logiques déjà énoncés dans le chapitre qui leur est consacré, plus deux autres, disponibles dans certaines versions ;
– fonctionnels : fonctions définies par l'utilisateur (DEF FNX) ou fonctions système ;
– division entière et modulo : opérateurs \ et MOD.
Le programmeur peut définir une fonction quelconque en utilisant la notation : DEF FNX (X = nom de la fonction, une lettre quelconque). Le Basic met à la disposition de l'utilisateur toute une bibliothèque de fonctions mathématiques déjà prédéfinies, et qu'il suffit d'appeler par leur nom symbolique.

Le contrôle s'effectue en vérifiant la divisibilité de ce nombre par 2 (si le nombre est divisible par 2, il est pair, sinon il est impair).

Constantes et variables

Comme dans tout langage symbolique, les données peuvent être de deux types : **constantes** et **variables.**
Les constantes ont une valeur fixe, alors que les variables prennent des valeurs qui peuvent changer au cours du traitement.
Les variables sont représentées par des noms symboliques, les constantes sont directement désignées par leur valeur.

Par exemple, si l'on veut calculer la circonférence d'un cercle, la formule est : circonférence = 2 * π * R. Le symbole π est une constante numérique (on verra qu'il existe un autre type de constante : la chaîne de caractères), tandis que le symbole R (rayon du cercle) est une variable, pouvant prendre une valeur quelconque (les chaînes peuvent également être des variables).

Constantes numériques

Il existe cinq types de constantes numériques en Basic :
1 / entières (integer) : représentant un nombre entier (et son signe) compris entre − 32 768 et + 32 767 (ou − 32 768 ÷ + 32 767), suivi du symbole %.
2 / réelles en virgule fixe (fixed point) : ce sont des nombres réels (contenant des décimales) et associés à des signes négatifs ou positifs (on les appelle des nombres décimaux signés).
3 / réelles en virgule flottante (floating point) : ce sont des nombres réels en représentation exponentielle.
La représentation exponentielle permet d'exprimer un nombre à l'aide de puissances de 10. Ainsi, le nombre 1 400 peut être écrit $14 * 100 = 14 * 10^2$.
De la même façon, le nombre 121.46 devient $12146 / 100 = 12146 / 10^2 = 12146 \times 10^{-2}$.
On doit se souvenir qu'écrire 1 : 100 équivaut à 1 : 10^2, soit encore 10^{-2}.
En Basic, le format à utiliser pour l'expression d'un nombre en virgule flottante est : Nombre (mantisse) E (symbole d'exposant) Exposant.
Par exemple :

Valeur numérique	Notation exponentielle	Notation en Basic
1400	14×10^2	14 E 2
1400	1.4×10^3	1.4 E 3
1400	0.14×10^4	0.14 E 4
121.46	12146×10^{-2}	12146 E − 2
121.46	1.2146×10^2	1.2146 E 2

Les trois représentations du nombre 1400 sont équivalentes.

4 / hexadécimales (HEX) : ce sont des nombres en représentation hexadécimale (base 16). Pour indiquer à la machine que l'on entend utiliser la notation hexadécimale, le

nombre doit être précédé des symboles &H. Ainsi, la valeur de nombre &H70 est 112 en décimal (70 en hexadécimal valant $7 \times 16 + 0 = 7 \times 16 = 112$).

5 / octales (OCT) : il s'agit de nombres représentés en base 8, et précédés du symbole & (« et » commercial), ou, sur certaines machines, &0 (lettre « o »). Ainsi, le nombre &70 vaut 56 en décimal.

Les notations octale et hexadécimale sont proches de la symbolisation binaire de l'ordinateur et on y aura recours pour toute référence à une adresse mémoire. Pour extraire le bit d'une donnée de nom X, on appliquera l'opérateur AND entre la donnée et un « masque », nombre binaire dont seul le bit 8 est égal à 1.

On convertit ce masque en octal :

Valeur	4 2 1	4 2 1	4 2 1
Binaire	1 0	0 0 0	0 0 0
Octale	2	0	0

10 000 000 (binaire) = 200 (octal)

Dans le programme, la fonction d'extraction s'écrira X AND & 200.

Précisions

Les nombres entiers sont représentés en machine sur 16 bits (2 octets), aussi leur taille est-elle limitée aux valeurs décimales que l'on peut écrire sur 16 bits, soit à l'intervalle $-32768, +32767$ ($-2^{15}, +2^{15}-1$). En fait les bits utilisés pour décrire le nombre ne sont qu'au nombre de 15, le bit numéro 16 indiquant le signe du nombre (négatif si le bit 16 vaut 1, positif si le bit 16 vaut 0). Rappelons qu'en binaire, pour passer d'un nombre négatif à sa valeur absolue, il faut prendre son complément à 1, puis ajouter 1. Ainsi, pour connaître la valeur du nombre négatif 1111110011000011 (négatif, car le bit 16 est à 1), il faut d'abord compléter à 1 chaque bit.

La démarche est la suivante :

valeur d'origine	1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1
complément	0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0
	+
ajouter 1	1
	1 1 0 0 1 1 1 1 0 1

On obtient : $1\,1\,0\,0\,1\,1\,1\,1\,0\,1 = 2^9 + 2^8 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 829$.

Les nombres réels peuvent être mémorisés en simple ou double précision. En simple précision, on utilise 32 bits et le nombre peut aller de 1×10^{-38} à $1 \times 10^{+38}$, avec 7 chiffres significatifs.

Par exemple, le nombre $135.794397 \times 10^{25}$ est en réalité tronqué à 135.7943×10^{25}.

Les sept chiffres significatifs sont utilisés pour les calculs, mais à l'affichage on n'en retient que 6.

La simple précision est indiquée par le symbole !. Par exemple, 1.35 E2 et 135! sont deux notations du même nombre en simple précision.

Pour accroître le nombre de chiffres significatifs, et donc la finesse des calculs, on emploie la double précision. Dans ce format, le nombre sera représenté sur 64 bits, et pourra varier de 1×10^{-38} à $1 \times 10^{+38}$, avec 16 chiffres significatifs (15 à l'affichage).

On symbolise un nombre en double précision comme un nombre en simple précision, en remplaçant le « E » par un « D ».

Le nombre 135.794397E25, en simple précision perd ses derniers chiffres (9 et 7) ; écrit ainsi, 135.794397D25, il est en double précision et on ne perd aucun chiffre.

Les nombres en simple ou double précision sont affichés avec un chiffre de moins que dans le format de stockage en mémoire, car le Basic effectue automatiquement un arrondi à la valeur la plus proche.

Le symbole D définit un nombre en double précision sous forme exponentielle ; si l'on ne veut pas utiliser l'exponentiation, on doit se servir du symbole # (dièse).

Ainsi, 372.0, en simple précision, s'écrit 372.0 # en double précision.

Ici, les deux notations 372.0 et 372.0 #, sont identiques (aucun chiffre ne déborde du format en simple précision) ; cependant, si dans le programme, des calculs associent ces nombres et des valeurs en double précision, il est conseillé d'utiliser une notation homogène.

Le tableau en haut de la page suivante résume les différents formats de représentation des nombres en machine.

En résumé, les différents types de constantes numériques rencontrées en Basic sont :

PRECISION DE LA REPRESENTATION DES NOMBRES

Type	Espace mémoire	Valeurs limites	Nombre de chiffres significatifs	Nombre de chiffres affichés	Exemple
Entier	2 octets	$-32768 \div +32767$	5	5	21743
Simple précision	4 octets	$\pm 10^{-38} \div \pm 10^{+38}$	7	6	121.3E7
Double précision	8 octets	$\pm 10^{-38} \div \pm 10^{+38}$	16	15	275.8D9 127.41 #
Hexadécimal	2 octets	$0 \div$ FFFF			&H57
Octal	2 octets	$0 \div 177777$			&75

Illustrons cela par quelques exemples :

Valeur d'entrée	Signification
42745 %	Erreur. Un nombre entier ne peut être supérieur à 32767.
&12*5	Le résultat est 50 car le nombre octal &12 correspond à 10 décimal.
&12 + &H12	Résultat : 28 (&12 = 10 &H12 = 18 ; le symbole H indique que le nombre est codé en hexadécimal.
1357.92486E21	Les deux derniers chiffres (8 et 6) sont perdus car le nombre est en simple précision.
1357.92486 #	Tous les chiffres sont conservés car le symbole # indique la double précision.
13.5792486D2	Représente la même valeur que précédemment en notation exponentielle.
579.3E41	Erreur. L'exposant en simple précision doit être compris entre −38 et +38.
579.3D41	Représentation exacte du nombre précédent.
157.2!	Le symbole ! indique une valeur en simple précision.
157.2	Autre représentation du nombre précédent. En l'absence du symbole #, le nombre est en simple précision, même si le symbole ! manque.

Constante	Symbole	Exemple
Entière	%	127 %
Exponentielle	E	127E2 = 12700
Double précision	D	127D2 = 12700
Double précision	#	12700 #
Octale	&	&21
Hexadécimale	&H	&H21

Variables numériques

Les variables sont des grandeurs, représentées par un nom symbolique, dont la valeur peut changer en cours d'exécution d'un programme.

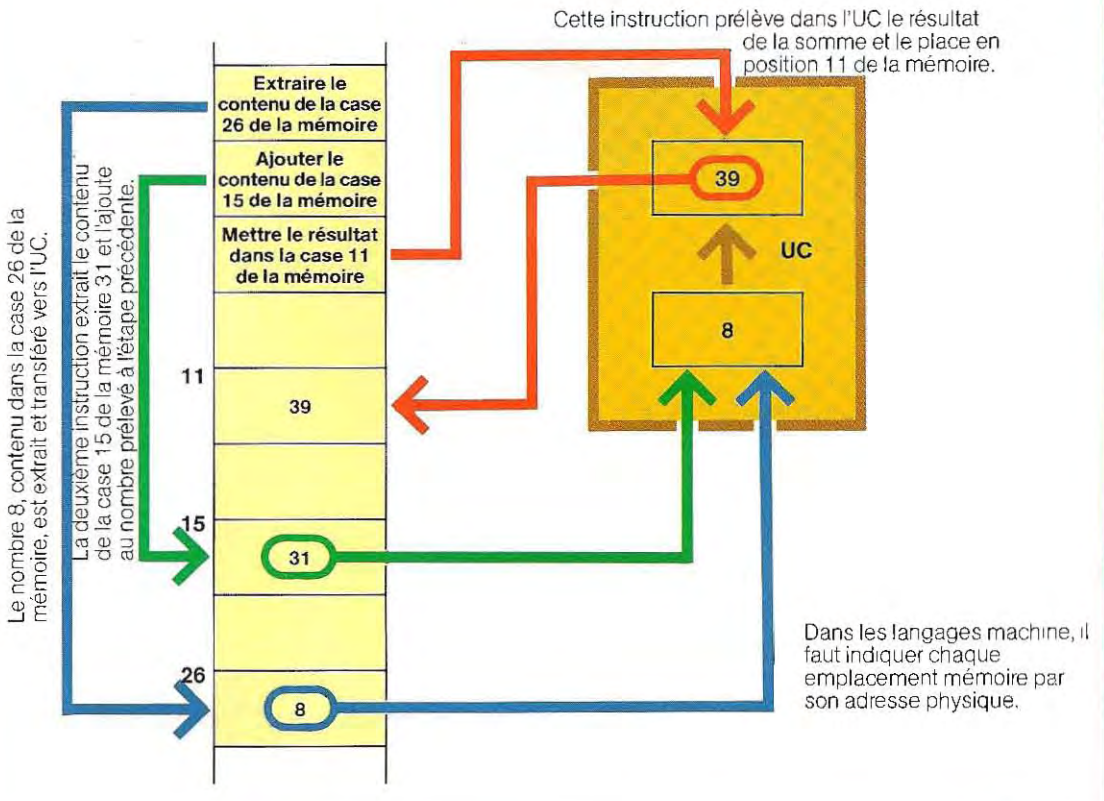
Dans les versions les plus simplifiées du Basic, les noms de variables ont deux caractères au maximum.

Des noms de longueur supérieure à deux caractères sont tronqués automatiquement (troncature automatique) et réduits à leurs deux premières lettres.

Ils risquent alors d'être confondus avec des noms différents mais commençant par les deux mêmes lettres.

Par exemple, les noms MOIS et MOYENNE, définissant deux variables différentes sont tous deux réduits, dans les versions simplifiées du Basic, au terme MO ; lors de l'exécution du programme, les deux variables prendront les mêmes valeurs, amenant à de graves erreurs dans les calculs.
Le nom d'une variable est équivalent à une adresse mémoire. La mémoire de la machine contient toutes les instructions des valeurs sur lesquelles elles opèrent : dans une première zone, résident les codes associés aux instructions, dans la seconde, les valeurs. Pour indiquer la valeur à utiliser il faut fournir l'adresse (position de la case mémoire qui la contient). Par exemple, pour additionner les nombres 31 et 8, les instructions (voir schéma ci-dessous), doivent faire référence aux adresses mémoire contenant ces valeurs (respectivement 15 et 26) ; de même, il faudra donner l'adresse où doit être stocké le résultat. Dans les langages de haut niveau, les

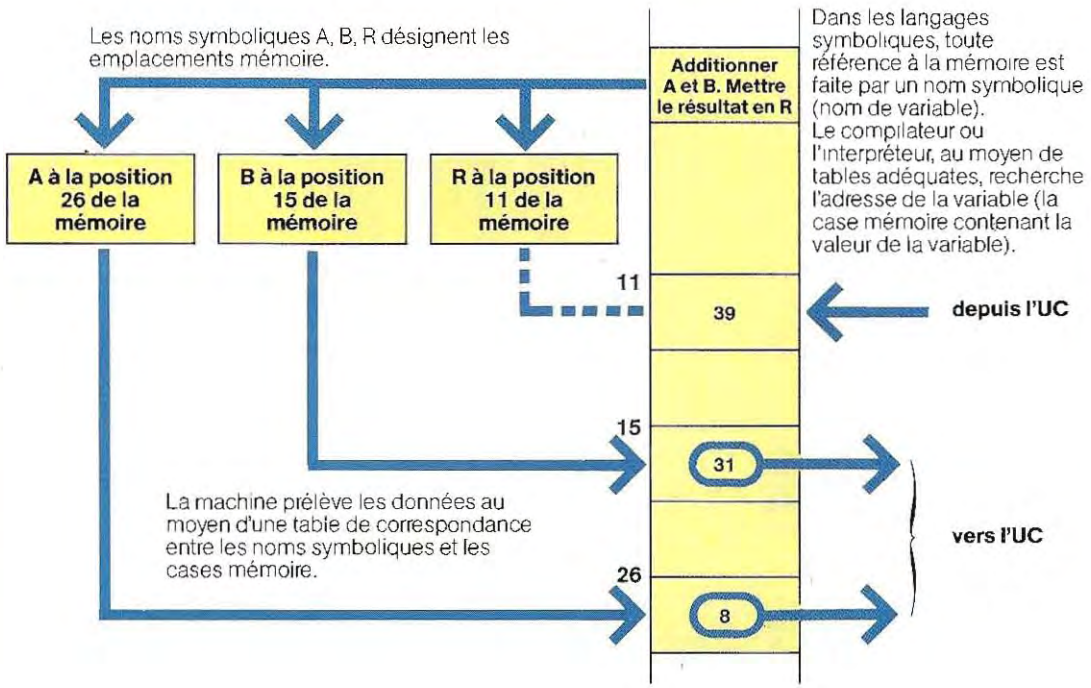
REFERENCE MEMOIRE PAR ADRESSAGE DIRECT



adresses sont représentées par des noms symboliques (A, B, R, dans notre exemple) et le système crée une **table de correspondance** associant à chaque nom symbolique du programme une adresse en mémoire.
Le programmeur, sans se préoccuper des positions mémoires des données, désignera ces dernières par leur seul nom symbolique. L'interpréteur Basic prendra en charge la recherche d'adresses.
En Basic (comme dans d'autres langages évolués) l'addition décrite précédemment s'écrit simplement R = A + B. L'interpréteur identifie les symboles + et =, et traduit l'instruction (R = A + B) en une séquence appropriée d'instructions en format machine, et ce en recherchant les adresses des variables dans la table de correspondance (voir schéma ci-dessous).
Les variables peuvent être, comme les constantes numériques :

- **entières** (symbole % accolé au nom)
- **en simple précision** (symbole !)
- **en double précision** (symbole #).

REFERENCE MEMOIRE PAR NOMS SYMBOLIQUES



L'occupation mémoire et la précision (le nombre de chiffres significatifs) sont les mêmes que pour les constantes numériques. On attribue sa valeur à une variable par une instruction d'assignation, (A = 21, B = 185, etc.), ou bien par un calcul (A = 3 * B + 2).
En début de programme, toutes les variables sont initialisées à zéro et conservent cette valeur tant qu'elles ne sont pas utilisées. C'est ce que figure le bloc 500 de l'organigramme de la page 336, qui remet à zéro les variables Surface totale et Compteur.
Cette réinitialisation est automatiquement exécutée au début du programme mais généralement il est bon de l'insérer en vue de futures modifications dans le programme.
En l'absence du module 500, si on modifiait (par ex.) le programme par une boucle, pour calculer successivement deux surfaces, le second calcul serait faux du fait de l'absence de remise à zéro. La première fois, au démarrage du programme, la mise à zéro serait faite automatiquement ; mais la seconde fois, les valeurs se cumuleraient aux précédentes.

Exemples de variables

R! = 356.8 — La variable R, définie en simple précision (symbole !), reçoit la valeur numérique 356.8.

A! = 37E20*8E20 — Faux. La variable A est définie en simple précision alors que le résultat du calcul ne peut être contenu que dans une variable en double précision :
37E20 = 37×10^{20}, 8E20 = 8×10^{20} et le produit $37 \times 10^{20} \times 8 \times 10^{20}$ donne 296×10^{40}, soit 296E40 :
L'exposant E40 ne peut être accepté en simple précision (le maximum étant 38).

B! = 5 / 2E9 — Faux. Le résultat de 5 divisé par 2E9 (c'est-à-dire 2000000000) est trop petit pour être accepté en simple précision (7 chiffres significatifs qui seront tous nuls).

C% = 721.3 — Faux. Le symbole % définit la variable C comme entière et il ne peut donc contenir de valeurs décimales.

C! = 721.3 — Exact. On peut également utiliser la notation C = 721.3, dans la mesure où l'absence de symboles implique la simple précision.

R = 1240.2145# — Faux. A une variable R en simple précision on ne peut affecter un nombre en double précision (symbole #) car certains chiffres du nombre sont perdus.

Conversion des différents types de variables

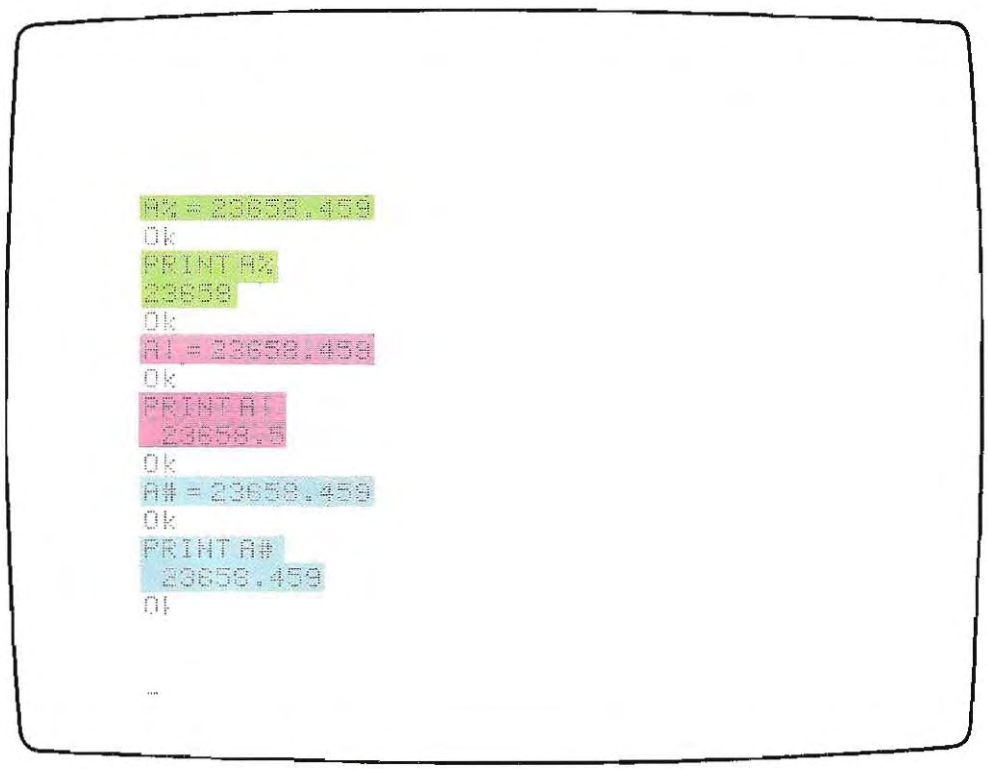
Au cours de l'exécution d'un programme, il peut s'avérer nécessaire de convertir une variable d'un type à un autre. C'est durant cette conversion que sont effectués des arrondis ou des troncatures de valeurs numériques. Les conversions ont lieu lors :

- d'instructions d'affectation : A = 275
- des calculs : A = 3.21 * 75.8 + B#
- d'opérations logiques : (5.2 + 4) AND (3.7 + 2)

Arrondis. Dans les procédures de calcul automatique, l'exactitude des résultats numériques est fortement influencée par les arrondis que la machine exécute systématiquement sur les données et les résultats eux-mêmes.
Aussi est-il essentiel de connaître, a priori, l'ordre de grandeur de l'erreur à laquelle on peut s'attendre.
La première source d'erreur réside dans l'arrondi exécuté par la machine lors de la phase d'affectation des constantes. On la minimi-

PRECISION DES VARIABLES

Sur l'écran on voit différentes assignations typiques pour une même valeur A, chacune étant suivie par une phase d'édition.



La variable A est entière.

La variable A est définie en simple précision.

La variable A est définie en double précision. Cette phase d'édition met en évidence un nombre de chiffres significatifs supérieur aux cas précédents, bien qu'une même valeur numérique ait été attribuée.

sera en utilisant la précision la plus appropriée.

Arrondis dans les affectations. La valeur numérique est mémorisée selon le format défini pour la variable.
Par exemple :

A! = 152.6D3 La valeur 152.6E3 est affectée à la variable déclarée en simple précision, et les chiffres au-delà du 7e sont perdus (par exemple 1359.743298D4 devient 1359.743E4).

A% = 28.3 Dans la variable entière A%, seule est transférée la partie entière du nombre ; le résultat est A% = 28.

A# = 562.5E4 La variable est en double précision (#) alors que le nombre est en simple précision : les chiffres significatifs non explicitement déclarés dans le nombre sont mis à zéro.

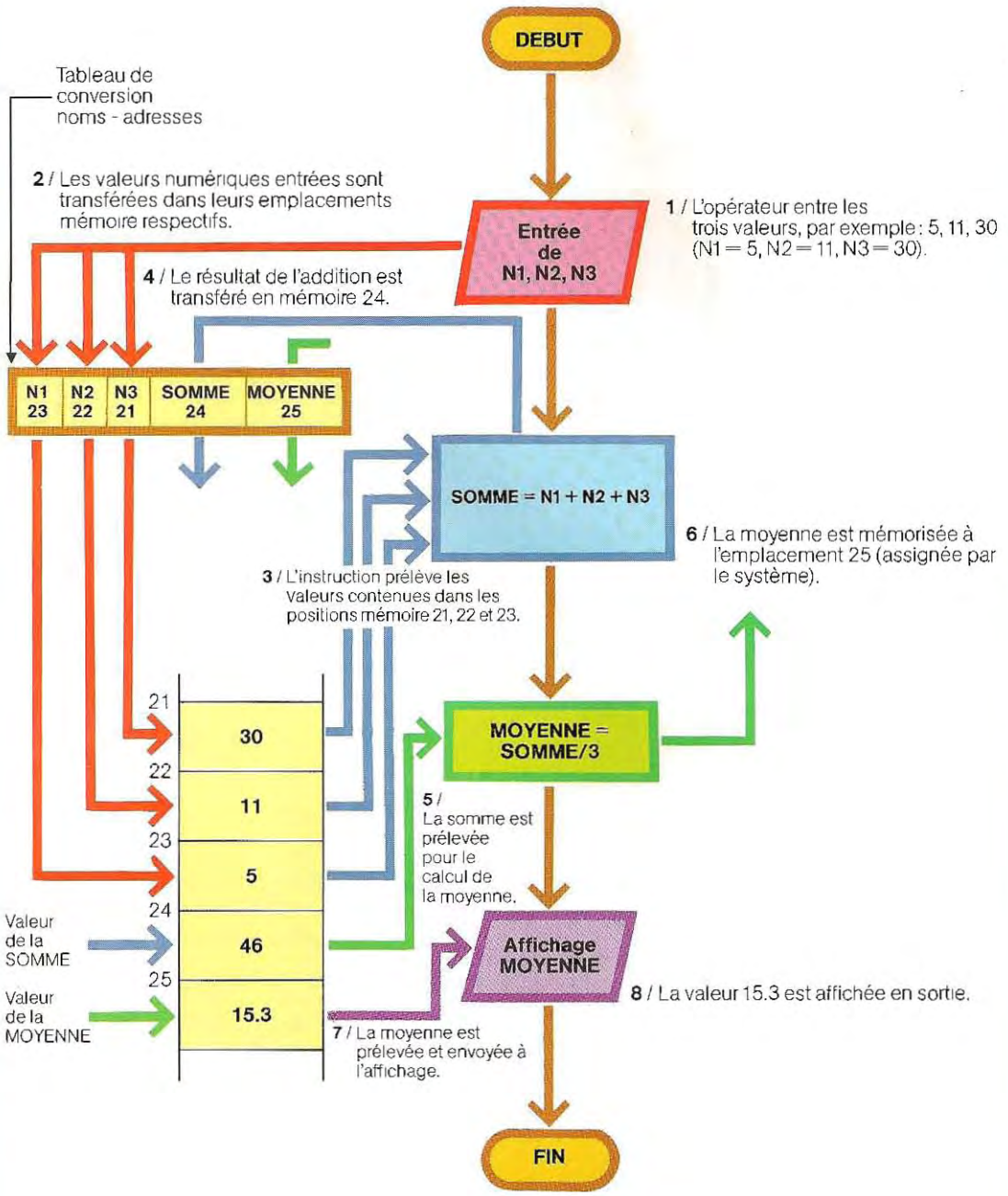
A% = 28.7 La décimale étant supérieure à 5, l'arrondi se fait à la valeur supérieure. Le résultat est A% = 29.

Arrondis dans les calculs. Dans les expressions, tous les opérandes sont évalués avec la même précision, à savoir, la plus haute en cours. Par exemple, l'expression A# = (7.56! + 8.26E4)/121D7 est calculée comme si tous les nombres étaient en double précision, comme l'opérande 121D7. Le résultat aussi est mémorisé en double précision.
En revanche, l'expression A! = (7.56! + 8.26E4)/121D7 (égale à la précédente) mémorisée en simple précision, celle de la variable A!.

En résumé :

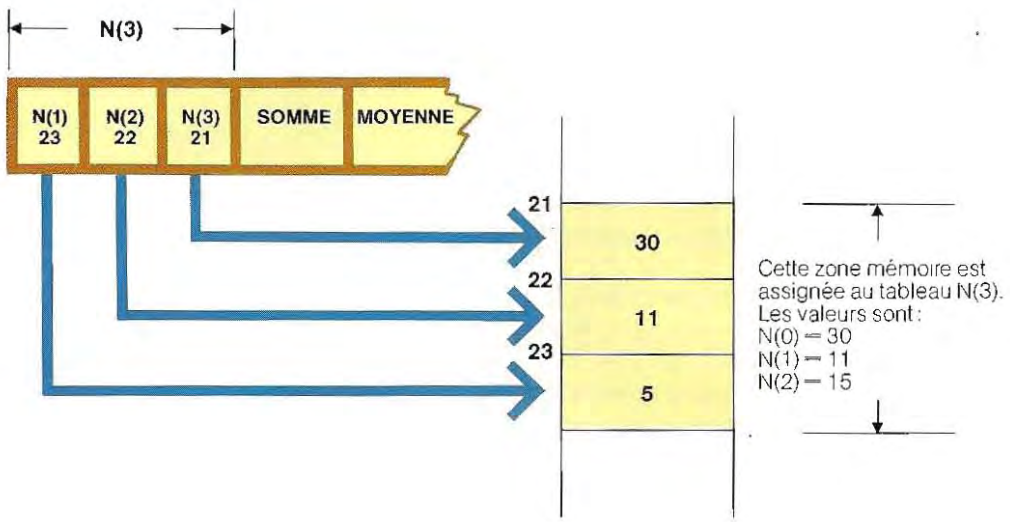
SI L'OPERATEUR VEUT...	IL DOIT ENTRER...
passer sous Basic	MBASIC (ou BA, ou autre)
définir une variable entière	le nom de la variable suivi du symbole % (exemple : A%, VALEUR%).
définir une variable en simple précision	le nom de la variable suivi du symbole ! (exemple : C!).
définir une variable en double précision	le nom de la variable suivi du symbole # (exemple : R #).
définir un nombre entier	la valeur du nombre sans décimales
définir un nombre en simple précision	la valeur du nombre suivie du symbole ! (exemple : 1265.3! ou, en notation exponentielle, 12.753E2).
définir un nombre en double précision	la valeur du nombre suivie du symbole # (exemple 1275.3 #, ou en notation exponentielle, 12.753D2).
définir une fonction	DEF FNX = ... où X est le nom de la fonction (une lettre).
déterminer le reste de la division de deux nombres	reste = Premier nombre MOD Second nombre, exemple 7 MOD 3
faire un calcul immédiat	PRINT calcul, exemple PRINT (7 + 3)/4
affecter une valeur à une variable	A = 7, TOTO = 121.3.
passer sous système d'exploitation	SYSTEM

ADRESSAGE MEMOIRE PAR PLUSIEURS NOMS SYMBOLIQUES



Mécanismes de traduction entre les noms symboliques des variables (N1, N2, N3, SOMME, MOYENNE) et les emplacements mémoire contenant les valeurs numériques de ces variables.

ADRESSAGE MEMOIRE PAR UN SEUL NOM SYMBOLIQUE



Plutôt que trois variables N1, N2, N3, on peut définir un seul nom N les contenant, (occupant 3 emplacements mémoire).
Le tableau symbolisé par N(3), a une « dimension de 3 ».

Arrondis dans les opérations logiques. Les opérandes qui apparaissent dans une expression logique sont convertis en nombres entiers. Par exemple, l'expression (5.2 + 4) AND (3.7 + 2) est évaluée 9 AND 6 (3.7 + 2 = 5.7 arrondi à 6). Si la conversion génère un nombre hors de l'intervalle − 32768, + 32767 il y a dépassement de capacité (overflow).

Les variables structurées

En programmation, on distingue deux types de variables : les variables simples et les variables structurées. Dans le second cas, le logiciel du système offre la possibilité de se référer à un ensemble de données distinctes, tout en utilisant un nom commun. Cela permet de réserver pour une variable structurée un certain nombre d'emplacements mémoire, où seront mémorisées, dans l'ordre, les composantes de la variable. Les variables structurées les plus simples, existant en Basic, sont les **tableaux** (array) et les **chaînes de caractères** (string).

D'autres types de structures plus complexes seront exposés par la suite.

Tableaux. Par ce terme, on désigne un ensemble de zones mémoire auxquelles on fait référence par le même nom symbolique. Imaginons un programme calculant la moyenne de trois nombres introduits au clavier.
Les fonctions à exécuter sont :

- lecture des trois nombres ;
- calcul de leur moyenne (somme des trois nombres/3) ;
- affichage du résultat.

Le schéma de la page 346 représente l'organigramme correspondant, en désignant les trois nombres par N1, N2, N3. Ces noms symboliques sont associés à trois zones mémoire distinctes. A la saisie (entrée) des valeurs numériques à affecter aux trois variables, le système réserve des adresses mémoire et y stocke les valeurs entrées au clavier.

Les trois variables N1, N2, N3 peuvent également être appelées par le nom unique N(3), le symbole (3) indiquant qu'à ce nom correspondent trois emplacements mémoire (voir schéma de la page 347). La variable N(3) ainsi dimensionnée peut contenir simultanément plus d'une valeur (ici, trois). Pour faire référence à l'une d'elles, il faut indiquer son numéro. La numérotation interne des tableaux partant de zéro, on a :

N(0) = premier élément du tableau ; correspondant à N1 ;
N(1) = deuxième élément, correspondant à N2.
N(2) = troisième élément, correspondant à N3.

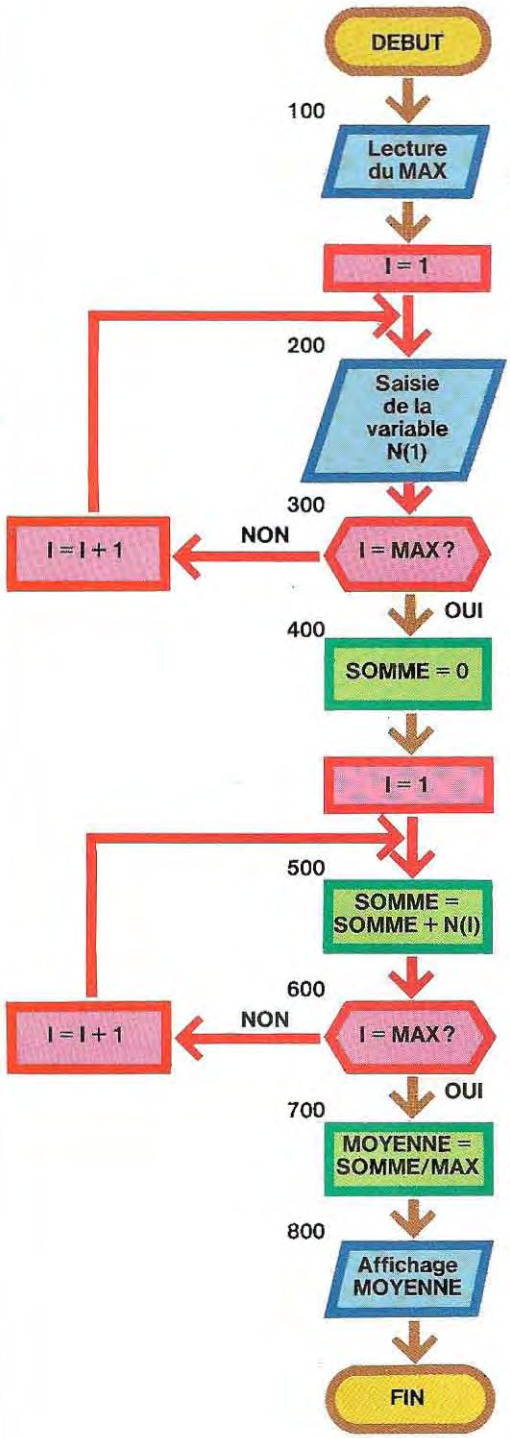
La notation N(3) sert, lors de la définition de la variable, à indiquer au système qu'il existe trois valeurs (toutes sous le nom N), alors que les symboles N(0), N(1), N(2) sont utilisés pour extraire ou écrire chacune des valeurs qui constituent le tableau N(3).
Appeler le premier élément d'un tableau par le nombre 0 peut sembler contraire à notre logique, c'est pourquoi, en Basic, une instruction est prévue, qui modifie la base de numérotation en la faisant débuter à 1.
Ainsi les éléments du tableau précédent deviennent N(1), N(2), N(3).
Pour faire référence à un élément d'un tableau, on peut « indiquer » le paramètre qui le repère. Par exemple, l'élément N(2), qui vaut 11, peut être appelé N(1) avec 1 = 2. En faisant varier l'indice 1 (1 = 1, 2, 3), on appelle tous les éléments du tableau. Cette méthode, très utilisée, permet de « paramétrer » les programmes.
Dans le calcul de la moyenne (exemple précédent) les instructions utilisées sont :

SOMME = N1 + N2 + N3
MOYENNE = SOMME/3
dans le cas de variables séparées.
SOMME = N(1) + N(2) + N(3)
MOYENNE = SOMME/3
dans le cas d'un tableau.

Si le nombre des variables change, (si par exemple il passe à 8), ces instructions ne sont plus valables. Il faut pratiquement récrire le programme. Avec un tableau indicé, le programme est paramétré et s'adapte sans aucune modification aux différentes applications. Le schéma de la page 349 présente l'organigramme (paramétré) du calcul de la moyenne d'un nombre quelconque de valeurs.
La limite de ce nombre est celle de la dimension de la mémoire fixée pour le tableau N. Avec N(3000), on peut calculer la moyenne d'un nombre de variables compris entre 2 et 3 000 (on ne calcule pas la moyenne d'une seule variable) ; le programme s'adapte alors automatiquement.

Les chaînes de caractères. Une chaîne est une suite de caractères alphanumériques (en code ASCII) délimitée par des guillemets. Ainsi, « NOM, 1, 3, 5 » est une chaîne qui contient les caractères NOM, 1, 3, 5. Les nombres figurant dans les chaînes de caractères ne sont pas représentés en format numérique. A moins de les convertir, on ne peut donc s'en servir pour effectuer des calculs.
Les chaînes de caractères peuvent être des constantes ou des variables, la distinction étant la même que pour les constantes et les variables numériques. Le symbole $ (dollar) informe le système que tel nom de variable désigne une chaîne de caractères (NOM$, A$, par exemple). La longueur maximale d'une chaîne, variable ou constante, est de 255 caractères. Si, par exemple, A$ = « ADC », la chaîne de nom A$ contient les caractères A, D et C. On appelle « chaîne vide » une chaîne qui ne contient aucun caractère. Par exemple lorsque A$ = « », le nom A$ désigne une chaîne vide. Notons que la chaîne A$ = « » n'est pas vide puisqu'elle contient un caractère, le blanc, ou espace, (en anglais, blank ou space).
La longueur d'une chaîne varie avec le nombre de caractères qu'elle contient. La chaîne définie en un point du programme par NOM$ = « Ravel », contient 5 caractères. Si, par la suite, on lui affecte une nouvelle valeur, NOM$ = « Hector Berlioz », elle aura alors une longueur de 14 caractères. Cette caractéristique, propre au Basic, s'appelle l'« allocation dynamique ». Son intérêt est de ne pas obliger le programmeur à définir à l'avance la longueur de chaque chaîne. Cette commodité présente toutefois un risque.
C'est le cas lorsqu'un long programme occupe la quasi totalité de l'espace mémoire

EMPLOI D'UN TABLEAU PARAMETRE POUR LE CALCUL D'UNE MOYENNE



Le programme demande le nombre de valeurs à saisir pour calculer leur MOYENNE. Ce nombre (MAX) ne doit pas dépasser la dimension prévue pour le tableau [N(3000), et donc MAX < = 3000]. Prenons, par exemple, MAX = 128.

Dans cette boucle, le programme demande les valeurs des variables. Il commence à I = 1 et affecte donc la valeur saisie à la première variable [N(I) — N(1) quand I = 1]. La boucle s'achève quand I = MAX = 128. Il faut donc absolument entrer les 128 valeurs annoncées.

Remise à zéro de SOMME.

La boucle cumule les 128 valeurs dans la zone SOMME. Au départ, on a posé SOMME = 0.
Au premier tour (I = 1), on a donc SOMME = SOMME + N(1) — N(1). Au tour suivant, (I = 2), on a SOMME = SOMME + N(2) = N(1) + (N2).

On additionne ainsi les 128 nombres.

On calcule la moyenne de cette façon :

$$\text{MOYENNE} = \frac{\text{Somme des valeurs}}{\text{Nombre des valeurs}} = \frac{\text{SOMME}}{\text{MAX}}$$

Affichage du résultat.

Solutions du test 9

1 / La moyenne de trois nombres est le quotient de leur somme par 3. On l'obtient à l'écran par l'instruction :
PRINT (5.2 + 15.21 + 7.43)/3
et sur l'imprimante par :
LPRINT (5.2 + 15.21 + 7.43)/3

---

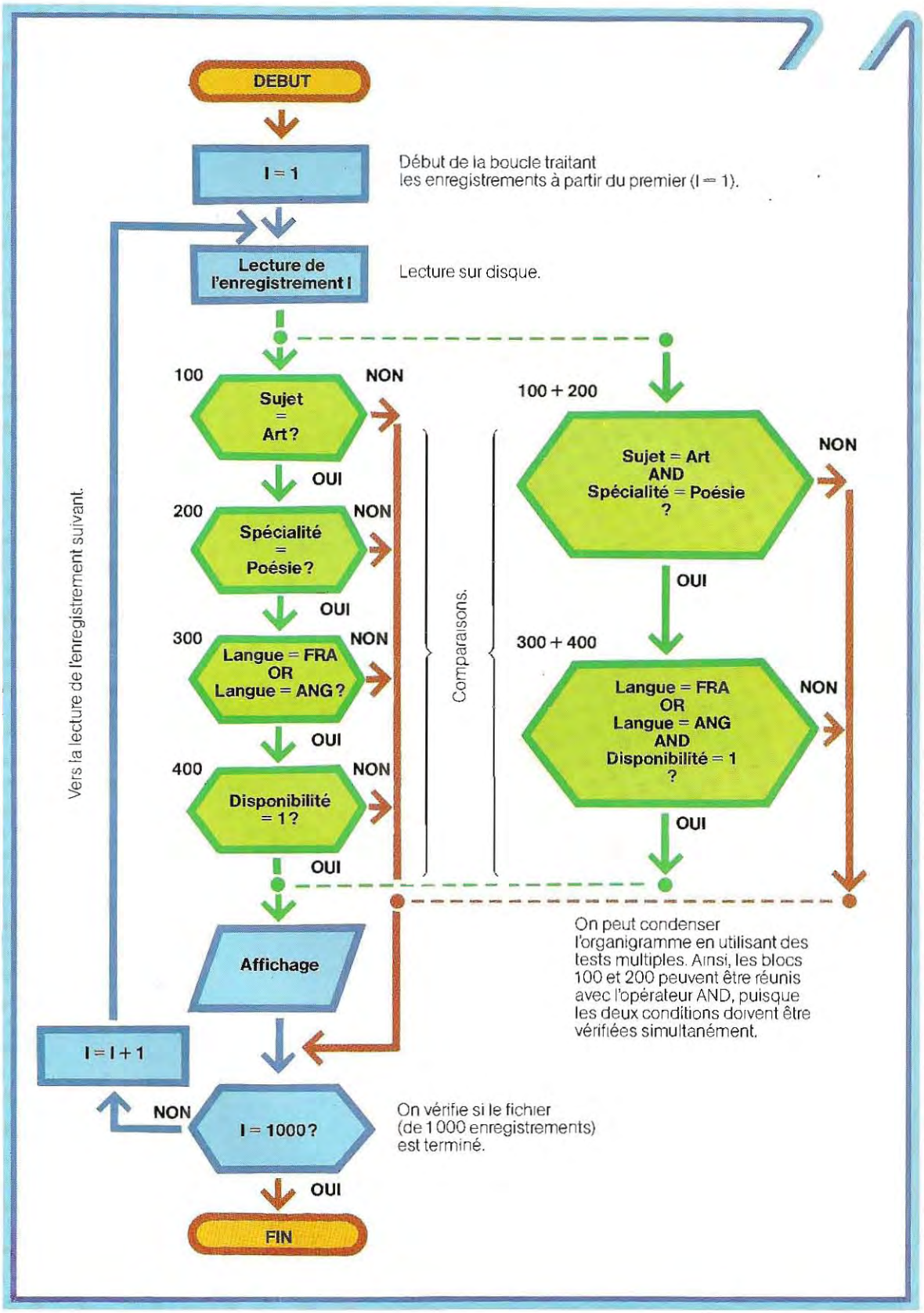
2 / L'opérateur logique AND a une priorité inférieure à celle des opérateurs arithmétiques. C'est pourquoi, dans l'expression 3+2 AND 3×4, les calculs (3+2 et 3×4) sont effectués en premier. On obtient l'expression 5 AND 12. Il faut alors convertir 5 et 12 en binaire, et leur appliquer l'opérateur AND.
5 décimal = 101 binaire
12 décimal = 1100 binaire
5 AND 12 = 0100 = 4 décimal
On a donc 3 + 2 AND 3 × 4 = 4
Dans la seconde expression, les parenthèses modifient l'ordre de priorité. La première opération, 2 AND 3, donne 10 AND 11 = 10 en binaire, soit 2 en décimal. L'expression devient donc :
5 + 2 × 4 = 13

---

3 / Il faut effectuer les opérations suivantes :
1 - lire un enregistrement du fichier de données ;
2 - regarder si le sujet est Art ;
3 - regarder si la spécialité est Poésie ;
4 - regarder si le livre est rédigé en français ou en anglais (FRA OR ANG) ;
5 - vérifier si le livre est en rayon (disponibilité = 1) ;
6 - vérifier si les conditions 2, 3, 4 et 5 sont remplies ; afficher le titre de l'ouvrage ;
7 - lire un nouvel enregistrement (si le fichier n'est pas terminé).
L'organigramme de cette succession de traitements se trouve à la page 351.

---

4 / La longueur d'un enregistrement correspond à la somme des caractères des différentes zones, soit, pour cet exemple, 30 + 10 + 10 + 3 + 1 = 54 caractères (rappelons qu'un caractère correspond à un octet). Si l'on fixe cette longueur à 70 pour conserver une marge, un disque d'un Mo (1 000 000 de caractères) pourra contenir 1 000 000 / 70, soit 14 285 enregistrements. Il est donc théoriquement possible de mémoriser les données concernant 14 000 volumes.
Toutefois, il ne faut pas surestimer les possibilités du système : les temps de traitement seraient très longs pour un tel fichier. L'accès au disque requiert, en moyenne, 0.1 seconde par enregistrement (notamment pour le positionnement de la tête). La lecture de 14 000 enregistrements dure donc environ 25 minutes, auxquelles il faut ajouter le temps nécessaire à l'exécution des instruction proprement dites. On peut estimer à 30 minutes la durée totale du traitement.
En outre, l'impression (pour le cas où on voudrait imprimer la liste de ces titres) est souvent la phase la plus longue. Un bon modèle d'imprimante (micro-ordinateur) a une vitesse de 120 lignes à la minute. Si on décide de faire figurer dans le listing seulement 10% des ouvrages, on imprimera 1 400 lignes (une par ouvrage), ce qui représente 1 400 / 120 = 11 fois 1 minute, soit 110 minutes. C'est-à-dire qu'il faudrait presque 2 heures pour imprimer toutes les données du fichier. La durée de l'exécution de ce programme peut donc varier entre 45 minutes et 3 heures.


DEBUT
I = 1
Début de la boucle traitant les enregistrements à partir du premier (I = 1).
Lecture de l'enregistrement I
Lecture sur disque.
100
Sujet = Art ?
NON
OUI
200
Spécialité = Poésie ?
NON
OUI
300
Langue = FRA OR Langue = ANG ?
NON
OUI
400
Disponibilité = 1 ?
NON
OUI
Comparaisons.
100 + 200
Sujet = Art AND Spécialité = Poésie ?
NON
OUI
300 + 400
Langue = FRA OR Langue = ANG AND Disponibilité = 1 ?
NON
OUI
Affichage
I = I + 1
NON
I = 1000 ?
OUI
FIN
Vers la lecture de l'enregistrement suivant.
On peut condenser l'organigramme en utilisant des tests multiples. Ainsi, les blocs 100 et 200 peuvent être réunis avec l'opérateur AND, puisque les deux conditions doivent être vérifiées simultanément.
On vérifie si le fichier (de 1 000 enregistrements) est terminé.

disponible et que plusieurs chaînes de caractères sont prévues en entrée (les noms et les adresses d'un répertoire, par exemple). Tant que les données saisies n'occupent pas trop de place en mémoire, l'exécution se déroule normalement. Mais si l'opérateur introduit une donnée trop longue dont l'allocation demande plus de place qu'il n'en reste en mémoire, l'exécution s'interrompt. Pour éviter ces incidents, on contrôle le nombre de caractères entrés, et sans interrompre le programme, on signalera toute longueur excessive à l'opérateur, qui abrégera alors la donnée.

Il est possible de dimensionner les chaînes, comme les variables numériques. En écrivant NOM$(10), on réserve l'espace mémoire nécessaire à 10 chaînes, chacune d'une longueur comprise entre 0 (chaîne vide) et 255 caractères. Chaque caractère occupant un octet en mémoire, l'espace mémoire total occupé varie donc de 0 (toutes les chaînes sont vides) à 10 × 255 = 2 550 octets. Mais il est impossible, à cause de l'allocation dynamique, de déterminer à l'avance l'espace réellement occupé.

On peut appliquer aux chaînes quelques opérateurs :

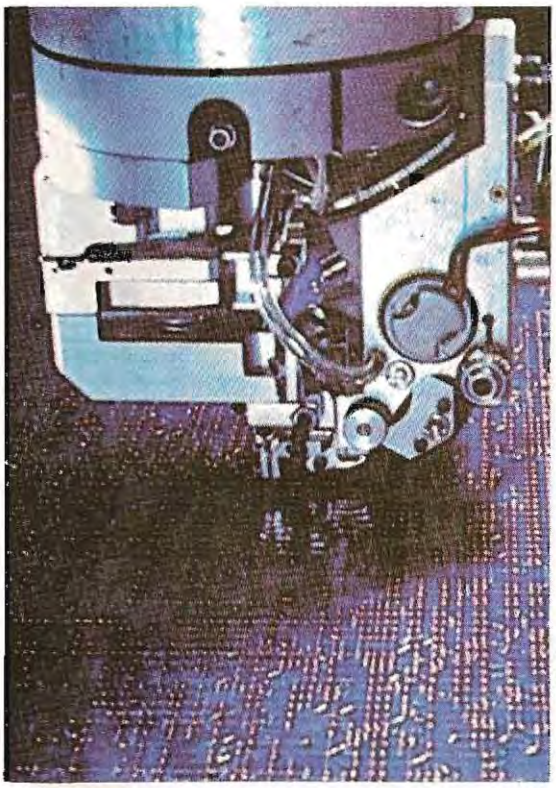
- **l'opérateur arithmétique de somme** (+), qui permet la concaténation de deux ou plusieurs chaînes. Ainsi, lorsqu'on pose : A$ = « Hector », B$ = « Berlioz », C$ = B$ + A$, C$ contient « Berlioz Hector ». Les chaînes B$ et A$ ont été mises bout à bout pour constituer une troisième chaîne : elles ont été concaténées.
  Sur certaines machines, le symbole de concaténation est & ; on écrira alors C$ = B$&A$.
- **tous les opérateurs logiques** (=, <>, <, >, <=, >=). Ainsi la condition NOM$ < « Berlioz » est vérifiée par toutes les chaînes dont la valeur lexicographique est inférieure à Berlioz (Bach, par ex.) et elle est fausse pour toutes les autres (Brahms, ou Mendelsshon, etc.). La valeur d'un caractère est fonction de sa place dans l'alphabet.

Les blocs 300 et 500 (organigramme de la page 331) constituent un exemple d'utilisation des chaînes.

Dans le bloc 300, on doit vérifier l'égalité entre le nom lu dans le fichier et la valeur « Martin ». Le nom est une variable de type chaîne (son contenu varie au cours du programme) ; on peut la désigner par NOM$. « Martin » est, au contraire, une constante qui sera écrite entre guillemets dans les expressions. La condition d'égalité s'écrira, tout simplement, NOM$ = « MARTIN ».

De même, dans le bloc 500, la profession constitue une variable qu'on désignera par PROF$ tandis que les valeurs Employé et Enseignant sont des constantes. La relation s'écrit donc :

PROF$ = « EMPLOYE » OR PROF$ = « ENSEIGNANT ».



B. Coleman/Marka

Robot pratiquant des soudures sur un panneau électronique.

Les portions de chaînes. Une portion de chaîne représente un groupe de caractères extaits d'une chaîne. Tout à l'heure, en écrivant A$ = « Hector », B$ = « Berlioz », C$ = B$ + A$, nous avons formé la chaîne C$ à partir de A$ et B$. L'inverse, c'est-à-dire l'extraction des 7 premiers caractères de la chaîne C$, puis les suivants, nous fournira respectivement les deux parties de chaîne.

En Basic, des instructions permettent d'extraire d'une chaîne le nombre désiré de caractères, à partir d'une position donnée.

Les commandes

Tout langage possède deux types d'« ordres » : les commandes et les instructions. Les premières déclenchent des fonctions générales du système, telles l'exécution ou l'interruption d'un programme. Les instructions représentent le codage des opérations à effectuer, leur ensemble constituant le programme.
En Basic, les commandes sont plus nombreuses que dans les autres langages et presque toutes sont utilisables au cours même du programme. L'emploi des commandes est beaucoup plus délicat dans les autres langages.
On peut les répartir en quatre groupes, selon les opérations qu'elles effectuent :

1 / écriture et sauvegarde de programmes ;
2 / visualisation à l'écran et impression de listages ;
3 / exécution et interruption de programmes ;
4 / fusion.

Commandes d'écriture et de sauvegarde des programmes

Les instructions d'un programme Basic sont numérotées en ordre croissant, avec un écart arbitraire entre deux numéros. Ainsi, nous pourrions numéroter les instructions d'un programme de 5 en 5 à partir de 10 (10, 15, 20, 25, etc.) et celles d'un autre programme de 10 en 10 (10, 20, 30, etc.). En écrivant ces programmes l'un après l'autre, sans éteindre la machine, on obtiendra un curieux mélange d'instructions : les instructions paires du second programme remplaceront les instructions de même numéro appartenant au premier programme, tandis que les instructions impaires de celui-ci (15, 25, etc.) seront inchangées.
La commande NEW (nouveau) évite ces confusions entre le programme en cours et les précédents (encore présents en mémoire), en remettant à zéro toute la zone utilisateur de la mémoire. L'introduction de la commande NEW pendant l'édition d'un programme entraîne sa destruction.
Une fois la mémoire réinitialisée, les instructions écrites constitueront un nouveau programme.
On peut se dispenser d'écrire le numéro de chaque ligne en passant en numérotation automatique, avec la commande AUTO, dont la syntaxe est :

AUTO ligne de départ, pas

Le premier terme (« ligne de départ ») désigne le numéro de la première ligne que l'on va écrire, et, le second (« pas »), le pas de numérotation. Ainsi, AUTO 120, 5 déclenche la numérotation automatique des lignes de 5 en 5 à partir de 120. La production automatique des numéros de ligne peut aboutir parfois à un numéro déjà existant. Il s'affiche alors à l'écran, suivi d'un astérisque, et l'opérateur choisit d'interrompre la numérotation automatique pour conserver l'ancienne ligne, ou de continuer pour la remplacer par une nouvelle.
Lorsqu'on écrit de longs programmes, la commande NEW et la possibilité de numéroter à partir de n'importe quel nombre sont très utiles. Grâce à la commande AUTO, on peut interrompre et reprendre à tout moment l'écriture d'un programme, en faisant repartir la numérotation au numéro immédiatement supérieur au dernier utilisé.
Le programme doit être sauvegardé sur disquette ou sur bande magnétique car la plus brève coupure de courant, ou même, parfois, une simple variation de tension, peut entraîner sa perte (la mémoire de l'ordinateur est volatile). Pour une sauvegarde sur disque, la commande est :

SAVE « nom de l'unité de disque :
nom du programme »

Si le code SAVE est commun à la plupart des machines, les paramètres (nom de l'unité et du programme) dépendent du système d'exploitation. Ainsi, pour mémoriser le programme résident en mémoire dans un fichier de nom ESSAI, on utilisera, par exemple, en CP/M, la commande SAVE « A:ESSAI », et, en PCOS, SAVE « O:ESSAI ».
Le système d'exploitation CP/M détectera que le programme est écrit en Basic, et le mémorisera avec la mention BAS. Si l'on demande l'affichage du répertoire du disque

par la commande DIR (voir le chapitre sur les systèmes d'exploitation), on obtiendra le nom complet du programme : ESSAI.BAS.
En fait, la commande SAVE est encore plus complexe car on peut mémoriser les programmes sous deux formes : en code ASCII et en binaire condensé.
En code ASCII, lettres et chiffres constituant les lignes du programme sont mémorisés en tant que symboles ASCII, c'est-à-dire comme de simples caractères alphanumériques, tandis que, dans le second cas, tous les caractères sont codés en binaire.
Tous les caractères frappés au clavier sont codés en ASCII mais, pour être exécutable, une instruction en ASCII doit d'abord être traduite en binaire par l'interpréteur. Un programme peut être mémorisé indifféremment dans l'un ou l'autre code, et l'interpréteur décide au moment de l'exécution si la traduction est (ou non) nécessaire.
Une mémorisation sous forme ASCII devra être spécifiée par la lettre A, tandis qu'en binaire, il n'y a rien à indiquer. En CP/M, la commande SAVE peut donc prendre deux formes :

SAVE « A:ESSAI »,A
mémorisation en ASCII
SAVE « A:ESSAI »
mémorisation en binaire

Il s'avère parfois nécessaire de protéger des programmes. En indiquant P au lieu de A, le programme sera sauvegardé en binaire mais on ne pourra plus le lister ou le corriger. Il est impossible d'annuler cette protection, aussi est-il conseillé de conserver une copie non protégée des programmes. Le binaire, plus compact que le code ASCII, convient généralement mieux à la sauvegarde. Cependant, certaines opérations comme la compilation, nécessitent une version en code ASCII.
Voyons maintenant en détail l'enchaînement des commandes et des instructions qu'il faut communiquer à la machine, après sa mise sous tension, pour écrire et sauvegarder un programme :

> MBASIC Après la mise sous tension, le système d'exploitation est chargé et envoie le symbole > (prompt). MBASIC commande le chargement de l'interpréteur Basic.

NEW Demande d'effacement du contenu de la mémoire.

AUTO 50, 5 Demande de mise en numérotation automatique de 5 en 5 à partir de 50.

50 A = B + C
55 D = A * 6
60 K = C − B Les numéros de ligne s'affichent dans l'ordre croissant et l'utilisateur entre les instructions du programme (A = B + C, etc.).

CTRL + C Clôture de la phase d'écriture par la frappe simultanée des touches CTRL et C (voir les fonctions de la touche CTRL p. 322). Le MBASIC attend une nouvelle commande.

SAVE « A:PROG−1 »,A Le programme qui vient d'être écrit est enregistré en code ASCII, sous le nom PROG-1, sur la disquette A.

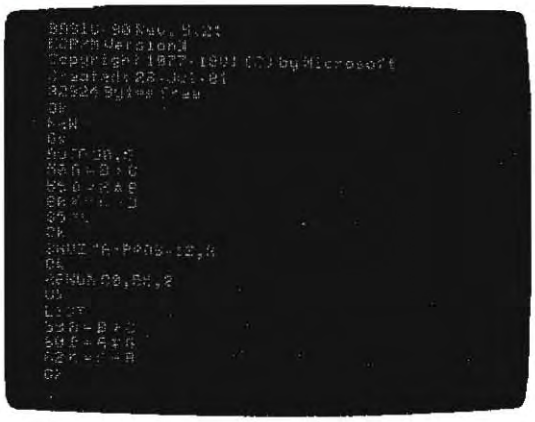
NEW La mémoire est réinitialisée. Il n'y reste aucune trace de ce qui vient d'être écrit.

Le programme PROG-1, ainsi stocké sur disquette, sera rappelé en mémoire pour être exécuté ou modifié. On utilise pour cela la commande LOAD « A:PROG-1 », identique sur tous les systèmes.
Le programme réside alors à nouveau en mémoire, et l'on peut le corriger ou l'exécuter. On dispose de trois commandes de correction : RENUM, EDIT et DELETE. RENUM permet de renuméroter un programme, en partie ou en totalité, avec la syntaxe suivante :

RENUM nouveau numéro ' ancien numéro ' pas

ECRITURE ET MEMORISATION D'UN PROGRAMME

Les différentes phases de la procédure s'affichent à l'écran.



```
BASIC-80 Rev. 5.21
[CP/M Version]
[Copyright 1977-1981 (C) by Microsoft]
[Created: 28-Jul-81]
32824 Bytes free
Ok
NEW
Ok
AUTO 50,5
50 A = B + C
55 D = A * B
60 K = C - B
65 ^C
Ok
SAVE "A:PROG-12,A
Ok
RENUM 80,55,2
Ok
LIST
80 A = B + C
82 D = A * B
84 K = C - B
Ok
_
```

Le système indique à l'utilisateur qu'il se trouve sous Basic et dispose de 32 824 octets.

La commande NEW (qui, ici, n'est pas nécessaire) réinitialise la mémoire. Le système signale que la commande a bien été exécutée (Ok).

L'opérateur peut désormais taper son programme.

On achève la phase d'écriture avec la commande CTRL + C (↑ C sur l'écran). Cette ligne 65, éliminée par l'interpréteur, ne fera pas partie du programme.

La commande SAVE provoque le stockage sur la disquette A du programme résidant en mémoire. Ici, on a frappé le caractère 2 au lieu des guillemets qui indiquent la fin d'une chaîne, à la suite du nom qu'on voulait donner au programme. Cette erreur est assez courante, car les deux symboles se trouvent sur la même touche et sont sélectionnés par l'utilisation de la touche des majuscules.
La machine n'a pas remarqué l'erreur (c'est une lacune du système), et le programme n'est donc pas enregistré sous le nom désiré.

Ces lignes illustrent l'emploi des commandes RENUM et LIST.

Par « nouveau numéro », on entend celui à partir duquel commencera la nouvelle numérotation.
L'ancien numéro désigne la première ligne à renuméroter et le pas, l'écart entre deux numéros consécutifs de la nouvelle numérotation.
La commande RENUM 80,55,2 renumérote de 2 en 2, à partir de la ligne 55, le programme que nous venons d'écrire.
Les nouveaux numéros de ces instructions seront donc 55, 80 et 82 (le numéro 50 reste inchangé puisque la renumérotation démarre à 55).
Pour corriger une ligne, on peut écrire son numéro, suivi de son nouveau contenu. L'ancienne instruction se trouve ainsi effacée, quelle que soit sa longueur.
Ainsi, en écrivant 55 D = A/6, nous remplaçons l'ancienne instruction 55 (D = A*6) par la nouvelle.
La commande EDIT fait passer le Basic en mode éditeur, et permet ainsi de corriger des lignes du programme sans les réécrire entièrement.

Avec EDIT, il aurait suffi de remplacer en ligne 55 le symbole de multiplication (*) par le symbole de division (/).
Superflu dans des cas aussi simples, ce mode de correction devient un outil très précieux pour la modification de lignes longues ou complexes.
En mode EDIT, nous disposons de cinq commandes, très semblables à celles de l'Editeur du système d'exploitation :

1 / déplacement du curseur le long d'une ligne ;
2 / insertion de caractères ;
3 / suppression de caractères ;
4 / recherche de caractères ou de groupes de caractères ;
5 / remplacement de caractères.

La connaissance approfondie des fonctions d'édition n'est utile que dans des cas particuliers ; à titre d'information, elles sont détaillées dans l'encadré ci-dessous.

Par ailleurs, on utilise en mode EDIT deux touches spéciales : RUBOUT (ou DEL) et ESCAPE.

Principales fonctions du mode EDIT (quelques exemples sont illustrés page 358).

1 / Déplacement du curseur
Le curseur (tiret ou rectangle lumineux qui indique la position où se trouvera le caractère entré) peut se déplacer le long de la ligne à « éditer » (c'est-à-dire à compléter ou à corriger).
On le déplace vers la droite, en appuyant sur la barre d'espacement, ou vers la gauche, avec la touche RUBOUT.

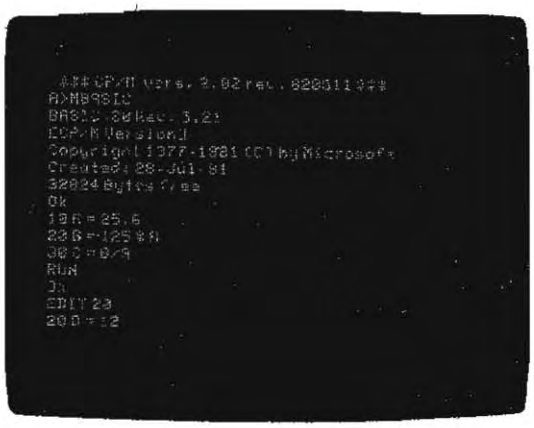
2 / Insertion de caractères dans une ligne
Pour insérer un ou plusieurs caractères dans une ligne, on positionne le curseur au caractère précédent, on frappe la touche I et on tape les caractères à insérer. La touche RUBOUT mettra fin à l'insertion.
Pour ajouter des caractères en fin de ligne, la touche X avance le curseur à la bonne position. L'insertion se déroule automatiquement.
On terminera l'insertion en appuyant sur la touche ESCAPE.

3 / Suppression de caractères
On l'effectue avec la touche D (delete) après avoir positionné le curseur sous le caractère à supprimer ; on efface ainsi les caractères un à un. Cette commande peut se compléter par un facteur de répétition : en tapant 4D, on supprimera quatre caractères à partir du curseur.
Pour effacer tous les caractères à droite du curseur, on utilisera la lettre H.

4 / Recherche de caractères
Elle est déclenchée par la touche S (search). Si l'on tape, par exemple, Sx, la machine cherche la lettre x dans la ligne éditée, à partir de la position du curseur.
L'utilisation de K au lieu de S permet en plus de supprimer le caractère trouvé.

5 / Remplacement de caractères
Si l'on appuie sur la touche C (conversion), le caractère sous lequel se trouve le curseur sera remplacé par celui qu'on entre aussitôt après. Ainsi, Cr remplace l'ancien caractère par la lettre « r ».

UTILISATION DE LA COMMANDE EDIT



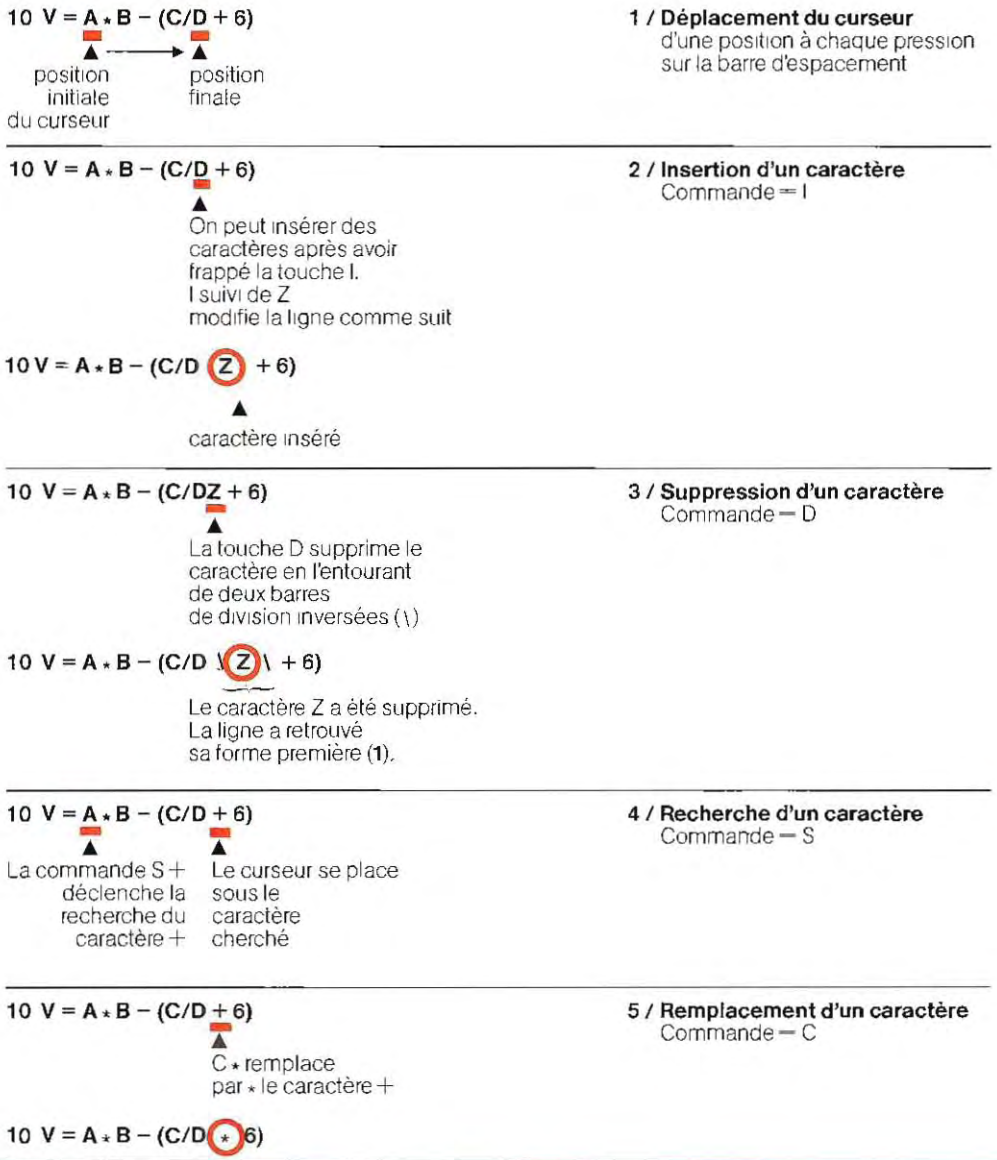
■ Ces lignes constituent le programme entré au clavier.

■ La commande RUN déclenche l'exécution du programme qui réside en mémoire. Le message Ok indique que l'exécution est terminée.

■ La commande EDIT 20 affiche à l'écran la ligne 20 du programme.

■ L'opérateur positionne alors le curseur sous le caractère voulu et apporte, en mode éditeur, les corrections désirées. Lorsqu'il a terminé, il appuie sur la touche de retour chariot ; la nouvelle ligne 20 remplace désormais l'ancienne dans le programme.

EXEMPLES D'OPERATIONS D'EDITION



En utilisant la commande EDIT, on doit préciser le numéro de la ligne à corriger : on écrit, par exemple, EDIT 55 pour modifier la ligne 55.

Tous les claviers possèdent la touche ESCAPE (généralement indiquée par l'abréviation ESC).

En revanche, la touche RUBOUT est parfois remplacée par la touche DELETE (DEL) ou encore par la touche curseur comportant le symbole ↰.

La dernière commande de correction des programmes, RUBOUT ou DELETE (qui, en anglais, signifie biffer, rayer), permet de supprimer des lignes, de tel à tel numéro. Ainsi, DELETE 10-28 supprime les lignes 10 à 28 inclus.

Sur certains systèmes, les deux numéros sont séparés par un tiret, alors qu'on utilise toujours la virgule pour les autres commandes (exemple : RENUM 20, 28, 5).

Cette particularité diminue le risque d'utiliser la commande DELETE par erreur.

Wafer, chip and Co.

Le microprocesseur est sans doute le produit le plus remarquable de la technologie du silicium. Il concentre sur une « puce » (ou chip) de silicium une capacité de calcul qu'on aurait obtenue, dans les années cinquante, avec 40 tonnes de matériel. En outre, il se fabrique en grande série, à un coût unitaire de quelques dollars seulement.
Ajoutez au microprocesseur quelques microcircuits de mémorisation, une alimentation, quelques accessoires (clavier, écran, entre autres), et vous obtiendrez un calculateur électronique.

* * *

Mais qu'est-ce qu'une puce ? Que peut-elle faire et de quoi est-elle faite ?
C'est une mince plaquette de silicium à la surface de laquelle s'étend le microscopique dessin du microcircuit, en une succession de couches de silicium pur ou impur (dopé), d'oxyde de silicium et d'aluminium. L'élément fondamental du chip est le transistor, plus précisément le transistor à effet de champ (FET : Field Effect Transistor).
Un tel transistor possède trois connections électriques appelées ***source, drain*** *et* ***porte.***
Le courant peut passer de la source au drain si une tension adéquate est appliquée à l'électrode porte. Le transistor à effet de champ constitue donc un interrupteur électronique commandé par une tension à la porte.
En associant un certain nombre de ces transistors, on réalise les opérateurs logiques simples qui sont à la base du calcul automatique.
Un transistor à effet de champ est formé de deux types de silicium différents, les matériaux « n » et « p ».
Le silicium « n » est dopé par des impuretés (comme le phosphore), qui produisent, au sein du métal, un excès d'électrons. Pour le type « p », d'autres impuretés (comme le bore) créent au contraire un déficit d'électrons. Dans les deux cas, le déséquilibre électrique assure la conductivité du silicium.
Dans le transistor, deux couches de type « p » forment la source et le drain, tandis qu'une électrode appliquée sur la couche intermédiaire de type n, constitue la porte.

En pratique, un microcircuit qui peut comprendre des milliers de transistors à effet de champ, est réalisé sur une plaquette (appelée wafer) d'un type de matériau déterminé (par exemple p).
Par « dopage », on crée des îlots de type opposé, en masquant soigneusement les zones qui ne doivent pas être dopées.

* * *

La production d'un nouveau microcircuit passe par plusieurs étapes : spécification du produit par une équipe de conception, dessin des circuits conformément à leur cahier des charges, test d'efficacité au moyen de simulateurs.
Les circuits satisfaisants sont ensuite tracés par un ordinateur graphique, et on attribue aux divers éléments du circuit des couches précises du bloc de silicium. L'ordinateur fournit un plan de contrôle à grande échelle et une bande magnétique codifiée, permettant la réalisation des masques.
Ces derniers servent au tracé du circuit sur les différentes couches du bloc. Autrefois réalisés par procédé photographique et réductions successives d'un plan à grande échelle, les masques sont aujourd'hui dessinés directement par un rayon électronique, commandé par la bande magnétique.

* * *

Comme on l'a mentionné, les microcircuits sont gravés sur des wafers de silicium. Pour préparer les wafers, on ajoute au silicium pur fondu des impuretés du dopage. Une lente extraction transforme le matériau fondu en un gros cristal cylindrique, que l'on sciera de façon à obtenir des éléments circulaires (wafers) d'un diamètre de 100 ou 120 mm. Après un polissage de haute précision, la surface du wafer est prête à recevoir les couches du circuit. Chaque wafer peut contenir jusqu'à 500 microcircuits.

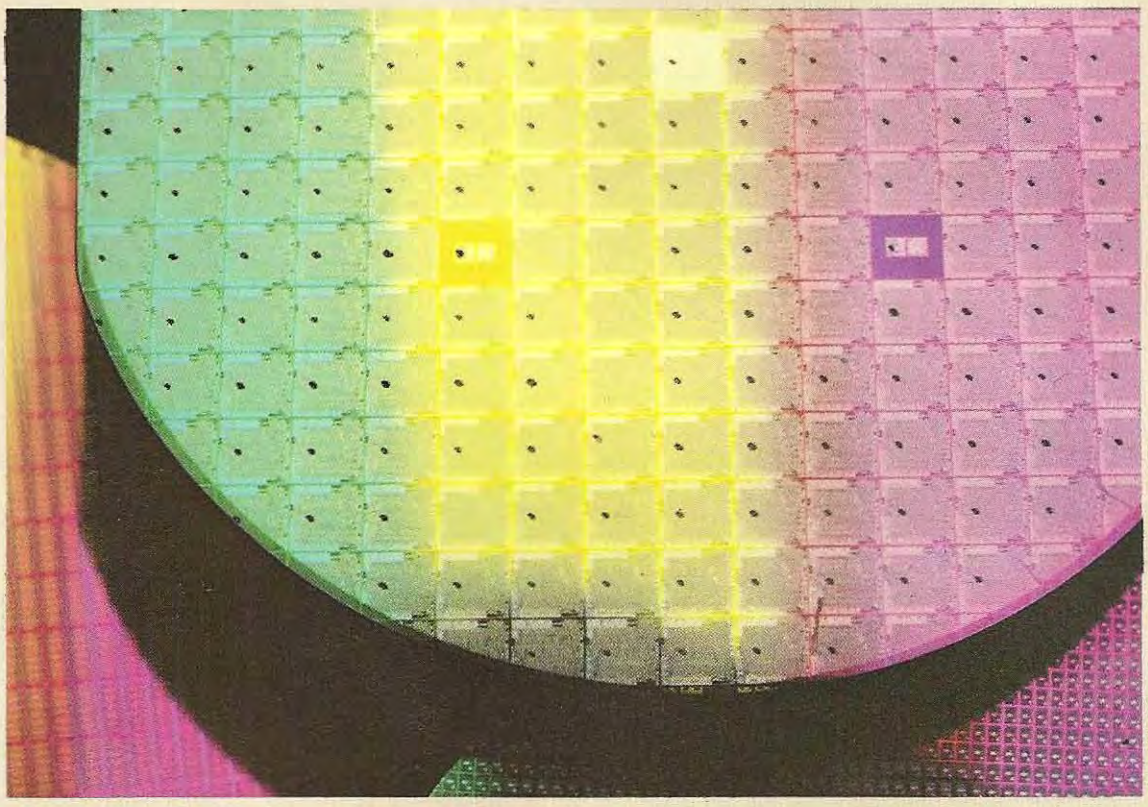
* * *

Une couche du microcircuit s'obtient classiquement de la façon suivante. On crée tout d'abord une couche de bioxyde de silicium (envoi de vapeur d'eau ou de gaz riche en oxygène) à la surface du wafer placé dans un four. En réglant la température et la quantité de

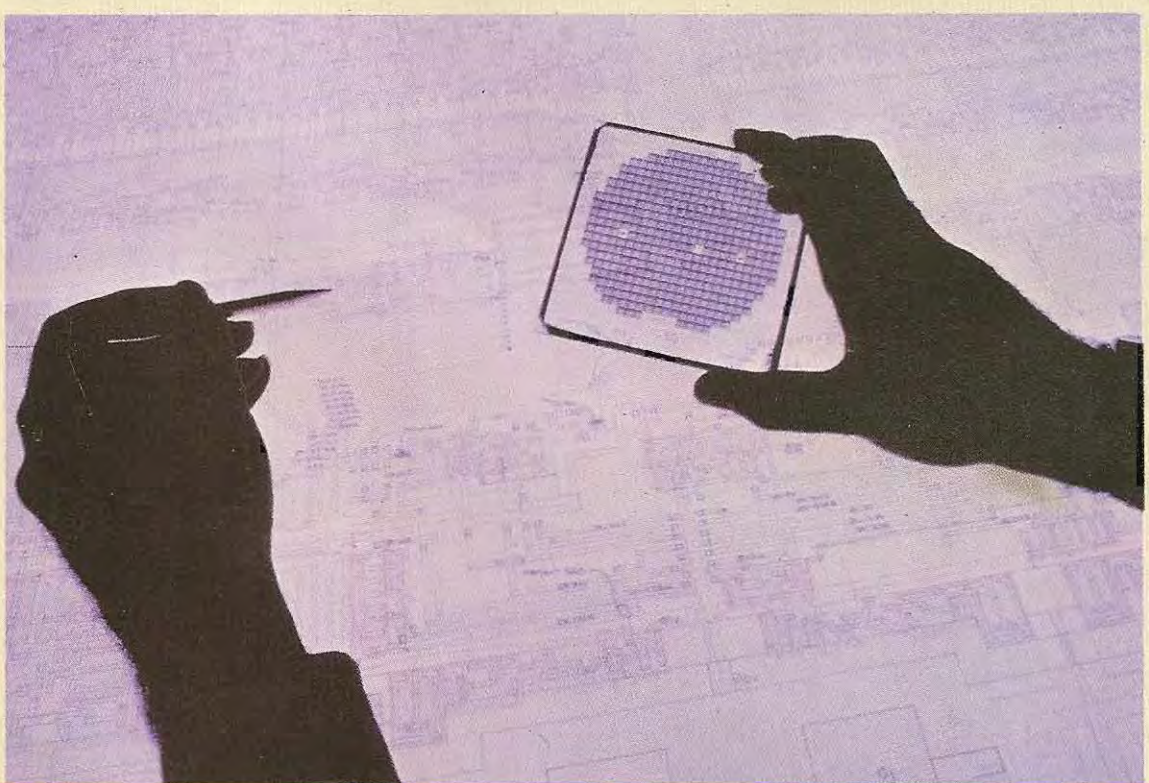
gaz, on détermine exactement l'épaisseur de la couche oxydée. On la recouvre ensuite d'un produit chimique photo-durcisseur, solidifié par exposition aux rayons ultraviolets.
Cette lumière est envoyée sur le wafer au travers d'un masque approprié où se répète des centaines de fois un motif de base.
On procède par contact ou par projection : dans le premier cas, le masque est directement posé sur le wafer, dans le second cas, un système optique projette le dessin. Quel que soit le système employé, le matériau durcit là où la lumière parvient.
Ailleurs, il sera dissout et retiré au moyen d'un révélateur chimique.
L'opération suivante est celle de la gravure par l'acide.

La gravure de la plaquette est réalisée généralement par attaque à l'acide fluorhydrique. Dans les zones exposées, l'oxyde est attaqué à l'acide fluorhydrique, tandis qu'il demeure intact là où il est protégé par une mince couche photo-résistante. Puis les résidus de la substance photo-sensible sont éliminés à l'aide d'un révélateur.
Des zones d'oxyde masquent ainsi les parties de la plaquette qu'on veut préserver du dopage.

Microcircuits imprimés sur une plaquette de silicium avant leur découpage. En bas : le circuit dessiné sur le plan est reproduit sur chaque puce de la plaquette.



B. Coleman/Marka



B. Coleman/Marka

Le wafer sera chauffé dans un four, en atmosphère borée ou phosphorée selon que l'on veut obtenir des zones actives p ou n. Il y restera le temps nécessaire à la diffusion de la substance dopante (impuretés) dans le silicium.
Une couche du microcircuit vient donc d'être réalisée. Cette succession d'opérations se répètera plusieurs fois, avec des variantes, pour créer les autres couches du circuit intégré (11 en moyenne, qui nécessitent à peu près 60 opérations).
On dépose ensuite une couche métallique (d'aluminium en général), gravée sur masquage, qui établit un réseau de connexions entre les différentes zones actives. Cette couche possède également des contacts reliant le circuit à l'extérieur.
Une dernière couche d'oxyde est enfin appliquée pour protéger le circuit.

* * *

La moindre pollution, à n'importe quel stade de la fabrication des puces, peut compromettre leur fonctionnement: la production s'effectue donc en atmosphère soigneusement filtrée et dans des limites de température parfaitement contrôlées. Les techniciens doivent même endosser des vêtements de protection stériles.
Les puces sont alors testées l'une après l'autre à l'aide de pointes reliées à un ordinateur, puis elles sont ensuite séparées les unes des autres.
Les puces défectueuses sont marquées à l'encre, puis éliminées lors du découpage. Ce dernier, qui suivait autrefois les lignes de coupe du cristal, utilise aujourd'hui les techniques modernes du diamant et du laser. Les chips non défectueux sont collés sur un socle et leurs pattes de contact sont soudées par de fins fils d'or à celles du socle. Ils sont ensuite encapsulés dans un boîtier de plastique ou de céramique avant d'être soumis à un nouveau contrôle.
Dans le cas d'une nouvelle fabrication, le pourcentage de puces sans défaut peut ne pas dépasser 5 %; le rendement d'une bonne production n'est d'ailleurs pas toujours supérieur à 25 %.
D'importants progrès sont accomplis tous les jours dans le domaine de la fabrication des microprocesseurs: on conçoit de nouveaux circuits en redessinant les zones défectueuses sur les précédents, on modifie les procédés de fabrication et on réduit la taille des puces.
Ce dernier point permet d'augmenter la production de chips au cours d'un même cycle de fabrication et d'en améliorer le rendement. De plus, on accroît ainsi la puissance, la compacité et la vitesse de fonctionnement des circuits en réduisant la longueur des connexions, ce qui est fondamental pour des ordinateurs qui effectuent des milliers de calculs à la seconde.

* * *

Un bref regard vers l'évolution passée s'impose. A l'apparition des circuits intégrés, dans les années soixante, Gordon Moore, le fondateur d'Intel, prédit que la complexité (nombre de composants intégrables dans un circuit) doublerait chaque année.
Bien peu de gens le crurent, et pourtant, la « loi de Moore » s'est, jusqu'ici, assez bien vérifiée. En effet, la densité d'intégration, qui était de 32 composants en 1965, atteignait 1 000 en 1970, 1 million en 1980, et 40 millions en 1984. Il est probable que les circuits intégrés vont continuer à obéir à cette loi. Cela sera d'autant plus facile que le schéma des éléments de mémoire sera plus régulier.
Les techniques de production devront être adaptées à l'augmentation constante du nombre d'éléments que l'on rassemble dans un même circuit.
Par exemple, lors des opérations de masquage, les ultraviolets, de trop grande longueur d'onde pour les circuits les plus compliqués (à haute intégration), devront bientôt être remplacés par les rayons X, dont la longueur d'onde beaucoup plus courte permettra de résoudre ce problème.
Dans un avenir plus lointain, les chercheurs envisagent de remplacer le support de silicium, aujourd'hui principal support des circuits intégrés.
*Pour cela, on utiliserait l'**arséniure de gallium**, plus délicat à manipuler, mais qui rend possible la réalisation de circuits plus rapides en consommant moins d'énergie.*
Cette baisse de la consommation, se traduisant par une diminution de la chaleur dissipée, permettra de réduire encore la taille des composants et d'en augmenter la complexité en faisant tenir plus de composants sur chaque puce.

Commandes de la bande magnétique. La bande magnétique peut remplacer le disque comme support de mémoire de masse. Elle possède ses propres commandes de sauvegarde (SAVE) et de chargement de programmes en binaire (LOAD).
Voici leur syntaxe :

CSAVE "NOM"
CLOAD "NOM"

NOM désigne le programme qu'on veut sauver ou charger. Toutefois, pour une bande magnétique, seul le premier caractère du nom (ici, N) est pris en compte : CSAVE "NOM" et CSAVE "N" sont donc équivalents. Dans les deux cas, le programme est stocké sous le nom N. Il faudra donc être très attentif à ne pas utiliser des noms commençant par la même initiale pour deux programmes différents.
Les instructions CSAVE et CLOAD, servent également, sous une forme légèrement différente, pour enregistrer des données en chaînes de caractères.

Affichage et listage des programmes

Le Basic standard possède deux périphériques de sortie : l'écran et l'imprimante. On peut demander l'affichage à l'écran ou l'impression sur papier, aussi bien de données que de programmes.
Pour les programmes, on utilise les commandes suivantes :

LIST ligne de début, ligne de fin
(pour l'écran)
LLIST ligne de début, ligne de fin
(pour l'imprimante)

« Ligne de début » et « ligne de fin » correspondent aux numéros des instructions qui délimitent la partie du programme à lister : LIST 20, 100 provoque l'affichage à l'écran du programme résident de la ligne 20 à la ligne 100. La commande équivalente pour l'imprimante est LLIST 20, 100.
Il existe deux commandes NULL et WIDTH peu usitées, que nous ne mentionnerons que pour mémoire.
NULL sert à indiquer, pour les machines équipées d'un perforateur de bande, le nombre de blancs que l'on veut laisser entre deux lignes. Si l'on précise, par exemple, NULL 4, la machine laissera quatre blancs après chaque ligne.
Quant à la commande WIDTH, elle détermine la longueur maximale des lignes à écrire. La commande WIDTH 20 pour l'écran (ou WIDTH LPRINT 20 pour l'imprimante), limite à 20 le nombre de caractère par ligne, en créant une « fenêtre » de 20 emplacements excluant le caractère supplémentaire éventuel.

Exécution, interruption et mise au point des programmes

On lance l'exécution du programme résidant en mémoire par la commande RUN.
Tout programme contient une instruction de fin (END) et l'exécution s'arrête donc, en principe, à cette instruction. Cependant, il arrive, lors des premiers essais surtout, que des erreurs de programmation conduisent à éviter systématiquement l'instruction END.
La machine ne trouvant pas cette instruction de fin, le programme continue alors à tourner indéfiniment si on ne l'interrompt pas en appuyant simultanément sur les touches CNT (ou CTRL) et C (control + C).
La commande CONT (continue), qui relance l'exécution, sert également à tester pas à pas le déroulement d'un programme.
Une instruction STOP, insérée aux endroits « stratégiques » du programme, en interrompra l'exécution.
L'opérateur pourra alors demander interactivement l'affichage des résultats intermédiaires, et les analyser.
La commande CONT fera reprendre l'exécution là où elle aura été interrompue. De cette manière, on peut examiner les étapes de chaque opération et s'assurer de leur validité.
On ne peut pas employer CONT si on a profité de l'arrêt du programme pour y apporter des corrections.
Les deux dernières commandes de mise au point, utilisables interactivement (commandes) ou dans le programme (instructions), sont TRON et TROFF, qui régissent le mode « trace ».
TRON (trace on) affiche au fur et à mesure les numéros des instructions dans l'ordre de leur exécution.
On peut ainsi connaître avec précision le chemin suivi par la machine, sans avoir à consulter l'organigramme ou le listing du programme.

Un programme pouvant contenir un grand nombre de décisions (tests) en fonction desquelles un parcours est choisi plutôt qu'un autre, la logique devient parfois si compliquée qu'il est difficile de discerner d'où viennent les erreurs. Il est alors plus simple d'utiliser la commande TRON que d'essayer de « dérouler » le programme à la main.
La commande TROFF (trace off) annule la commande TRON (trace on).

Concaténation

Lorsqu'on fractionne les programmes en sous-programmes, il est souvent possible de réutiliser, dans une nouvelle application, des suites d'instructions écrites pour une autre.
Ces « routines » d'usage général seront enregistrées sur des fichiers séparés, comme s'il s'agissait de programmes indépendants. Leur réemploi est ainsi facilité. Pour s'en servir, il suffit de les prélever sur le disque et de les fusionner au programme résident.
On utilise pour cela la commande MERGE suivie du nom du fichier. Cette instruction ne concerne que les programmes (fichiers en Basic) stockés en format ASCII ; utilisée sur un fichier binaire ou sur des données, la machine émet un message d'erreur et la commande n'est pas exécutée.
La commande MERGE « ESSAI » charge la routine ESSAI enregistrée sur la disquette et l'unit au programme résident, réalisant ainsi leur concaténation. Notons que le programme à fusionner doit avoir été sauvegardé en code ASCII.
Si ces deux programmes possèdent des numéros de ligne en commun, ce sont les instructions du sous-programme qui se trouvait sur le disque qui, lors de la fusion, remplacent celles du programme résident.
Prenons un exemple :

Programme en mémoire
```
10 A = B + C
20 D = A * 2 + SQR (B)
30 E = A + D
40 TOTAL = E * 2
50 T1 = E - TOTAL
```

Programme sauvegardé sur disque sous le nom PROG-2
```
30 F = A + B + C
50 SOMME = TOTAL + F
60 END
```

L'instruction MERGE « PROG-2 » charge le programme PROG-2 et le fusionne avec le programme résident. Celui-ci possède désormais l'instruction 60 et ses lignes 30 et 50 sont remplacées par celles de PROG-2.
Un programme « structuré » se compose d'une partie principale et de routines (sous-programmes).
Le programme principal esquisse la structure générale du programme, tandis que les sous-programmes traitent en détail l'exécution des différentes opérations.
La commande MERGE permet de réaliser cette structuration.
Lors de l'écriture, on prépare et on mémorise séparément chaque module (programme principal et sous-programmes) au lieu de tout réunir, dès le départ, dans un même fichier.
Il suffit ensuite de charger en mémoire le programme principal et de lui adjoindre, par fusions successives, tous les sous-programmes nécessaires.
On se réserve ainsi la possibilité de réutiliser les routines dans d'autres applications, en les fusionnant à d'autres programmes principaux.
Enfin, la commande CLEAR effectue les opérations suivantes :

- clôture des fichiers (un programme ne peut accéder à un fichier tant qu'il n'a pas été réouvert) ;
- remise à zéro des variables numériques et de chaînes ;
- réservation de l'espace mémoire disponible pour l'utilisateur (en particulier réinitialisation de la zone réservée aux variables).

Généralement, cette dernière commande s'effectue en début d'exécution d'un programme. Elle est prise en charge par le système, dans la mesure où réservations d'espace et initialisations des variables se produisent automatiquement à l'exécution d'un programme.

Les instructions

Les instructions représentent la codification, en langage de programmation, des opérations que l'ordinateur doit effectuer.
Dans ce chapitre seront exposées toutes les instructions qui traitent des variables et des constantes.

Test 10

1 / On a mémorisé un programme sur disque de trois façons différentes, en utilisant les commandes suivantes :

a) SAVE "A:ESSAI"
b) SAVE "A:TEST", A
c) SAVE "B:ESSAI2", P

Quelle est la version en ASCII ?
Sur quel disque le programme nommé ESSAI2 est-il mémorisé ?
Lesquelles de ces trois versions peut-on corriger ?

---

2 / Le programme suivant :

```
10 A = 6
20 B = 8
30 C = 10
40 D = C * (A + B)
50 E = 2 * D/A
```

a été renuméroté par la commande REN 20, 10, 2, puis fusionné à la routine par la commande MERGE :

```
20 B = 2
30 F = D + E
```

Donner les valeurs des variables D et E dans les cas suivants :
a) exécution du premier programme seul et avant la renumérotation ;
b) exécution du programme obtenu après renumérotation et fusion.

---

3 / Ecrire le programme obtenu par la séquence suivante :

```
NEW
10 A = 3
20 B = 2
30 C = 3 * (A + B)
REN 10,10,2
AUTO 20,5
20 F = 2 * (C + A)
25 PRINT F
```

---

4 / Comment remplacer la ligne 20 F = 2 * (C + A) par F = 4 * (C + A) ?

---

5 / A l'exécution du programme suivant que lira-t-on à l'écran ?

```
10 A = 6
20 TRON
30 B = A + 2
40 C = 2 * B
50 TROFF
60 D = B + C
```

Quelle commande doit-on utiliser pour sauvegarder ce programme sur le disque A, en format ASCII et sous le nom ESSAI ?

Les solutions de ce test se trouvent pages 374 et 375.

Tableau récapitulatif des commandes

Commande	Effet	Exemple
NEW	Remet à zéro le contenu de la mémoire.	NEW
AUTO n, m	Déclenche la numérotation automatique à partir de la ligne n, et avec un pas m.	AUTO 10, 5
CLEAR	Réinitialise les variables numériques et les chaînes de caractères.	CLEAR
SAVE "X:YYY", Z	Sauvegarde sur le disque X (A/B ou 0/1) le programme de nom YYY, avec l'option Z (en binaire par défaut ; en ASCII si Z = A ; protégé si Z = P).	SAVE "A:PROG-1", A SAVE "A:PROG-2" SAVE "0:NOM", P
RENUM n1, n2, m	Renumérote le programme de m en m, à partir de la ligne n2 qui prend le numéro n1.	RENUM 10,5,5 RENUM 300,2,12
EDIT n	Edite la ligne n.	EDIT 10
LIST n, m	Affiche à l'écran les lignes n à m.	LIST 5, 50
LLIST n, m	Effectue la même opération sur l'imprimante.	LLIST 3, 25
RUN	Lance l'exécution du programme résident.	RUN
TRON	Fait passer en mode « trace » l'affichage des numéros des instructions exécutées.	TRON
TROFF	Annule la commande TRON.	TROFF
MERGE "YYY"	Charge la routine YYY stockée sur disque, et la concatène au programme résident.	MERGE "PROG-1"

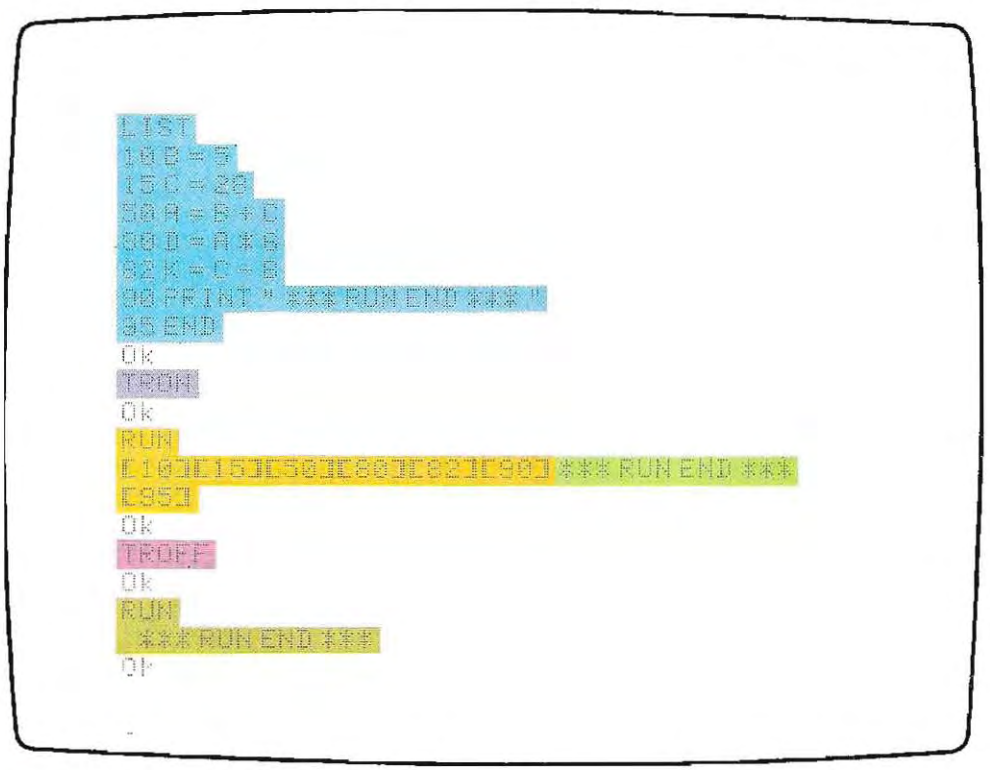
Les instructions qui concernent les mémoires de masse (bandes ou disques) feront l'objet d'une étude particulière.
En exposant la syntaxe des instructions, nous avons souvent inséré des blancs entre les mots clés, les noms de variables et les opérateurs par souci de clarté, afin de mieux distinguer les différentes parties d'une même instruction. Toutefois, il faut savoir dès maintenant que ces espaces n'ont aucune signification particulière pour l'interpréteur (ou le compilateur) Basic, qui reconnaît les éléments d'une instruction où qu'ils se trouvent dans une ligne, pourvu que l'orthographe soit correcte et l'ordre respecté.

Début et fin des programmes

De même que les organigrammes commencent par un bloc de début et se terminent par un bloc de fin, un programme comporte nécessairement une instruction START et une instruction END.
Cependant, dans certains langages, et notamment le Basic, il n'existe pas d'instruction de début : l'exécution du programme démarre automatiquement à la ligne portant le plus petit numéro.
Cependant, aucun automatisme similaire ne peut déterminer la fin du programme car les instructions qui portent les plus grands numéros ne sont pas toujours celles qui doivent être exécutées les dernières.
Comme nous l'avons vu, un programme se décompose en plusieurs modules : le programme principal (portant le nom de Main, en anglais) et les sous-programmes (appelés aussi routines, de l'anglais Subroutines). Le programme principal appelle les sous-programmes au fur et à mesure de ses besoins. A la fin de chaque sous-programme, l'exécution

EMPLOI DES COMMANDES RUN, TRON ET TROFF

```
LIST
10 B = 5
15 C = 20
50 A = B + C
80 D = A * B
82 K = C - B
90 PRINT " *** RUN END *** "
95 END
Ok
TRON
Ok
RUN
[10][15][50][80][82][90] *** RUN END ***
[95]
Ok
TROFF
Ok
RUN
 *** RUN END ***
Ok
```



La première commande (LIST) demande le listage du programme qui se trouve en mémoire. Ce programme s'affiche à l'écran, suivi du message Ok qui indique que l'exécution de la commande est terminée.

Avec TRON, l'opérateur passe en mode trace, demandant la visualisation des numéros des instructions exécutées.

La commande RUN lance l'exécution du programme. Les numéros de ligne apparaissent successivement à l'écran (entre crochets).

Cet affichage de caractères résulte de l'exécution de l'instruction de la ligne 90 (PRINT).

L'exécution terminée, l'opérateur annule le mode trace par la commande TROFF.

Lors de la nouvelle exécution, la seule sortie sera celle de la ligne 90.

du programme principal reprend juste après l'instruction d'appel. Certaines lignes sont donc « sautées » pour être reprises ensuite. L'exécution d'un programme n'est donc pas obligatoirement séquentielle.
Le schéma de la page 369 détaille un programme structuré en deux sous-programmes (A et B). Arbitrairement, la routine A est la première appelée, mais elle peut, en effet, s'insérer à n'importe quel endroit du programme (par exemple, de 3700 à 4580). De même, B, bien qu'appelée plus tard, pourrait très bien la précéder (elle occupe ici les lignes 1350 à 1920).
Le choix de l'ordre d'appel des routines est déterminé par les lignes 50 et 95 du programme principal. A la fin du premier sous-programme (A), se trouve l'instruction RETURN (ligne 4580) qui rend la main au programme principal à partir de la ligne 60.
L'ordinateur effectue les instructions 60, 65 et 95. Cette dernière appelle la routine B. L'exécution se poursuit donc de la ligne 1350 à la fin de la routine B, c'est-à-dire à la ligne 1920 où se trouve l'instruction de retour au programme principal (ligne 100).
A partir de cette ligne, l'exécution se poursuit dans l'ordre des numéros (instructions 100, 110, etc.). Si la machine ne rencontre pas auparavant l'instruction de fin (END), elle exécute à nouveau le sous-programme B (bien qu'il n'ait pas été rappelé). Elle rencontre alors, à la ligne 1920, l'instruction RETURN, exécute à nouveau la partie du programme comprise entre les instructions 100 et 1920 et ainsi de suite. Puis elle signale une erreur.
Sur le schéma de la page 369, on a placé l'instruction END au terme logique de l'exécution, c'est-à-dire après les deux sous-programmes. Elle devrait, en fait, se trouver « physiquement », (selon son numéro de ligne) dans le programme principal, avant le sous-programme A. L'organigramme c) détaille le schéma a) et résume le mécanisme décomposé en b), en représentant l'ordre effectif des instructions.
L'exécution du programme démarre automatiquement à l'instruction de plus petit numéro. C'est donc ici que le programmeur devra éventuellement spécifier les options qu'il choisit, comme par exemple, la base de numérotation des tableaux, avec l'instruction OPTION BASE 1. L'ordinateur numérote habituellement les tableaux à partir de 0, ce qui peut s'avérer gênant pour certaines applications.
Ainsi, pour un programme en Basic, les éléments d'un tableau de cinq valeurs sont numérotées de 0 à 4. Si le programmeur le désire, l'instruction OPTION BASE 1 insérée en tête de programme fera débuter la numérotation à 1. Dès lecture de cette instruction, la machine cessera d'utiliser l'indice 0.
Il ne s'agit, en fait, que d'une question de présentation : la machine fonctionne toujours de la même façon et décrémentera de 1 chaque indice, pour retrouver son propre système de numérotation.
Alors que l'instruction OPTION BASE 1 peut commencer un programme, l'instruction END le termine obligatoirement, constituant ainsi la dernière ligne au sens logique (et non selon l'ordre de numérotation).
Il est toujours utile, lors de la rédaction d'un programme, d'écrire des commentaires expliquant les opérations effectuées et décrivant les méthodes employées.
Un programme « nu » s'avère, en effet, fort difficile à ré-examiner et à comprendre après quelque temps. On y insère donc un certain nombre de lignes qui ne constituent pas des instructions, mais de simples commentaires. Cette différence est spécifiée à l'ordinateur par REM (abréviation de « remark »), placé en tête de ligne, ou par un symbole particulier (apostrophe, point d'exclamation, étoile, « c » minuscule, etc., selon les systèmes et les langages).
Voici un exemple :

```
10 OPTION BASE 1
20 REM Cette ligne est un commentaire
30 A = B + C 'Somme
(30 A = B + C !Somme)
40 END
```

A la ligne 20, le commentaire est indiqué par REM et par les symboles ' ou ! à la ligne 30. Tout ce qui se trouve à droite du symbole est considéré comme commentaire et n'est donc pas traité. Cependant, ces commentaires occupent de la place en mémoire.

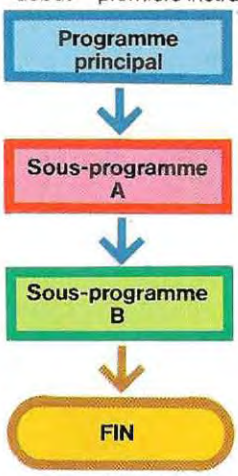
Déclaration du type de variables

Le programmeur dispose de quatre types de variables ou de constantes : entiers, réels en

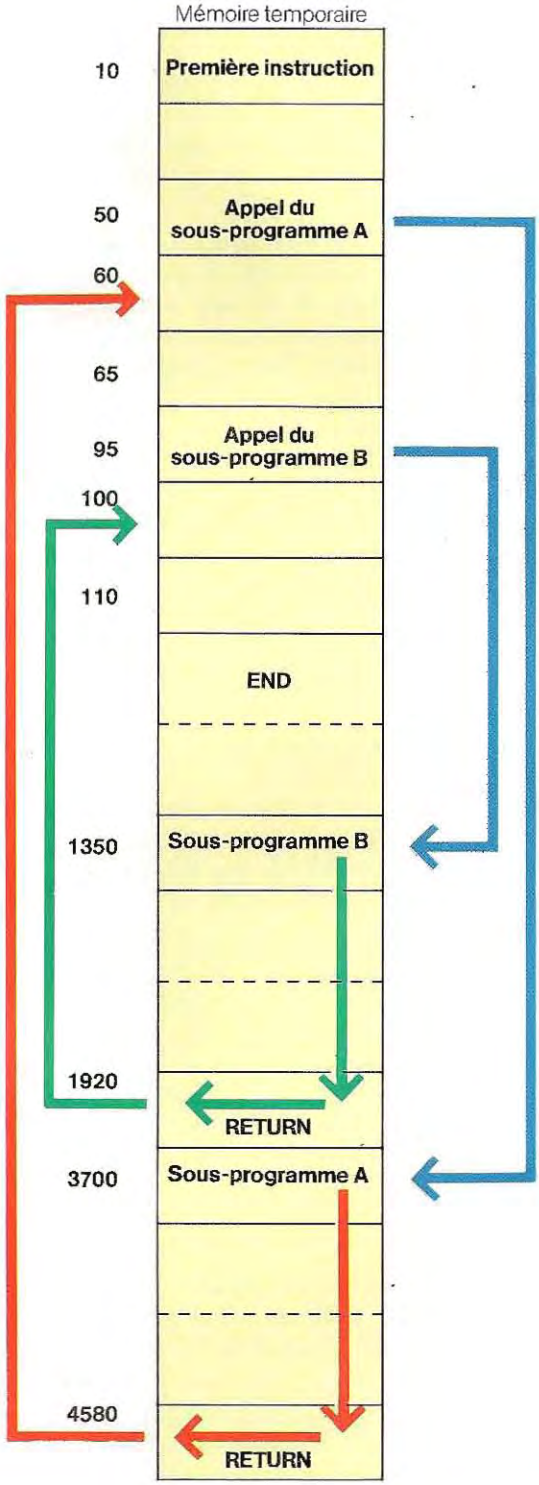
STRUCTURATION D'UN PROGRAMME ET DEROULEMENT DE SON EXECUTION

a) Structure d'un programme contenant deux sous-programmes

début = première instruction

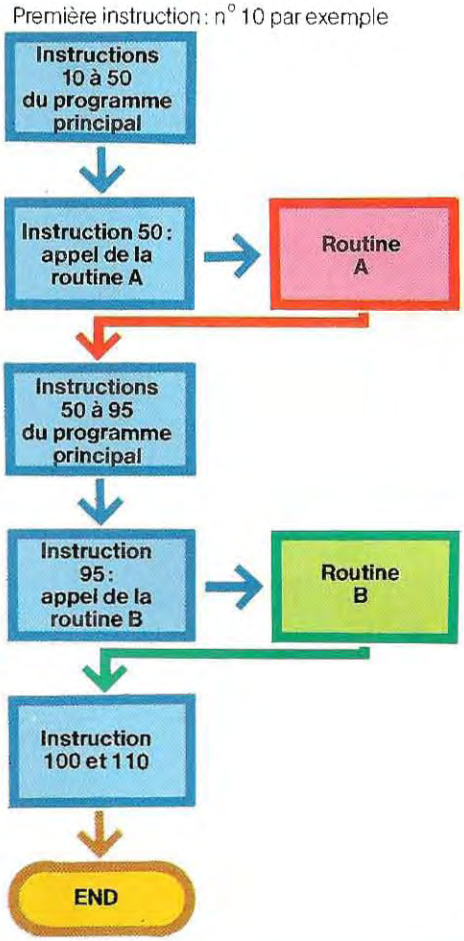


b) Schéma du parcours effectué lors de l'exécution



c) Organigramme correspondant

Première instruction : n° 10 par exemple



« simple précision » et en « double précision », chaînes de caractères. Pour déclarer un type, le nom (ou la valeur) sera suivi du symbole correspondant (% = entier, ! = simple précision, # = double précision, $ = chaîne).
Les noms de variables seront toujours accompagnés de l'un de ces symboles (sauf le symbole ! qui peut être omis, car, en l'absence d'indication, la machine considère « par défaut » qu'il s'agit d'une variable en simple précision).
Il serait très fastidieux de répéter continuellement ces symboles lorsqu'on écrit un programme. De plus, la lecture d'un listage encombré de symboles finit par être difficile. Pour éviter cela, on peut remplacer la « déclaration explicite » (spécification du symbole) par une « déclaration implicite ».
Une instruction DEF (à ne pas confondre avec DEF FN vu précédemment), précisera une fois pour toutes le type d'une variable ou d'un groupe de variables.
En voici la formulation :

DEFINT pour les entiers
DEFSNG pour les réels en simple précision
DEFDBL pour les réels en double précision
DEFSTR pour les chaînes de caractères

On indiquera, dans l'instruction, les lettres par lesquelles commenceront lesdites variables. Par exemple, DEFINT I-N signifie que toute variable du programme dont le nom commence par une lettre entre I et N est entière. La définition peut contenir plusieurs groupes de lettres séparés par des virgules. Ainsi :

DEFINT I-N, X-Z	Les noms ayant pour initiale une lettre comprise entre I et N ou X et Z désignent des variables entières.
DEFDBL A-D, P-T	Les noms des variables en double précision commencent par une lettre comprise entre A et D ou P et T.

Cette déclaration implicite n'existe pas sur tous les systèmes. En outre, il arrive parfois qu'on ne puisse définir un intervalle de lettres mais qu'on doive déclarer le nom des variables en entier et donc définir séparément chaque variable.
La déclaration implicite est surtout utile pour les entiers et les réels en double précision.
La déclaration facultative des réels en simple précision est cependant conseillée pour plus de clarté.
L'instruction DEFINT, citée en exemple, est très utile pour une programmation rigoureuse. Dans une boucle (voir page 174), on utilise, en effet, un indice qui part d'une valeur entière et subit des incrémentations successives, entières elles aussi, jusqu'à la valeur finale de fin d'exécution de la boucle.
Le déclarer comme réel ne constituerait pas vraiment une erreur, mais ralentirait l'exécution de la boucle.
En outre, on utiliserait plus d'espace mémoire que nécessaire (seule la partie entière de l'indice sera utile).
Le Fortran (langage de haut niveau d'où dérive le Basic), prévoit d'ailleurs à cet effet que tous les noms qui commencent par une lettre comprise entre I et N désignent systématiquement des variables entières, sans qu'il soit nécessaire de les déclarer.
C'est donc avec profit qu'on adoptera la déclaration DEFINT I-N et qu'on réservera ces lettres initiales aux indices de boucles.

Instructions d'affectation

Ce sont les instructions qui servent à attribuer à des variables ou à des constantes une valeur déterminée, numérique ou alphanumérique et résultant éventuellement d'un calcul.
Toute grandeur se mentionne dans un programme par son nom symbolique, auquel le système d'exploitation associe une place déterminée en mémoire.
Utiliser ce nom symbolique revient à prélever le contenu de l'emplacement de mémoire ou à y écrire une donnée.
Schématiquement, tout se passe un peu comme si le nom d'une variable représentait une adresse de mémoire où l'on stocke, dans une mise à jour continuelle, les valeurs que cette variable prend successivement.

LET

L'instruction d'affectation la plus simple est :

LET nom = valeur

On demande ainsi à la machine d'écrire la donnée « valeur » dans l'emplacement de mémoire « nom ».
L'instruction LET A = 2.5 place la valeur 2.5 dans un emplacement mémoire qui prend alors le nom A.
Le symbole A sera par la suite mentionné à la place de sa valeur numérique.
Dans la plupart des langages, on peut omettre LET et écrire simplement A = 2.5.
Le signe = (prononcé « reçoit » et non « égale ») suffit alors à symboliser une instruction d'affectation.
On mémorisera ainsi le résultat de n'importe quelle opération dans une variable, qu'on pourra ensuite afficher, imprimer, ou utiliser dans de nombreux calculs.
L'instruction d'affichage (qui sera étudiée plus loin) sera mentionnée ici, et pour l'instant, sous sa forme la plus simple : PRINT nom.
Cette instruction provoque l'affichage à l'écran de la valeur associée au symbole « nom ».
Dans le cas de l'exemple précédent (LET A = 2.5), l'instruction PRINT A affichera à l'écran le nombre 2.5.
Calculons, par exemple, la surface d'un cercle de 3 cm de rayon (R).
Nous appliquerons la formule Surface = $3.14 \times R^2$, puis remplacerons R par 3, ce qui donne : Surface = 3.14×3^2.
Cette instruction constitue, à elle seule, presque tout le programme :

```
10' OPTION BASE 1 est inutile puis-
15' qu'aucun indice n'est utilisé
20 SURFACE = 3.14 × 3 ∧ 2 ' le symbole
25' ∧ indique l'élévation à une puissance
30 PRINT SURFACE
40 END
RUN (lancement de l'exécution)
28.26 (résultat du calcul)
```

La ligne 10 est un commentaire (symbole ').
La ligne 20 contient le calcul de l'affectation du résultat à la variable SURFACE.
A la ligne 30 se trouve l'instruction d'affichage de SURFACE, et la ligne 40 termine le programme.
Plusieurs instructions peuvent tenir sur une même ligne, à condition de les séparer par le symbole : (ou / sur certains systèmes). Ainsi, en omettant les commentaires, on obtient le programme suivant :

```
20 SURFACE = 3.14 × 3 ∧ 2 : PRINT
SURFACE : END
RUN
28.26
```

En fait, ce programme n'a aucune utilité pratique car il faut le récrire pour chaque nouvelle valeur du rayon.
Or, un programme d'application, pour être adapté à d'autres données, doit toujours avoir été paramétré.
Dans le programme ainsi récrit :

```
10 R = 3
20 SURFACE = 3.14 x R ∧ 2
30 PRINT SURFACE
40 END
```

le rayon n'est plus désigné, en ligne 20, par sa valeur numérique mais par un symbole : il est devenu une variable. En changeant seulement la valeur numérique affectée à R en ligne 10, on pourra calculer la surface de n'importe quel cercle.
Cependant, le programme n'est pas encore réellement parfait, puisqu'il faut modifier une instruction (ligne 10) pour chaque nouveau calcul.
Le seul moyen de le généraliser à toutes les valeurs de R est de donner à l'utilisateur la possibilité de fixer lui-même, à chaque fois, la nouvelle valeur de R.
Pour cela, on utilise l'instruction INPUT R, qui stoppe la machine jusqu'à ce que la valeur de R soit entrée au clavier.
Une fois saisie, cette valeur est affectée à la variable et l'exécution se poursuit comme précédemment.

```
10 INPUT R
20 SURFACE = 3.14 × R ∧ 2
30 PRINT SURFACE
40 END
```

Lors du déroulement du programme, la machine s'arrête à l'instruction 10 et affiche le signe ? sur l'écran, pour informer l'utilisateur qu'elle attend l'entrée d'une donnée.
L'opérateur tape la valeur numérique de R au clavier, puis la valide en actionnant la touche RETURN.

La variable R prend alors la valeur saisie, valable dans toutes les instructions où elle apparaît, et ce, jusqu'à l'affectation suivante. Le symbole ^, utilisé à la ligne 20, désigne l'élévation à une puissance. Toutes les machines ne comportent pas cet opérateur, et on doit alors expliciter la multiplication (ici $R^2 = R \times R$) ; la ligne 20 devient :
SURFACE = 3.14 × R × R.
La page 373 reproduit le listing d'une partie d'un sous-programme qui calcule, à partir de paramètres en entrée, la surface et le périmètre de figures planes.
La valeur de l'indicateur K indique la figure choisie, et c'est en fonction de cette valeur (1, 2, etc.) qu'est sélectionnée la formule appropriée (lignes 1670, 1700, etc.). Nous verrons plus loin comment s'effectue cette sélection.
Le symbole : a permis d'aligner plusieurs instructions correspondantes à la suite l'une de l'autre sur une seule ligne.
Toutes les lignes contenant le symbole ' après leur numéro sont des commentaires.
En entrée, on fournit les données suivantes : type de figure géométrique (K) et valeurs nécessaires au calcul ; on obtient en sortie la surface de la figure (variable SURF) et son périmètre (variable PER).
Si l'on pose, par exemple, K = 5 (cercle) et COTE1 = 7 (dans un cercle, Côté1 désigne le rayon et les autres variables sont donc inutiles), on obtient SURF = 153.86 et PER = 43.96 (PER désigne la circonférence). Si l'on veut ensuite calculer la surface d'un losange (K = 4), il faudra fournir trois données à la machine, le côté et les deux diagonales.
Remarquons que l'instruction END n'apparaît pas dans le listing, car elle est remplacée par RETURN, puisqu'il s'agit d'un sous-programme. L'instruction de fin se trouve donc dans le programme principal correspondant.
LET, qui est facultatif, ne figure presque jamais dans les instructions d'affectation. On écrit donc, (par ex.) A = 21 au lieu de LET A = 21. Cette forme implicite est le seul exemple d'instruction qui ne commence pas par un « mot réservé » du langage, c'est-à-dire un mot désignant une opération donnée.
Dans tous les autres cas, le mot réservé doit apparaître derrière le numéro de ligne ou le symbole : .
Comme leur nom l'indique, les mots réservés (codes d'instructions ou mots clés) ne peuvent pas être utilisés pour désigner des variables dans les programmes. Si nous avions, par exemple, utilisé PRINT au lieu de PER pour le périmètre, nous aurions provoqué une erreur, car l'interpréteur aurait reconnu un mot réservé et aurait tenté d'exécuter une instruction d'affichage erronée en lui affectant une valeur comme une variable.
Certains interpréteurs Basic sont capables de discerner si un mot désigne une instruction ou une variable, selon son contexte. Ils restent cependant l'exception.

DATA, READ et RESTORE

Les instructions DATA et READ permettent également d'affecter des valeurs à une ou plusieurs variables. S'il faut toujours les utiliser conjointement, elles peuvent cependant ne pas se trouver au même endroit du programme.
La machine sait que le code DATA est suivi d'une liste de nombres ou de lettres, constituant des valeurs à affecter à des variables. Ces dernières sont mentionnées après l'instruction READ. Les instructions

```
10 READ X
20 DATA 3.5
```

affectent à X la valeur 3.5, de même que l'instruction X = 3.5.
Il existe cependant des différences entre DATA et l'affectation ordinaire :
- on peut attribuer des valeurs à plusieurs variables par une seule instruction DATA ;
- les valeurs du DATA peuvent être utilisées plusieurs fois pour des variables différentes.

L'instruction DATA permet donc des affectations multiples. Si l'on veut, par exemple, affecter respectivement les valeurs 2, 7.3, 15 et 174 aux variables R, VAL, NOM et ETAT avec la première instruction, on écrira :

```
10 R = 2 : VAL = 7.3 : NOM = 15 : ETAT = 174
```

et les valeurs numériques 2, 7.3, etc. ne pourront plus être réutilisées.
Si une autre variable du programme, T par exemple, doit prendre la valeur 2, nous devrons ajouter une nouvelle instruction d'affectation (T = 2) sans pouvoir nous servir de l'instruction 10.

SOUS-PROGRAMME DE CALCUL DE SURFACES ET DE PERIMETRES

```
1500 '   ** Sous-programme de calcul de surfaces et de périmètres
1510 '
1520 '      SYMBOLES UTILISES :
1530 '                          SURF  = Valeur de la surface calculée
1540 '                          PER   = Valeur du périmètre
1550 '                          COTE1 = Longueur du premier côté (en entrée)
1560 '                          COTE2 = Longueur du second côté (en entrée)
1570 '                          COTE3 = Si nécessaire
1580 '                          COTE4 = Si nécessaire
1590 '   K = FLAG indiquant le type de figure
1600 '                              K = 1 CARRE
1610 '                              K = 2 RECTANGLE
1620 '                              K = 3 TRAPEZE
1630 '                              K = 4 LOSANGE
1640 '                              K = 5 CERCLE
1650 '
1660 '   Carré (K = 1)
1670 SURF = COTE1 * COTE1 : PER = 4 * COTE1 : RETURN
1680 ' Rectangle (K = 2)
1690 ' COTE1 = Longueur   COTE2 = Largeur
1700 SURF = COTE1 * COTE2 : PER = 2 * (COTE1 + COTE2) : RETURN
1710 ' Trapeze (K = 3)
1720 ' COTE1 = Pte base  COTE2 = Gde base  COTE3 et COTE4 = côtés obliques  HAUT = hauteur
1730 SURF = (COTE1 + COTE2) * HAUT/2 : PER = COTE1 + COTE2 + COTE3 + COTE4 : RETURN
1740 ' Losange (K = 4)
1750 ' COTE1 = Petite diagonale  COTE2 = Gde diagonale  COTE3 = Côté
1760 SURF = COTE1 * COTE2/2 : PER = 4 * COTE3 : RETURN
1770 ' Cercle (K = 5)
1780 ' COTE1 = Rayon
1790 SURF = 3.14 * (COTE1 * COTE1) : PER = 6.28 * COTE1 : RETURN
1800 '
```

Solutions du test 10

1 / La version en ASCII est celle qui porte le nom TEST ; elle est mémorisée sur le disque monté dans l'unité A.
Le programme ESSAI2, qui est protégé (P = protégé) se trouve sur la disquette qui était montée dans l'unité B quand on a entré la commande.
On ne peut apporter de corrections qu'aux versions sauvegardées sous les noms ESSAI et TEST (versions a et b).

---

2 / A l'exécution du programme principal seul, la valeur des variables D et E s'obtient (lignes 40 et 50) en remplaçant A, B et C par les nombres qui leur ont été affectés aux lignes 10, 20 et 30 :

$$D = 10 * (6 + 8) = 140$$

$$E = 2 * \frac{140}{6} = 46.666... = 46.\overline{6}$$

Le tiret placé au-dessus du 6 signifie que ce chiffre se répète indéfiniment. Ce résultat mathématiquement exact est arrondi par le système qui ne conserve qu'un nombre restreint de décimales, en fonction de la précision voulue.
En renumérotant le programme de 2 en 2 à partir de 20, nous obtenons :

```
20 A = 6
22 B = 8
24 C = 10
26 D = C * (A + B)
28 E = 2 * D/A
```

Fusionnons maintenant ce programme avec :

```
20 B = 2
30 F = D + E
```

La ligne 20 du premier programme (A = 6) est remplacée par celle du second (B = 2) et l'instruction 30 est ajoutée :

```
20 B = 2
22 B = 8
24 C = 10
26 D = C * (A + B)
28 E = 2 * D/A
30 F = D + E
```

Deux valeurs sont affectées successivement à la variable B (lignes 20 et 22). La dernière efface la première et B vaudra donc 8 dans les calculs qui suivent. En Basic, une variable est initialisée à 0 tant qu'aucune valeur ne lui est affectée. A est donc nulle aux lignes 26 et 28. La ligne 26 donne

$$D = 10\ (0 + 8) = 10 * 8 = 80$$

mais il se produit une erreur (débordement ou overflow) à l'exécution de la ligne 28, car on tente d'effectuer une division par 0 :

$$E = 2 * \frac{80}{0}$$

3 / La commande REN 10, 10, 2 entraîne la renumérotation des lignes 10, 20 et 30. On obtient :

```
10 A = 3
12 B = 2
14 C = 3 * (A + B)
```

La commande AUTO 20,5 provoque la numérotation automatique des lignes, de 5 en 5 et à partir de 20. Les instructions suivantes, F = 2 * (C + A) et PRINT F, portent donc les numéros 20 et 25. Voici le programme qui résulte de ces deux commandes :

```
10 A = 3
12 B = 2
14 C = 3 * (A + B)
20 F = 2 * (C + A)
25 PRINT F
```

La variation de pas entre les numéros de ligne importe peu puisque la numérotation est totalement arbitraire (mis à part le respect d'un ordre de progression).

---

4 / Il y a deux façons de procéder :
a) retaper complètement la ligne 20,
b) passer en mode éditeur.
Dans le premier cas, il suffit de taper, à la fin du programme, la nouvelle ligne 20 F = 4 * (C + A), qui remplacera automatiquement l'ancienne. Dans le second cas, il faut entrer la commande EDIT 20 pour se mettre en édition ; l'ancienne ligne F = 2 * (C + A) s'affiche à l'écran et on peut alors remplacer localement le 2 par un 4.

---

5 / L'instruction 20 TRON, provoquant l'affichage des numéros d'instructions exécutées, sera annulée à la ligne 50 par l'instruction TROFF. Les numéros de toutes les instructions exécutées comprises entre ces deux commandes vont apparaître à l'écran. Nous verrons ainsi s'afficher :

```
[30] [40]
```

car seules ces deux lignes se trouvent entre le TRON et le TROFF. Pour sauvegarder le programme, il faudra entrer la commande :

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

Il n'en est pas de même pour les valeurs qui suivent l'instruction DATA : elles peuvent servir à des variables différentes.
Avec READ et DATA, la précédente instruction d'affectation devient :

```
10 READ R, VAL, NOM, ETAT
.....
.....
273 DATA 2, 7.3, 15, 174
```

L'instruction DATA peut se trouver à différents endroits du programme mais, sur certaines machines, on doit la placer à la fin. Il ne s'agit pas, en effet, d'une instruction réellement exécutable, mais d'une zone de mémoire où sont enregistrées des valeurs (numériques ou non) et cela pourrait provoquer des erreurs si elle se trouvait dans le corps du programme.
L'affectation des valeurs se déroule dans l'ordre d'écriture : la première donnée (2) est donc attribuée à la variable R, la deuxième à VAL et ainsi de suite.
L'instruction READ peut être divisée en plusieurs blocs, contenant chacun un certain nombre de variables.
Au fur et à mesure, le système tient le compte

Concerto pour clavier et microprocesseur

C'est à juste titre que le stand de la Moog Company à la foire musicale de Francfort s'enorgueillissait de cette inscription : « From the men who started it all », faisant référence à l'idée, alors novatrice, de Robert Moog. Si elle nous semble aujourd'hui d'une simplicité déconcertante, l'idée de contrôler par des tensions électriques, et non manuellement, des dispositifs électroniques (oscillateurs, filtres, amplificateurs, etc.) servant à générer et à transformer des sons, était alors inédite.
Des traitements qui demandaient auparavant des heures, voire plusieurs jours de travail dans un studio d'enregistrement, devenaient dès lors presque instantanés, rendant possible la construction des premiers synthétiseurs, appareils pouvant servir en direct sur scène.

Le ***synthétiseur*** *constitue l'instrument électronique par excellence. Offrant à l'utilisateur la plus large liberté d'action sur la production et la modification des timbres, il est aussi, par là-même peut-être, l'instrument le plus difficile à utiliser. En effet, s'il arrive qu'on trouve par hasard une combinaison intéressante, il est pratiquement impossible de produire un effet précis en procédant par tâtonnements, à moins de connaître parfaitement le fonctionnement de la machine et d'improviser à la perfection.*
Pour bien se servir d'un synthétiseur, il faut posséder des notions de base en acoustique et en mécanique ondulatoire si c'est possible, afin d'avoir une réelle maîtrise des différents éléments de l'appareil (oscillateurs, filtres, générateurs d'enveloppe, amplificateurs, modulateurs).

Contrairement à l'orgue électrique, le synthétiseur n'est pas nécessairement muni d'un ***clavier*** *semblable à celui d'un piano. Néanmoins, la plupart des modèles en possèdent un, car c'est tout de même le périphérique d'entrée le plus commode pour un musicien habitué aux instruments traditionnels.*
Cependant, il en existe de nombreux autres types, tels des systèmes modulaires (le Moog et Roland en particulier) qui peuvent parfaitement fonctionner sans clavier.
Le VCS3, qui dut sa célébrité aux Pink Floyd, à Klaus Schulze et à bien d'autres, possédait accessoirement un clavier, mais constituait à lui seul un véritable petit studio de musique électronique.

Le clavier peut notamment être remplacé par un ***séquenceur,*** *ou contrôleur de bande. C'est un instrument très répandu qui mémorise des tensions de contrôle successives (s'il est analogique), ou des informations numériques. On enregistre le plus souvent des instructions correspondant à des notes et à des timbres particuliers. Le séquenceur guide ensuite le synthétiseur et lui fait jouer, à la vitesse désirée, les notes préprogrammées.*
Il est possible de programmer le séquenceur à partir du clavier en jouant une suite de notes qu'on peut ensuite faire reproduire par le séquenceur pendant qu'on exécute un autre morceau au clavier.
Il est facile de résoudre le problème de l'enregistrement avec une machine numérique : il suffit d'emmagasiner les informations dans des mémoires (RAM, bandes et autres supports magnétiques) semblables à celles d'un ordinateur.
Un tel instrument peut, par exemple, posséder seize mémoires de travail. Il est possible de mémoriser dans chacune d'elles l'ensemble des réglages des différents éléments de la machine correspondant à un timbre particulier. Il suffit d'appuyer sur un bouton pour passer très vite d'un registre à un autre. Le contenu de ces mémoires se conserve généralement même après qu'on ait éteint l'instrument, et, sur les machines les mieux conçues, il est possible de le sauvegarder sur bande, avec un magnétophone ordinaire. Le musicien, ayant programmé toutes les combinaisons qui l'intéressent, les archivera pour les rappeler au fur et à mesure de ses besoins. Les informations sont toutes enregistrées sous forme numérique, ce qui garantit la précision et la fiabilité propres à l'informatique.

*Avec leur séquenceur et leur batterie électronique, les synthétiseurs les plus sophistiqués ne sont pas loin de répondre à l'idéal de l'***homme-orchestre** *: le séquenceur peut travailler en polyphonie et en synchronisa-*

tion avec la batterie et enregistrer des milliers de notes. Pendant que ces deux instruments répètent les morceaux enregistrés à une vitesse modifiable à volonté, le musicien exécutera solos, mélodies et improvisations.
Avec une bonne amplification, l'effet obtenu est souvent impressionnant.
Korg a également proposé une version améliorée du Polysix, le Poly-61, dont les douze oscillateurs à contrôle numérique permettent de produire six voix (c'est-à-dire de jouer simultanément jusqu'à six notes sur le clavier). Il est muni d'un interface pour enregistrer les programmes sur bande et possède plusieurs effets sonores automatisés. L'un d'eux permet de faire durer une note en point d'orgue. Un autre dispositif permet au Poly-61 d'enregistrer et de restituer arpèges et accords, en synchronisation avec le séquenceur ou une batterie électronique externe. La mémoire de travail peut contenir soixante-quatre programmes en même temps.

Les modèles DX Yamaha appliquent un procédé original de génération des sons, la « synthèse numérique à modulation de fréquence par algorithmes programmables ». De plus, ils autorisent le doigté dynamique : comme un piano, le clavier tient compte de la force avec laquelle les touches sont enfoncées. Contrairement au piano, pour lequel seule compte l'impulsion initiale, le clavier de ces synthétiseurs est sensible aux variations ultérieures de pression. On peut donc augmenter le volume d'une note de faible intensité au départ en appuyant plus fort sur la touche.
Parmi les nouveaux synthétiseurs, remarquons également le Chroma, conçu par Arp, l'un des plus grands fabricants d'instruments électroniques.
C'est un synthétiseur polyphonique entièrement programmable, basé sur une technologie à mi-chemin entre l'analogique et le numérique. Il est équipé de deux microprocesseurs, dont l'un (7 K de RAM et 16 K de ROM) prend en charge tous les contrôles du son, tandis que l'autre veille à la sensibilité du doigté. Il mesure, en effet, au 1/1000 de seconde près, la vitesse à laquelle une touche est frappée et simule le doigté en fonction de cette vitesse. La RAM du microprocesseur principal permet d'emmagasiner une cinquantaine de programmes, qu'on sélectionne sur un tableau de commandes. Des batteries de secours permettent de conserver les programmes si on éteint le synthétiseur. On peut également les enregistrer sur des cassettes. L'appareil est fourni avec une cassette de 150 programmes.

Le secteur le plus fascinant, où beaucoup reste encore à découvrir, est celui de la **musique assistée par ordinateur**. On trouve notamment dans le commerce de nombreux micro-ordinateurs munis d'oscillateurs produisant mélodies ou effets sonores. Les possibilités deviennent réellement intéressantes lorsque l'ordinateur se complète de cartes enfichables spéciales, ou s'il est couplé à un synthétiseur dont il contrôle tous les réglages. Extensions et programmes prometteurs ont déjà été conçus, par exemple, pour l'Apple II. Citons notamment le Music System et l'Alpha Syntauri. Inspiré de deux produits de Mountain Computer Music System, ce dernier fonctionne comme un synthétiseur numérique polyphonique à huit voix. Il permet de créer des timbres (par synthèse additive ou en dessinant sur l'écran la forme de l'onde recherchée) et possède un clavier avec contrôle du doigté.
Signalons, parmi les dernières nouveautés, un programme de composition musicale mis

Synthétiseur à modulation de tension, à mémoire numérique et à mécanisme de track tape.

Barry Plummer

En 1972, le groupe « Yes » enregistra « Close to the edge », l'un des premiers albums de musique rock produite avec synthétiseur.

au point par Amdek pour l'Apple II, mais dont les versions compatibles avec d'autres ordinateurs individuels sont sérieusement étudiées : certaines sont déjà commercialisées. Un périphérique spécial (le CMU-800, version améliorée du Microcomposer de Roland) permet d'écrire morceaux et arrangements, et de contrôler jusqu'à huit synthétiseurs.

Si les machines très perfectionnées sont encore réservées aux grands studios et aux musiciens célèbres (elles coûtent un minimum de 100 000 F), l'évolution dans ce domaine se poursuit à un rythme rapide et le prix de certaines d'entre elles devrait diminuer d'ici quelques années.

Citons en exemple les performances de l'une de ces machines : le Wave 2.2 et son complément, le Waveterm.

Le Wave 2.2 comporte seize oscillateurs et trente « tables » de soixante-quatre formes d'onde chacune (soit, au total, deux mille formes d'onde environ). Ces tables sont produites numériquement ; on peut les filtrer et les mixer par microprocesseur, grâce à un tableau de commande mi-numérique, mi-analogique, qui satisfait à toutes les exigences et favorise un réglage plus poussé de l'utilisateur.

Le Digital Recording System permet d'enregistrer les sons et les airs joués au clavier sur huit pistes différentes, réglables indépendamment lors du play-back.

Déjà très achevé par lui-même, le Wave 2.2 donne toute sa mesure en association avec le Waveterm, qui comprend un terminal vidéographique à disquettes, et une unité de traitement des signaux audio.

Cette dernière accepte en entrée les sons captés par un microphone, les analyse et affiche le tracé de leur onde ; avant de les transmettre au Wave 2.2, on peut les corriger à l'aide de dix touches qui se trouvent au-dessous de l'écran.

On peut donc enregistrer une voix humaine, la reproduire au clavier en modifiant sa fréquence et obtenir ainsi une voix de basse, de ténor ou de soprano, quelle que soit sa hauteur initiale.

On peut d'ailleurs obtenir une reproduction d'une qualité extraordinaire. Toutefois, le Wave 2.2 ne se limite pas à reproduire les sons, et les traitements qu'on peut réaliser à l'aide de filtres, générateurs d'enveloppe et dispositifs de mixage produisent les effets les plus surprenants.

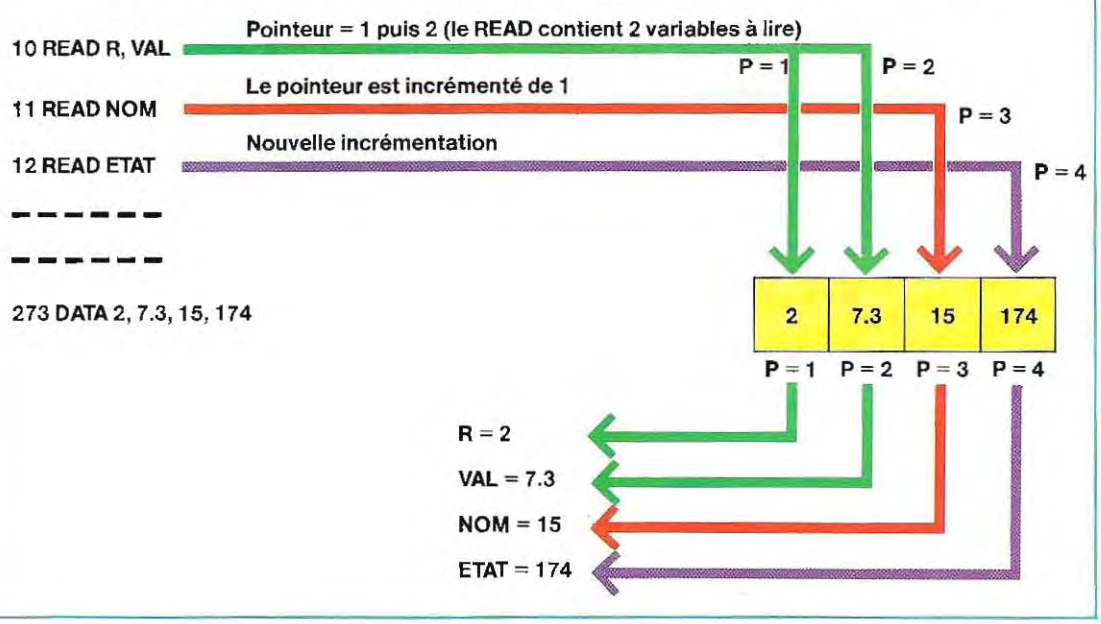
des valeurs déjà affectées au cours de l'exécution afin de pouvoir continuer dans l'ordre. On peut décomposer la ligne 10 de la façon suivante :

```
10 READ R, VAL
11 READ NOM
12 READ ETAT
...
273 DATA 2, 7.3, 15, 174
```

Le résultat est le même, mais les affectations sont réparties sur trois lignes.
Le mécanisme d'affectation des valeurs est illustré par le schéma ci-dessous. Le système associe à l'instruction DATA un pointeur (appelé P* sur la figure) qui indique la valeur à lire. P est initialisé à 1 par le premier READ et désigne d'abord la première valeur de la donnée. A chaque nouvelle affectation, une incrémentation de 1 le fait pointer vers la valeur suivante.
Comme cela apparaît sur le schéma, la première instruction READ concerne deux variables. La valeur 2 est associée à la première (P = 1). P est alors incrémenté (P = P + 1 = 2) ; il associe donc la seconde valeur (7.3) à la variable suivante, et ainsi de suite. A la dernière lecture, P prend sa valeur maximale (P = 4). D'autres tentatives d'affectations provoqueraient une erreur car P a atteint son maximum. Si l'on veut réutiliser les valeurs de la donnée, il faut réinitialiser le pointeur, ce qui sera fait avec l'instruction RESTORE.
Après l'exécution de cette instruction insérée en n'importe quel point voulu du programme, on peut de nouveau accéder à toutes les valeurs du DATA en partant de la première. L'emploi de RESTORE est illustré en haut de la page 380. Les lectures successives de cinq variables (A, B, C, X, Y), demandées aux lignes 20 et 30, déplacent le pointeur de la première valeur de l'instruction DATA (8) à la cinquième (25). La prochaine valeur lue devrait donc être 73. Mais la ligne 50 contient une instruction RESTORE qui remet le pointeur à 1 et lui fait donc indiquer la première valeur du DATA. L'instruction de lecture 60 entraîne donc l'affectation des valeurs 8, 19 et 11 aux variables E, F et G. En l'absence de RESTORE, le pointeur ne serait pas réinitialisé et il associerait aux variables de la ligne 60, la sixième valeur de DATA et les suivantes. Les instructions des lignes 40 et 70 impriment le contenu des variables.

* Le nom P et les valeurs attribuées au pointeur ne sont donnés qu'à titre d'exemples destinés à illustrer le mécanisme physique d'affectation par l'instruction DATA, qui est pris en charge par l'interpréteur et qui reste totalement transparent pour le programmeur.

READ ET DATA : MODE D'ACTION DU POINTEUR DES DONNEES



EMPLOI DE L'INSTRUCTION RESTORE (1)

```
10 OPTION BASE 1
20 READ A,B,C
30 READ X,Y
40 LPRINT "A = ";A, "B = ";B, "C = ";C, "X = ";X, "Y = ";Y
50 RESTORE
60 READ E,F,G
70 LPRINT "E = ";E, "F = ";F, "G = ";G
80 DATA 8,19,11,2.7,25,73,80,9.1
```

```
A = 8     B = 19    C = 11    X = 2.7    Y = 25
E = 73    F = 80    G = 9.1
```

En supprimant la ligne 50, on obtient

```
A = 8     B = 19    C = 11    X = 2.7    Y = 25
E = 73    F = 80    G = 9.1
```

EMPLOI DE L'INSTRUCTION RESTORE (2)

```
10 OPTION BASE 1
20 READ A,B,C
30 LPRINT "A = ";A, "B = ";B, "C = ";C
40 READ X,Y
50 RESTORE 90
60 READ K,Z
70 LPRINT "X = ";X, "Y = ";Y, "K = ";K, "Z = ";Z
80 DATA 7,9,200
90 DATA 26,48,11,90
```

```
A = 7     B = 9     C = 200
X = 26    Y = 48    K = 26    Z = 48
```

En supprimant 90 en ligne 50, on obtient :

```
A = 7     B = 9     C = 200
X = 26    Y = 48    K = 7     Z = 9
```

S'il existe plusieurs DATA, on peut préciser, derrière l'instruction RESTORE, le numéro de ligne concerné. Seules les valeurs contenues dans cette ligne et les suivantes sont alors réutilisables. Dans le programme ci-dessus, l'instruction RESTORE 90 qui a été écrite à la ligne 50, rend accès aux valeurs de DATA, seulement à partir de la ligne 90. L'instruction 40 affecte à X et Y les valeurs 26 et 48 : le pointeur désigne ensuite la troisième valeur de la ligne (11). Mais, l'instruction RESTORE 90 réinitialisant le pointeur à la première valeur du DATA, l'instruction de lecture suivante (ligne 60) attribuera donc les valeurs 26

MAUVAISE UTILISATION DES INSTRUCTIONS READ ET DATA

```
10 OPTION BASE 1
20 READ A,B
30 X=A+B
40 READ C,D,F
50 PRINT X,C,D,F
60 DATA 3,7,9,11
```

```
Out of DATA in 40
```

EMPLOI DE L'INSTRUCTION DATA POUR DES CHAINES DE CARACTERES

```
10 OPTION BASE 1
20 READ A$
30 READ B$
40 READ C$
50 READ D$
60 READ E$
70 F$=A$+C$+B$+C$
80 G$=D$+C$+C$
90 H$=E$+C$
100 PRINT F$
110 PRINT G$
120 PRINT H$
130 L$=F$+G$+H$
140 PRINT L$
150 DATA "Nom"
160 DATA "Prénom"
170 DATA "..........."
180 DATA "Rue"
190 DATA "Commune"
```

```
Nom..........Prénom..........
Rue.....................
Commune..........
Nom..........Prénom..........Rue.....................Commune....
```

et 48 à K et à Z. Si l'on ne précisait pas de numéro de ligne après RESTORE, le pointeur se positionnerait au début du premier DATA (ligne 80) et la ligne 60 déterminerait l'affectation des valeurs 7 et 9 (et non 26 et 48) aux variables qu'elle contient.

Le nombre des valeurs contenues dans une instruction DATA est au moins égal au nombre de variables à lire (s'il est supérieur, les valeurs superflues sont ignorées). Dans le cas contraire, le système diagnostique une erreur et stoppe l'exécution. Le programme présenté en haut de cette page est celui d'un programme où on tente de lire cinq variables

dans un DATA de quatre valeurs, et sans utiliser de RESTORE. L'interpréteur interrompt le programme et émet le message « Out of DATA ». On peut également utiliser l'instruction DATA pour traiter les variables alphanumériques. Les valeurs de DATA s'écrivent alors entre guillemets (le symbole " marque le début ou la fin d'une chaîne de caractères. Le second programme de la page 381 utilise des chaînes et leur affecte, par les instructions 20 et 60 les valeurs contenues dans les DATA. Les lignes 70 et 80 concatènent ces variables en insérant des points aux endroits désirés (la chaîne C$ contient 10 fois le signe), créant trois chaînes destinées à l'impression de formulaires d'état-civil. L'impression peut se faire sur une ou plusieurs lignes, selon la façon dont on combinera les différentes chaînes.

LSET et RSET

Ces instructions permettent de déplacer des variables en les alignant à gauche (LSET; L = Left), ou à droite (RSET; R = Right).

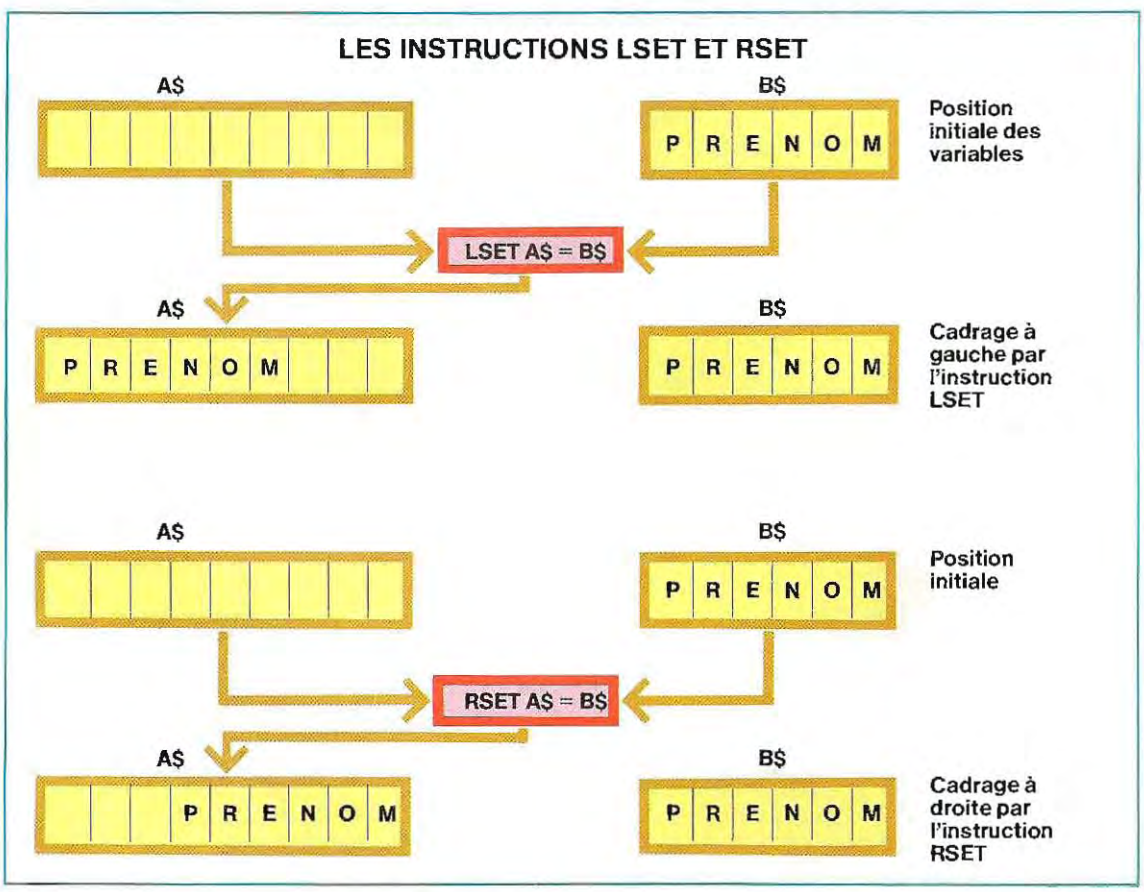
Ainsi, l'instruction

LSET A$ = B$

affecte à A$ le contenu de B$ en le cadrant à gauche (B$ demeure inchangée).
On utilise principalement ces instructions pour préparer les données à l'impression ou à la mémorisation sur disque.
On évite ainsi d'avoir recours à des commandes de mise en page spéciales quand on a besoin d'imprimer les données. De plus, LSET et RSET convertissent, tout en les cadrant, les variables numériques et alphanumériques. La plupart des systèmes ne pouvant stocker que des données en ASCII (chaînes), ces instructions s'avèrent très précieuses : on est assuré de retrouver les valeurs exactes lors de la conversion des caractères en nombre après lecture du disque.

SPACE$(N)

Cette instruction représente en réalité une fonction d'une variable entière N. Elle attribue



UTILISATION DES INSTRUCTIONS SPACE$, LSET ET RSET

```
1000 '
1010 ' ** Emploi des instructions : SPACE$ LSET RSET
1020 '
1030 ' Attribution aux variables PRÉNOM1$, PRÉNOM2$, PRÉNOM3$, PRÉNOM4$
1040 ' d'une longueur de 20 caractères et initialisation à blanc
1050 PRENOM1$=SPACE$(20):PRENOM2$=SPACE$(20):PRENOM3$=SPACE$(20)
1060 :PRENOM4$=SPACE$(20)
1070 ' Lecture de 4 prénoms. On peut écrire les 4 instructions (1090 à 1120)
1080 ' sur une seule ligne en utilisant le signe ":"
1090 INPUT A1$  ' Ces instructions lisent des chaînes de caractères et
1100 INPUT A2$  ' les affectent aux variables (A1$,A2$,A3$,A4$)
1110 INPUT A3$
1120 INPUT A4$
1130 '
1140 ' Cadrage à gauche des valeurs de A1$,...A4$
1150 ' On traite 2 variables par ligne, mais on pourrait toutes les
1160 ' traiter sur une seule ligne, ou chacune sur une ligne séparée
1170 LSET PRENOM1$=A1$:LSET PRENOM2$=A2$
1180 LSET PRENOM3$=A3$:LSET PRENOM4$=A4$
1190 '
1200 ' Impression des données telles qu'on les a entrées (A1$..A4$)
1210 '
1220 PRINT A1$:PRINT A5$:PRINT A3$:PRINT A4$
1230 PRINT:PRINT  ' Ces instructions servent à sauter 2 lignes entre les résultats
1240 ' Impression des données cadrées à gauche (PRENOM1$, etc.)
1250 '
1260 PRINT PRENOM1$:PRINT PRENOM2$:PRINT PRENOM3$:PRINT PRENOM4$
1270 PRINT:PRINT  ' Ces instructions servent à séparer les résultats
1280 ' Cadrage à droite
1290 '
1300 RSET PRENOM1$=A1$:RSET PRENOM2$=A2$
1310 RSET PRENOM3$=A3$:RSET PRENOM4$=A4$
1320 '
1330 ' Impression des données cadrées à droite
1340 '
1350 PRINT PRENOM1$:PRINT PRENOM2$:PRINT PRENOM3$:PRINT PRENOM4$
1360 '
1370 END
```

```
PIERRE
CATHERINE
JULIE
OLIVIER

PIERRE
CATHERINE
JULIE
OLIVIER

            PIERRE
         CATHERINE
             JULIE
           OLIVIER
```

à une chaîne de caractères la longueur indiquée entre parenthèses (soit N caractères) et la remplit de blancs (code héxadécimal 40, selon la table de correspondances du code ASCII). En écrivant A$ = SPACE$(10), on fixe la longueur de A$ à 10 caractères initialisés à blanc.

L'allocation des chaînes est dynamique en Basic : la longueur d'une variable alphanumérique (nombre de caractères qu'elle contient), peut donc varier au cours du programme. Les problèmes d'alignement, lors de l'impression ou de la mémorisation, sont résolus par les instructions SPACE$ et LSET (ou RSET).

Le programme de la page 383 accepte quatre prénoms entrés au clavier (A1$, etc., instructions 1090 à 1120)*. L'instruction 1220 imprime les données telles qu'on les a entrées. La deuxième écriture (1260) intervient après le cadrage à gauche et la dernière (1350) après le cadrage à droite (voir listing).

L'instruction SPACE$(N), nécessaire au paramétrage de la longueur d'une chaîne (en fonction du résultat d'un calcul, par exemple), n'existe pas sur toutes les machines. On devra alors, dans une boucle, construire une chaîne d'espaces blancs de longueur donnée. L'organigramme correspondant se trouve ci-dessous mais nous présenterons les instructions de boucle lors d'une prochaine étude.

En affectant à N la valeur 20 avant d'appeler cette routine, nous obtiendrons le même résultat qu'avec l'instruction A$ = SPACE$(20).

On trouvera page 386 un programme commenté qui établit des listes d'adresses.

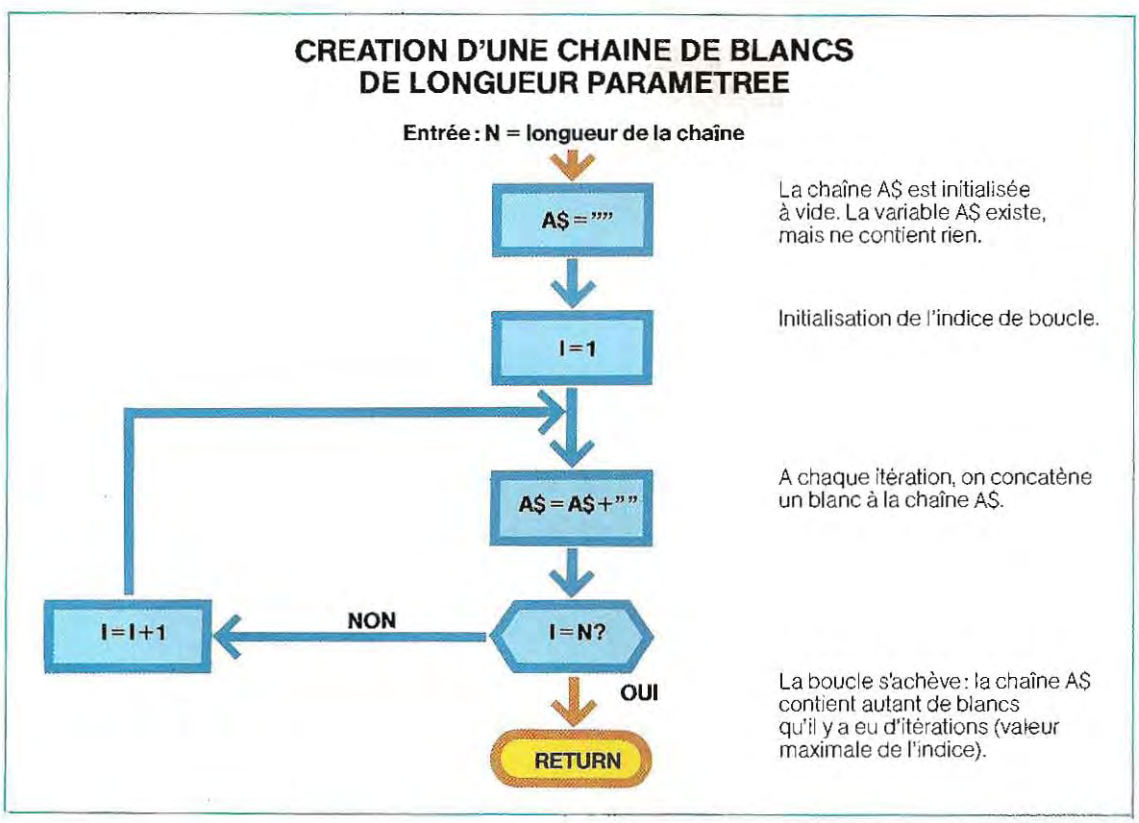
Tel qu'il est présenté, ce programme ne peut lire qu'un nom à la fois car les instructions d'itération ont été remplacées par des lignes d'astérisques (lignes 150 à 410).

SPC(N)

Comme SPACE$(N), cette instruction est une fonction. Elle a un effet semblable et c'est pourquoi nous la traitons ici, même si elle ne réalise aucune affectation.

SPC(N) permet d'imprimer ou d'afficher N blancs.

* Nous n'avons pas encore vu l'instruction INPUT. Elle permet la saisie des données au clavier et leur affectation dans une variable.



Test 11

1 / Classer les constantes suivantes selon leur catégorie (entière, simple et double précisions, chaîne)

a) 12.57 b) "Allo" c) 79.3D4 d) 9159E6 e) &H12
f) 92735 g) &177 h) &HAF i) "AF" j) %2121

2 / Si l'on pose A% = 3, B = 3, C# = 3, D$ = "3", lesquelles parmi les expressions suivantes sont erronées ?

a) D% = A% + 7 b) D% = A% + 46721 c) D% = C# * B!
d) R! = B! * 120 e) D# = D$ + D# f) H$ = "7" + D$

3 / Si le tableau A (10) est dimensionné sans autre indication, combien compte-t-il d'éléments ?

4 / A l'aide de la fonction MOD dessiner l'organigramme d'un programme déterminant la divisibilité par 2, 3, 5, 7, 11 d'un nombre quelconque entré au clavier.

5 / Généraliser le programme précédent en prévoyant l'entrée au clavier des diviseurs (10 nombres au maximum). Le programme s'articulera selon les étapes suivantes :

1) lecture du nombre de diviseurs à saisir (NMX) ;
2) boucle de lecture des diviseurs (de 1 à NMX) et leur mémorisation dans un tableau A dimensionné à 10 éléments ;
3) entrée de la valeur dont on veut vérifier la divisibilité (par chacun des nombres précédemment introduits) ;
4) contrôle de divisibilité et impression des résultats ;
5) préparation du programme pour une nouvelle entrée.

Le programme se termine lorsque le chiffre 0 est introduit comme valeur de NMX.

Les solutions du test se trouvent pages 398 et 399.

Ainsi, l'instruction PRINT SPC (6) provoque l'affichage de 6 blancs de la même manière que l'ensemble des deux instructions A$ = SPACE$(6) : PRINT A$. Dans ce cas, le contenu de la chaîne de caractères A$ (6 blancs) est imprimé. L'instruction SPC (N) reste cependant assez peu utilisée.

SWAP

Cette instruction, qui sert à échanger les valeurs de deux variables, peut être appliquée à un format quelconque (entier, réel en simple ou double précision, chaîne de caractères) à condition que les deux variables soient homogènes.

Exemple :

SWAP A, B	Echange entre eux les contenus de A et B.
SWAP A, B#	Erreur. La variable A est en simple précision alors que B est en double précision.
SWAP A$, B$	Echange les contenus des chaînes de caractères A$, B$.

En insérant l'instruction SWAP A$, B$ dans le programme de la page 386 avant l'impression, par exemple en ligne 355, le nom s'écrit (deuxième colonne) à la place du prénom, et vice versa.

Instructions de boucle

Dans ce paragraphe, seules sont présentées les deux principales d'entre elles :

FOR... NEXT...
WHILE... WEND...

prévues en Basic 80. Les instructions propres à d'autres versions du Basic feront l'objet d'une étude ultérieure.

LECTURE DES DONNEES D'ETAT CIVIL ET IMPRESSION SOUS FORME DE TABLEAUX

```
10  ' ** LECTURE DES DONNEES D'ETAT CIVIL ET IMPRESSION SOUS FORME DE TABLEAUX **
20  '
30  ' DECLARATIONS :
40  '
50  OPTION BASE 1              ' Les indices partent de la valeur 1
60  DEFINT I-N                 ' Tous les noms ayant pour initiale les lettres de I à N
70  '                            représentent des variables entières
80  ' ENTREES :
90  '
100  INPUT PRENOM$
110  INPUT NOM$
120  INPUT NUMERO$
130  INPUT RUE$
140  ' Les données brutes ne permettraient pas une impression ordonnée en tableaux
150  ' **********
160  ' Le format d'impression doit être :
170  '       Prénom          15 caractères dans la chaîne A$
180  '       Nom             15 caractères dans la chaîne B$
190  '       Numéro           4 caractères dans la chaîne C$
200  '       Rue             20 caractères dans la chaîne D$
210  '
220  ' Création des fenêtres d'impression A$, B$, etc.
230  '
240  A$ = SPACE$(15)            ' devra contenir PRENOM$
250  B$ = SPACE$(15)            ' devra contenir NOM$
260  C$ = SPACE$( 4)            ' devra contenir NUMERO$
270  D$ = SPACE$(20)            ' devra contenir RUE$
280  '
290  ' Cadrage des données lues dans les fenêtres d'impression
300  '
310  LSET A$ = PRENOM$
320  LSET B$ = NOM$
330  RSET C$ = NUMERO$    ' les chaînes de caractères représentant des valeurs
340  LSET D$ = RUE$       ' numériques sont justifiées à droite pour être lues en colonne
350  '
360  ' Les fenêtres d'impression A$, B$, etc. sont réunies pour former une seule chaîne
370  ' appelée LIGNE$
380  '
390  LIGNE$ = A$ + B$ + C$ + D$
400  PRINT LIGNE$
410  ' **********
420  END
```

```
LEON        ROUSSEAU          123 rue d'AQUITAINE

HERVE       TISSERAND        1234 rue du MAINE

ARMAND      ROY                 6 place de la VICTOIRE
```

FOR... NEXT...

Etudions plus en détail l'organigramme d'une boucle indicée classique exposée page 387. L'indice I varie entre les valeurs N1 et N2 (N2 > N1) avec un pas P. A la première exécution de la boucle, l'indice I prend la valeur N1. Successivement incrémenté du pas P, l'indice prend les valeurs I + P, I + 2P, etc., jusqu'à la valeur finale N2. La programmation de la boucle nécessite deux instructions : la première spécifie le pas (STEP), et les valeurs (N1 et N2) entre lesquelles l'indice doit varier, la seconde indique le point de « sortie de la boucle ». A ce point du programme, la valeur de l'indice est comparée à la limite supérieure (N2). Selon le résultat, on a la sortie de la boucle ou l'incrémentation de l'indice pour un nouveau passage.

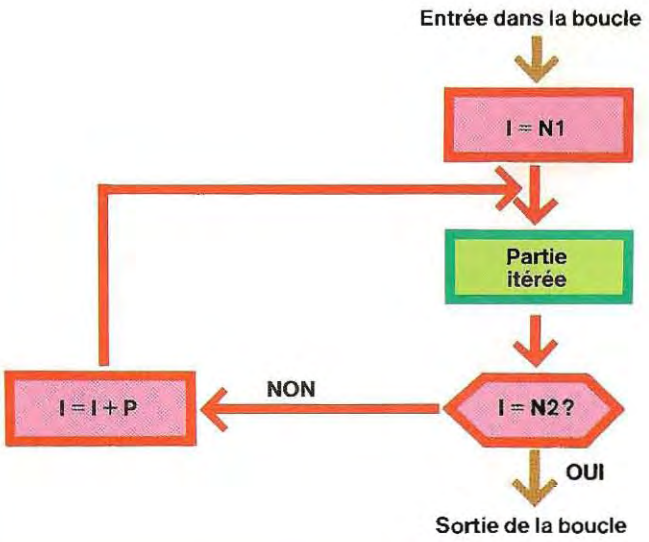
L'instruction spécifiant les paramètres est :

FOR I = N1 TO N2 STEP P

Le code opération FOR indique à la machine que l'instruction est une boucle ; l'indication I = N1 TO N2 STEP P en spécifie les modalités d'exécution (bornes, pas).
L'instruction de fin de boucle est NEXT I (NEXT = prochain).
Par conséquent, la traduction en Basic de l'organigramme est :

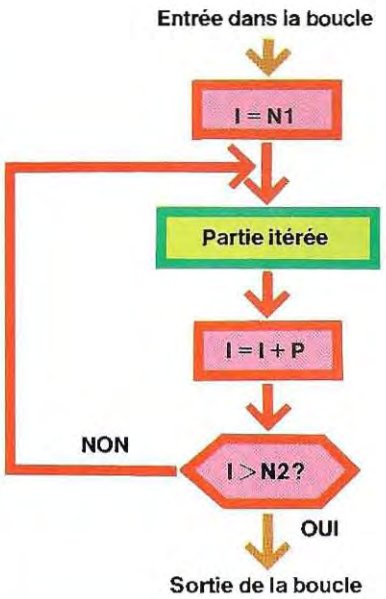
20 FOR I = N1 TO N2 STEP P

SCHEMA LOGIQUE D'UNE BOUCLE



I = Indice de boucle
N1 = Valeur initiale de l'indice
N2 = Valeur finale
P = Pas

ORGANIGRAMME DE DEROULEMENT D'UNE BOUCLE



(Partie itérée)
1270 NEXT I

Le premier schéma de la page 387 est l'équivalent logique du programme en langage machine, mais non sa forme exacte. Une comparaison avec le mécanisme d'exécution réel (schéma au bas de la page 387) met en évidence une différence dans la position du calcul de la nouvelle valeur de I (I = I + P).
Logiquement, et selon le premier schéma, ce calcul s'effectue avant le retour dans la boucle, tant que la condition I = N2 n'a pas encore été vérifiée. Lorsque l'indice I est parvenu à la valeur N2, la boucle s'interrompt avant une nouvelle itération. De cette manière, à la sortie de la boucle, I vaut donc N2.
En réalité, l'interpréteur (ou compilateur Basic) procède de manière différente: d'abord, l'indice est incrémenté (I = I + P, schéma du bas de la page 387) et on procède ensuite au test de comparaison interrompant éventuellement la boucle. En comparant les deux organigrammes, il paraît évident que dans le second cas, on ne peut pas sortir avec le test I = N2, car I prend sa valeur finale (I = N2) dans la boucle, juste après les calculs correspondant à I = N2 − 1. On sortira de la boucle sous la condition I > N2.
Précisons ici qu'à la sortie de chaque boucle, l'indice n'est pas égal à la limite supérieure (N2), mais à la limite supérieure augmentée du pas (N2 + P). Par exemple, la boucle d'indice désignée par la lettre J:

FOR J = 3 TO 11 STEP 2
(Partie itérée)
NEXT J

se termine avec la valeur J = 11 + 2 = 13.
Le déroulement de la boucle est transparent pour le programmeur: tous les tests, décisions et incrémentations sont effectués automatiquement par l'interpréteur.
Les paramètres d'une boucle (bornes et pas) doivent, dans la plupart des langages, être obligatoirement de type entier: l'instruction FOR I = 2.9 TO 3.1 STEP 0.1 est donc généralement erronée. Pour programmer une boucle à valeurs non entières, on transformera les valeurs en entiers pour l'instruction FOR, par exemple en les multipliant par 10, et on les divisera ensuite par 10 à l'intérieur de la boucle.
Le programme listé ci-dessous illustre cette technique en imprimant les nombres de 2.1 à 3 avec un pas de 0.1. L'interpréteur Basic considère les paramètres d'une boucle comme entiers. S'ils sont représentés par des noms symboliques non explicitement déclarés comme entiers (DEFINT), le système n'évaluera de toute façon que leur partie entière.
Bien qu'admise, l'utilisation de variables non entières n'est jamais avantageuse: elles occupent inutilement davantage d'espace

EXEMPLE DE BOUCLE A VALEURS NON ENTIERES

```
100  '
110  ' ** EXEMPLE DE BOUCLE A VALEURS NON ENTIERES **
120  '
130  ' Le programme imprime les nombres compris entre 2.1 et 3 par pas de 0,1 :
140  ' Valeur initiale de l'indice = 2,1  est portée à 21
150  ' Valeur finale               = 3    est portée à  30
160  ' Pas                         = 0.1  devient       1
170  ' On a donc introduit un facteur multiplicateur de 10, et il faudra
180  ' diviser l'indice avant de l'imprimer.
190  '
200  '
210  N1 = 2.1*10 : N2 = 3*10 : P = 0.1*10          ' Paramètres multipliés par 10
220  FOR I = N1 TO N2 STEP P                       ' BOUCLE
230  R = I/10
240  LPRINT R;       ' Le symbole ; indique qu'il ne faut pas passer à
250  NEXT I          ' La ligne après l'impression
260  END


 2.1   2.2   2.3   2.4   2.5   2.6   2.7   2.8   2.9   3
```

EXECUTION D'UNE BOUCLE SUR ECRAN

```
LIST
0  '        FICHIER = PHOTO 1
10 DEFINT A-Z
20 INPUT "QUELLE EST LA VALEUR MAXIMALE DE L'INDICE ";NF
30 INPUT "QUEL EST LE PAS DE LA BOUCLE          ";S
40 K=0
50 FOR I = 1 TO NF STEP S
60 K=K+1
70 PRINT "PAS: ";K,"INDICE: ";I
80 NEXT I
90 K=0
100 PRINT "BOUCLE INVERSEE"
110 FOR I = NF TO 1 STEP -S
120 K=K+1
130 PRINT "PAS: ";K,"INDICE: ";I
140 NEXT I
150 END
Ok
```

L'écran ci-contre fait apparaître le listage d'un programme illustrant l'utilisation d'une boucle FOR... NEXT.. Le programme demande à l'opérateur les paramètres de la boucle : valeur maximale (NF) de l'indice et pas (S) à prendre en considération. Notons que le pas S de la boucle est l'incrément de l'indice I, tandis qu'en phase d'affichage (et éventuellement d'impression), le mot PASSE introduit la valeur du compteur K, qui s'incrémente de 1 à chaque itération et permet de suivre le déroulement de la bande. PASSE équivaut donc au nombre d'itérations.

```
RUN
 QUELLE EST LA VALEUR MAXIMALE DE L'INDICE ? 8
 QUEL EST LE PAS DE LA BOUCLE                ? 2
PASSE: 1     INDICE: 1
PASSE: 2     INDICE: 3
PASSE: 3     INDICE: 5
PASSE: 4     INDICE: 7
BOUCLE INVERSEE
PASSE: 1     INDICE: 8
PASSE: 2     INDICE: 6
PASSE: 3     INDICE: 4
PASSE: 4     INDICE: 2
Ok
```

Cet écran et le suivant montrent deux exécutions différentes du programme. Ici, à la question : « quelle est la valeur maximale de l'indice ? », l'opérateur indique le chiffre 8, et impose, par la saisie suivante, un pas de 2. A chaque incrémentation, le programme affichera la valeur de l'indice I à côté du compteur, dans la boucle directe, et dans la boucle inversée.

```
RUN
 QUELLE EST LA VALEUR MAXIMALE DE L'INDICE ? 8
 QUEL EST LE PAS DE LA BOUCLE                ? 1
PASSE: 1     INDICE: 1
PASSE: 2     INDICE: 2
PASSE: 3     INDICE: 3
PASSE: 4     INDICE: 4
PASSE: 5     INDICE: 5
PASSE: 6     INDICE: 6
PASSE: 7     INDICE: 7
PASSE: 8     INDICE: 8
BOUCLE INVERSEE
PASSE: 1     INDICE: 8
PASSE: 2     INDICE: 7
PASSE: 3     INDICE: 6
PASSE: 4     INDICE: 5
PASSE: 5     INDICE: 4
PASSE: 6     INDICE: 3
PASSE: 7     INDICE: 2
PASSE: 8     INDICE: 1
Ok
```

Ici, on pose NF = 8 et S = 1. Puisque le pas est divisé par deux et NF inchangé, on exécutera deux fois plus d'itérations (passes) que précédemment. A la fin de l'exécution, le système répond par le message de commande exécutée (Ok).

EXECUTION D'UNE BOUCLE INTERPRETEE ET COMPILEE

```
10  ' ** COMPARAISON DES TEMPS D'EXECUTION D'UNE BOUCLE
20  '    AVEC INDICE REEL ET ENTIER
30  OPTION BASE 1  ' Cette instruction sert à faire débuter
40  '              les indices à la valeur 1
50  INPUT "N avec R = REEL "; N  ' Cette instruction sera expliquée
60  '                            par la suite
70  FOR R = 1 TO N STEP 2
80  NEXT R
90  PRINT N
100  DEFINT R     ' Impose à R et N le type entier
110  DEFINT N
120  INPUT "N avec R = ENTIER"; N     ' Cette instruction sera
130  '                                  expliquée par la suite
140  FOR R = 1 TO N STEP 2
150  NEXT R
160  PRINT N
170  END
```

Nombre d'itérations	Programme interprété (temps en secondes)		Programme compilé (temps en secondes)	
	R = Réel	R = Entier	R = Réel	R = Entier
10000	5	4	3.5	0.5
15000	8	6	5	1
25000	14	9.5	9	1
32000	17.5	12	12	1

mémoire et obligent l'interpréteur à exécuter des opérations supplémentaires (ici, extraction de la partie entière), avec la perte de temps qui en découle dans l'exécution de la boucle. Etant donné la lenteur de l'exécution du Basic interprété, toute surcharge se répercute lourdement sur le temps de déroulement des programmes. Le tableau ci-dessus, présentant, en secondes, les temps nécessaires à l'exécution d'une simple boucle, en donnera l'ordre de grandeur.

Dans les boucles, la valeur du pas, si elle n'est pas déclarée, est égale à 1, par défaut. Ainsi,

```
FOR I = N1 TO N2 STEP 1
FOR I = N1 TO N2
```

sont deux instructions équivalentes.

Les valeurs limites de l'indice (N1, N2) peuvent être calculées dans l'instruction même. Par exemple :

```
FOR I = 5 + N  TO  20 + N
```

est une forme valide de limites paramétrées. Les bornes de la boucle dépendent alors de la valeur de N, variable à laquelle on affecte une valeur avant d'entrer dans la boucle. Une forme particulière de calcul des bornes d'une boucle se présente lorsque l'une des deux est calculée à partir de la variable d'indice I.

Par exemple, avec les instructions :

```
10  I = 2
20  FOR  I = 6  TO  I + 25
```

la limite supérieure (I + 25) est obtenue d'après la valeur affectée à l'indice en ligne 10

(instruction 10 I = 2). Généralement l'interpréteur Basic calcule en premier la valeur de la limite supérieure, et ensuite seulement celle de la borne inférieure. L'instruction 10 affecte à I la valeur 2; par conséquent, en ligne 20 (I + 25), la limite supérieure est 2 + 25 = 27. La boucle se déroulera entre les valeurs I = 6 (I = 6 est établi à la ligne 20 après le calcul I + 25) et I = 27.
D'autres interprêteurs, au contraire, calculent d'abord la valeur initiale et ensuite celle finale; en ce cas, l'instruction 10 ne servirait à rien car, à la ligne suivante, on écrit I = 6. L'autre limite vaudra donc I + 25 = 31, et la boucle se déroulera entre 6 et 31.

Cette dernière méthode est caractéristique des interpréteurs Basic antérieurs aux versions actuelles, qui proposent parfois l'autre système.
Les boucles peuvent être « imbriquées » c'est-à-dire contenues les unes dans les autres, à condition que les premières ouvertes soient fermées en dernier (voir le chapitre consacré aux organigrammes).
Page 391, on a reproduit le listing de deux programmes : le premier est correct, le second contient une erreur dans la clôture des boucles. La page 384 illustrait un organigramme de génération d'une chaîne de longueur paramétrée, contenant seulement des blancs. La

EXEMPLES DE TROIS BOUCLES IMBRIQUEES

```
100 '
110 '** EXEMPLE DE TROIS BOUCLES IMBRIQUEES **
120 '
130 DEFINT I-N          ' Les indices sont déclarés comme entiers
140 FOR I=1 TO 5        ' Première boucle (sera fermée la dernière en ligne 230)
145 PRINT               ' Sert à séparer la boucle de 1 à 5
150 PRINT I             ' Imprime l'indice de boucle de plus haut niveau
160 FOR J=3 TO 9 STEP 2    ' Deuxième boucle (fermée ligne 220)
170 K=I*J
180 PRINT K;            ' Le symbole ; impose l'écriture sur une seule ligne
190 FOR L=10 TO 12      ' Troisième boucle (fermée ligne 210)
200 PRINT L;
210 NEXT L
220 NEXT J              ' Comparer la séquence des fermetures avec la position
230 NEXT I              ' de début des boucles correspondantes

RUN


 1
 3  10  11  12  5  10  11  12  7  10  11  12
 2
 6  10  11  12  10  10  11  12  14  10  11  12
 3
 9  10  11  12  15  10  11  12  21  10  11  12
 4
 12  10  11  12  20  10  11  12  28  10  11  12
 5
 15  10  11  12  25  10  11  12  35  10  11  12


100 DEFINT I-N
110 FOR L=3 TO 21
120 A=2*L+10
130 FOR N=7 TO 15
140 B=N*A
150 NEXT L              ' Les clôtures des deux boucles sont inversées,
160 NEXT N              ' générant ainsi une erreur

RUN

NEXT without FOR in 160
```

SOUS-PROGRAMME 1000 DE GENERATION DE CHAINES

```
1000 '
1010 ' ** SOUS-PROGRAMME DE GENERATION DE CHAINE= **
1020 '
1030 ' ENTREES :
1040 '          LS = Longueur (en caractères) de la chaine
1050 '          B$ = Symbole qui doit remplir la chaine
1060 '
1070 ' SORTIES :
1080 '          A$ = Chaine de longueur désirée remplie par le
1090 '               caractère B$
1100 A$ = ""        ' La chaine est initialisée à vide
1110 FOR I = 1 TO LS        ' BOUCLE sur la LONGUEUR
1120 A$ = A$ + B$           ' Le symbole B$ est concaténé à A$, un
1130 '                        nombre de fois égal à LS
1140 NEXT I
1150 RETURN
1160 '
```

EXEMPLE D'UTILISATION DU SOUS-PROGRAMME 1000

```
100 '
110 ' ** EXEMPLE D'UTILISATION DE LA ROUTINE 1000 (generation d'une chaine) **
120 '
130 DEFINT I-N
140 B$ = "*"        ' La chaine voulue sera constituée du caractère *
150 LS = 15         ' répété 15 fois
160 GOSUB 1000  ' Appel du sous-programme 1000
170 LPRINT A$
180 '
190 B$ = "X"        ' Nouveau symbole
200 GOSUB 1000  ' On n'assigne pas de nouvelle valeur à LS
210 LPRINT A$   ' La longueur reste donc 15
220 '
230 B$ = "1"
240 LS = 5
250 GOSUB 1000  ' A cet appel, symbole et longueur ont tous deux changé
260 LPRINT A$
270 '
280 LS = 260
290 GOSUB 1000  ' Cet appel générera une erreur dans la routine ; LS = 280
300 LPRINT A$   ' alors que les chaines ont au maximum 255 caractères
310 '
320 END
```

traduction en Basic de cet organigramme passe par l'utilisation des boucles ; l'unique adjonction nécessaire est un paramétrage ultérieur.

Ce paramétrage permettra de transmettre la longueur de la chaîne, mais aussi le symbole (unique) que l'on désire y faire apparaître. De cette façon, le programme pourra générer des chaînes de longueur voulue, et contenant un symbole quelconque. Sur cette page, figure le listing de cette routine, suivi d'un programme illustrant un exemple d'utilisation de cette routine. Le dernier appel au sous-programme, avec un paramètre de valeur erronée (LS > 255), générera une erreur si on ne prévoit pas un contrôle avant la boucle créant la chaîne, et le programme s'arrêtera. La meilleure parade consiste à prévoir, en entrée du sous-programme, un contrôle de validité des paramètres transmis par le programme appelant.

En cas d'erreur, on affectera aux paramètres une valeur conventionnelle, et un indicateur d'erreur sera activé.

De cette manière, l'exécution du programme n'est pas stoppée et l'indicateur d'erreur,

transmis au programme appelant, permettra de déterminer quelle routine a été appelée avec des valeurs erronées, et quelles sont les corrections possibles. Le schéma du bas de cette page et celui de la page 394 reproduisent les modifications apportées à l'organigramme initial pour inclure le contrôle du paramètre LS.

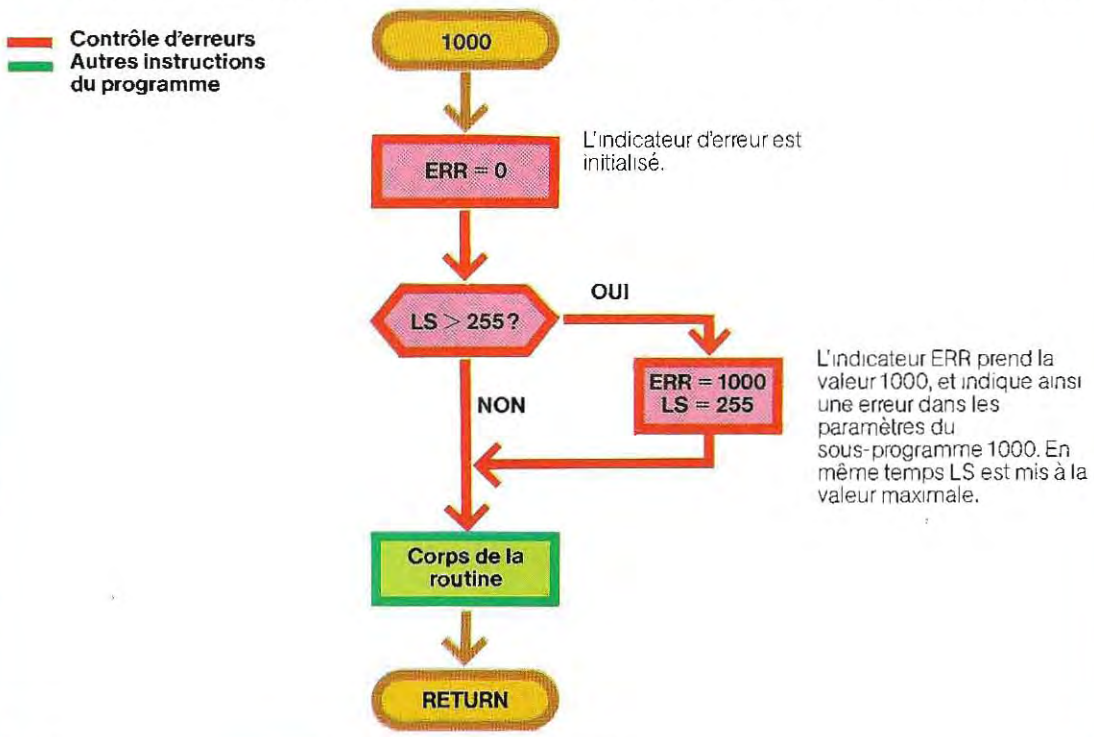
WHILE... WEND

La séquence des instructions WHILE... WEND active une autre forme de boucle : son exécution est conditionnée par la valeur (vrai ou faux) d'une expression donnée (résultat d'un calcul, valeur d'un indicateur, etc.). Une condition est « vraie » quand sa valeur est différente de zéro ; au contraire, elle est « fausse » quand sa valeur est zéro. La condition de vérité est souvent indiquée par le terme anglais true (vrai), signifiant non-zéro ; la condition opposée est false (faux) signifiant zéro.
La forme de cette instruction est :

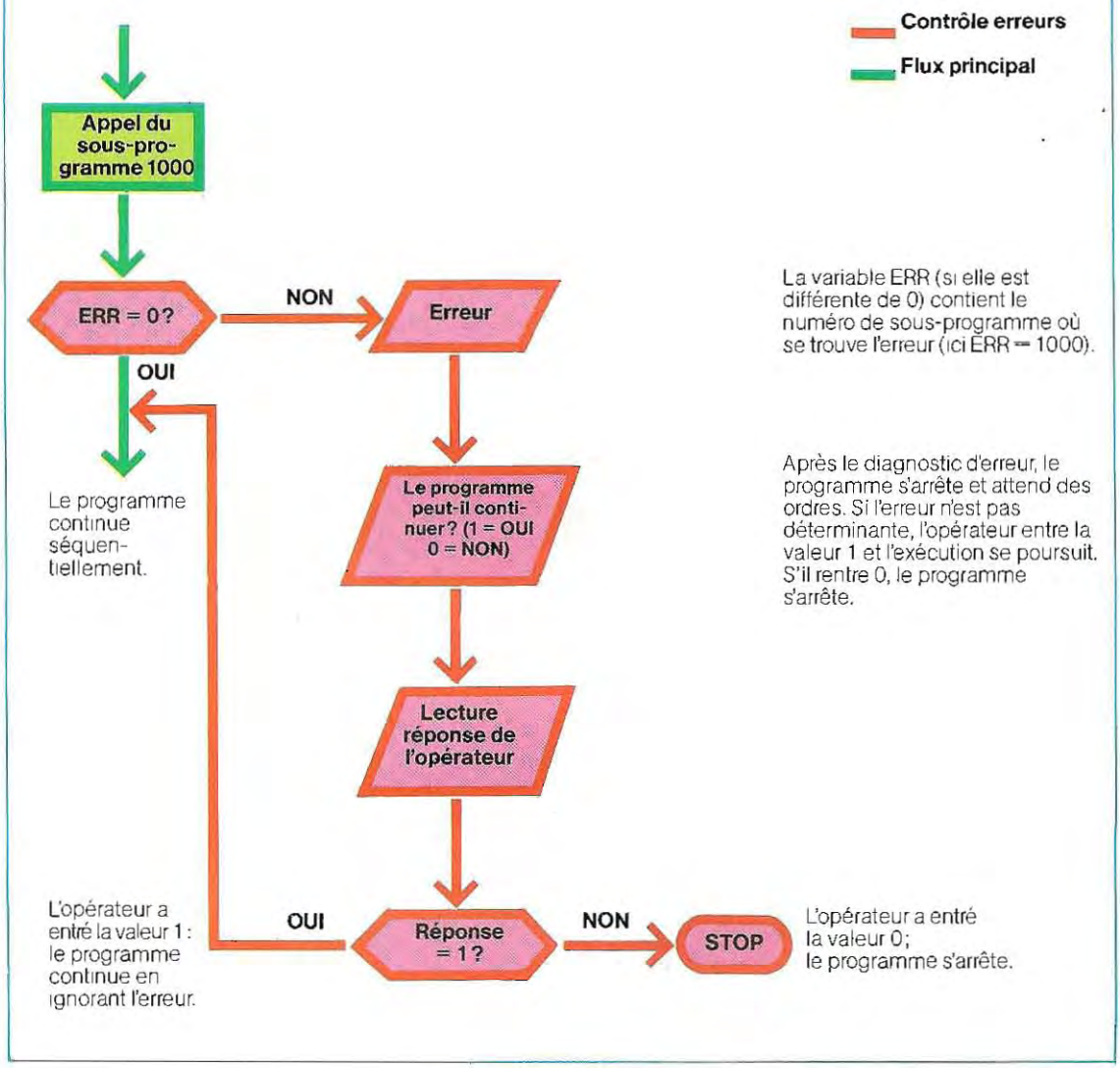
```
10  WHILE condition
(Partie itérée)
120 WEND
```

Le calcul (ou le paramètre) indiqué par le terme général de « condition » est évalué en début de boucle. S'il a une valeur différente de zéro, toutes les instructions comprises entre WHILE (ligne 10) et WEND (ligne 120) sont effectuées.
A ce point, l'exécution du programme remonte ensuite à l'instruction WHILE pour un nouveau test sur la valeur de « condition ». Si le résultat est différent de zéro, les instructions sont à nouveau exécutées. L'exécution du programme suit le chemin décrit jusqu'à ce que la condition soit fausse. A ce point, la boucle est abandonnée et le programme continue en séquence, à partir de la ligne suivant l'instruction WEND (while end).
Le schéma de la page 395 présente l'organigramme de la boucle, dans lequel la variable C représente le terme condition. La structure et le fonctionnement de cette boucle sont très différents de l'autre boucle FOR... NEXT. Sur cette dernière, les limites, et donc le nombre d'exécutions de la boucle, sont prédéterminées (bornes fixes de l'indice), tandis qu'avec

EXEMPLE D'UTILISATION DU SOUS-PROGRAMME DE GENERATION DE CHAINES



EXEMPLE D'APPEL DU SOUS-PROGRAMME 1000 AVEC CONTROLE DES ERREURS



l'instruction WHILE l'itération peut se poursuivre un nombre de fois déterminé, jusqu'à la modification de la valeur représentée par « condition ». Le traitement itératif doit naturellement contenir des instructions appropriées qui, à un moment donné, modifieront la valeur prise par la condition, et provoqueront ainsi la sortie de la boucle. Le schéma de la page 396 représente l'organigramme d'un exemple d'application du programme listé en bas de la page 395. Le programme doit lire les enregistrements d'un fichier de données contenant les noms et les adresses d'un répertoire. Il doit ensuite imprimer les valeurs lues, tant qu'il s'agit de la ville de Paris.

Dès que le nom de la ville n'est plus Paris, la boucle de lecture de données sera abandonnée, et l'exécution du programme se poursuivra séquentiellement.

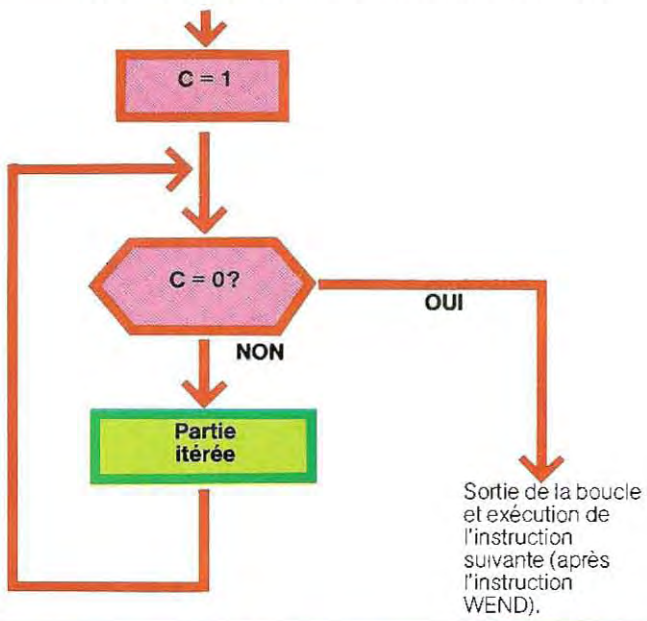
Pour simplifier le programme, on a omis, dans le listing de la page 395, toutes les instructions de gestion du disque mentionnées sur l'organigramme qui figure page 396, et les données sont entrées au clavier grâce à l'instruction 70 .

Instructions de saut

On désigne ainsi les instructions qui permettent de « sauter » d'un point à l'autre du pro-

gramme. Ce type d'instruction ne doit pas être confondu avec l'appel d'un sous-programme, même s'il lui ressemble.
Avec l'appel d'un sous-programme, le contrôle est transféré aux instructions de la routine, qui peuvent se trouver en un point quelconque du programme, mais RETURN fait reprendre l'exécution à la ligne qui suit celle qui contient l'appel. En fait, aucune instruction n'est « sautée ». L'exécution d'une partie du programme sera seulement reportée au retour de la routine. Par contre, dans les instructions de saut (ou de branchement), il n'existe pas de retour automatique. L'exécution du programme se poursuit séquentiellement depuis la ligne du saut en direction des

ORGANIGRAMME DE L'INSTRUCTION WHILE... WEND



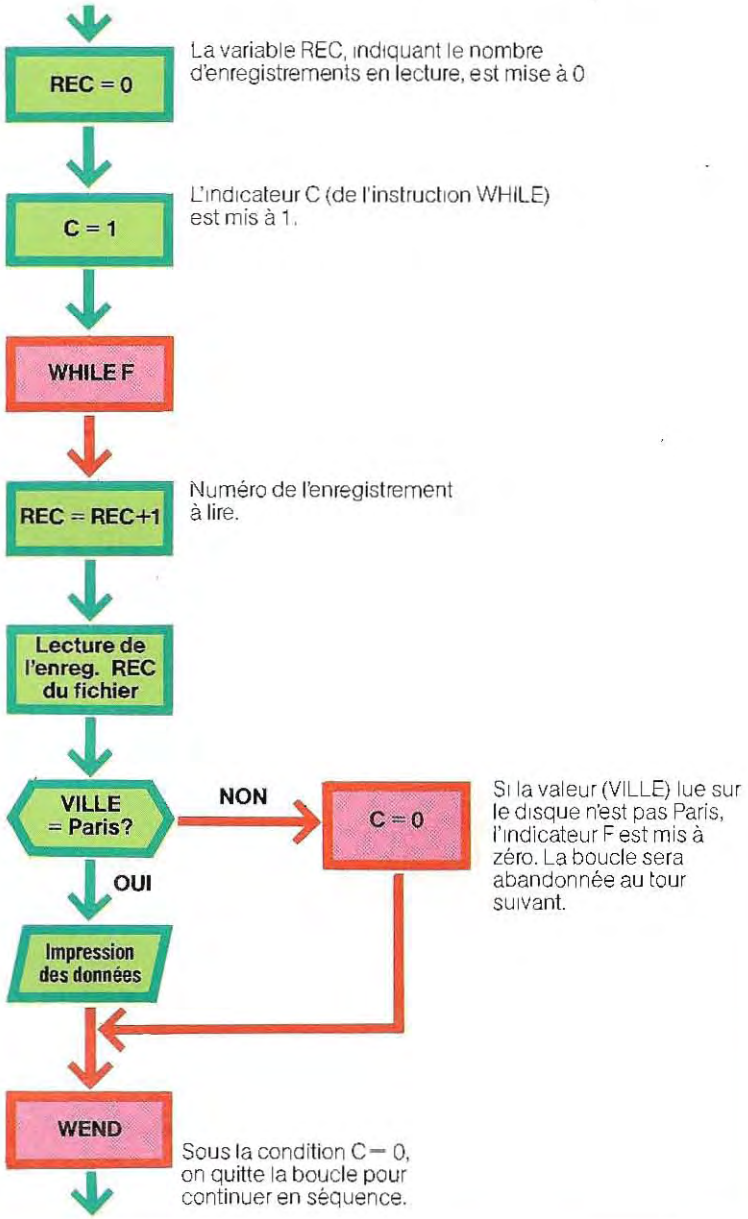
C = 1
WHILE F
Partie itérée
WEND

PROGRAMME EMPLOYANT LES INSTRUCTIONS WHILE... WEND

```
10  ' ** EXEMPLE DE BOUCLE WHILE-WEND **
20  '
30  ' FICHIER = TEST
40  '
50  C = 1           ' Initialisation de l'indicateur
60  WHILE C         ' Début de la boucle
70  INPUT "VILLE";VILLE$
80  LPRINT  "VILLE = " ;VILLE$
90  IF VILLE$ <> "PARIS" THEN LPRINT "CONDITION FAUSSE" :C=0
100  ' La condition de non-egalite arrete la boucle
110  IF VILLE$="PARIS" THEN LPRINT "CONDITION VRAIE"
120  WEND
130  LPRINT  "FIN DU PROGRAMME, VILLE = ";VILLE$, "CONDITION FAUSSE"
140  END
```

```
VILLE = PARIS CONDITION VRAIE
VILLE = PARIS CONDITION VRAIE
VILLE = PARIS CONDITION VRAIE
VILLE = LILLE CONDITION FAUSSE
FIN DU PROGRAMME, VILLE = LILLE          CONDITION FAUSSE
```

APPLICATION DE L'INSTRUCTION WHILE... WEND



numéros de ligne plus élevés, et les instructions du programme qui ont été sautées ne sont plus exécutées, à moins qu'elles ne soient expressément rappelées par le programmeur grâce à une autre instruction de branchement.

Le but principal de ces instructions de saut est de permettre des parcours différents à l'intérieur d'un même programme.

Le choix d'un parcours déterminé peut se faire en fonction du résultat de calculs particuliers ou en fournissant des valeurs clés à la machine durant la saisie des données.

GOTO...

Littéralement traduit par « allez en... », c'est l'instruction de saut la plus simple. La syntaxe est GOTO n, où n représente le numéro de la ligne à laquelle doit se poursuivre l'exécution du programme. Par exemple, GOTO 153 con-

duit le programme en ligne 153, et la partie comprise entre GOTO et l'instruction 153 ne sera pas exécutée. Dans l'utilisation de cette instruction, il est important de se souvenir de l'existence de la ligne vers laquelle on achemine l'exécution (il suffit d'écrire un simple mot). L'instruction GOTO est dite « de saut inconditionnel », car elle est toujours exécutée. Il existe également des instructions de saut « conditionnel », qui ne sont exécutées que sous certaines conditions.

ON... GOTO...

Cette nouvelle instruction de saut se comporte comme la précédente, avec, en plus, la possibilité d'adresser simultanément des lignes différentes, selon la valeur d'une variable. Une forme complète de l'instruction pourrait être, par exemple :

ON V GOTO 100, 250, 300, 1610, 2000

qui brancherait l'exécution selon la valeur de V à l'une des lignes explicitées.

Pour V = 1, on saute à l'instruction 100 ; pour V = 2, on saute à la ligne 250 et ainsi de suite, jusqu'à V = 5 qui provoque un saut à l'instruction 2000.

Si V est nul, ou supérieur au nombre de lignes d'arrivée figurant dans l'instruction (5 dans cet exemple), le programme continue avec l'instruction qui vient immédiatement après. Par exemple, dans l'instruction précédente, si V vaut 0 ou 6 (aucun saut n'étant demandé), l'exécution du programme se poursuit séquentiellement.

Les limites de validité du paramètre (V dans l'exemple) sont 0 et 255 ; des valeurs négatives ou supérieures à 255 génèrent une erreur, et le programme s'arrête.

La variable utilisée comme élément de sélection (V dans notre exemple) est toujours considérée comme entière.

L'instruction ON... GOTO... peut contenir aussi une expression à la place de la variable. En ce cas, l'évaluation de l'expression a d'abord lieu, puis le saut est exécuté en prenant le résultat comme indicateur. Par exemple, dans l'instruction :

ON K + 6 GOTO 10, 12, 21, 70

la somme K + 6 est d'abord évaluée, et ensuite, se fondant sur le résultat, le saut correspondant est exécuté (pour K + 6 = 1, on saute à 10, pour K + 6 = 2, on saute à 12, etc.).

L'exemple précédent nous indique une méthode pour utiliser des paramètres négatifs de l'indice. On ne peut obtenir le résultat K + 6 = 1 qu'avec K = − 5 (− 5 + 6 = 6 − 5 = 1). Si, à cette valeur négative, qu'on ne peut mentionner directement dans l'instruction, on ajoute une quantité opportune, on retrouve des paramètres positifs (de 1 à n). Par exemple, si l'on veut sélectionner un saut par un nombre compris entre − 5 et − 10, il faudra ajouter 11, obtenant ainsi un paramètre qui variera entre 1 (11 − 10) et 6 (11 − 5).

La solution qui apparaît la plus évidente, consistant simplement à inverser le signe, ne donne pas le même résultat.

En cas de variation de − 5 à − 10, un changement de signe produirait une valeur comprise entre 5 et 10, adressant depuis le cinquième numéro de ligne de GOTO jusqu'au dixième.

En revanche, en ajoutant 11 à notre variable, le résultat est compris entre 1 et 6, et permet l'adressage du premier numéro de ligne jusqu'au sixième. Les deux situations sont comparées ci-dessous :

- **Première méthode (addition d'une valeur positive)**
  ON K + 11 GOTO 120, 36, 100, 10, 21, 60
  Pour K variant entre − 10 et − 5, la quantité K + 11 varie de 1 (l'instruction 120 est exécutée) à 6 (l'instruction 60 est exécutée).

- **Seconde méthode (changement de signe)**
  L = − K

ON L GOTO ?, ?, ?, ?, 60, 21, 10, 100, 36, 120

La valeur de L est l'opposé de K (elle est donc positive). Pour K variant entre − 10 et − 5, le paramètre L varie de 10 à 5. Pour respecter la correspondance entre les valeurs de K et les lignes à exécuter, il faut insérer quatre numéros de ligne fictifs et inverser l'ordre des numéros valides par rapport au cas précédent. Les quatre premières lignes indiquées par le symbole ?, ne seront jamais adressées, étant donné le champ possible des valeurs prises par L (de 10 à 5).

La première méthode (addition d'une valeur positive) est plus fiable et présente des risques d'erreur moindres.

Netteté et simplicité restent cependant deux conditions essentielles à une bonne programmation. Il convient donc parfois d'ajouter quelques lignes de plus, et d'allonger ainsi le

Solutions du test 11

1 / a, d, f = réels en simple précision ; j = entier ; b,i = chaînes de caractères ; c = réel en double précision ; g = octal ; e,h = hexadécimaux.
Notons toutefois que, selon le Basic utilisé, certaines différences sont possibles (& peut marquer un nombre hexadécimal et &O un nombre octal, par ex.).

2 / Les expressions erronées sont :
b) D% et A% sont entières alors que le montant 46721 ne peut pas l'être (il dépasse 32767). Une erreur de débordement sera diagnostiquée par le système ;
c) D% est entière alors que C# est un réel en double précision et B! un réel en simple précision. Toutefois, le produit donne un résultat égal à 9 et donc, dans ce cas particulier, le calcul est quand même exécuté ;
e) D# est un réel alors que D$ est une chaîne. On ne peut effectuer aucune opération entre une chaîne (représentée en ASCII) et une variable numérique.
En revanche, l'expression H$ = "7" + D$ donne comme résultat H$ = "73". Pour les chaînes de caractères, le symbole + signifie concaténation, et il génère une chaîne formée de la succession des caractères des deux chaînes.

3 / La numérotation des tableaux part de 0. Aussi, sauf indication contraire, A(10) contient 11 éléments (de l'élément numéro 0 à l'élément 10).

4 / L'organigramme ci-dessous prévoit une série de calculs des valeurs NUM MOD d (pour les diviseurs d valant successivement 2, 3, 5, 7, 11). Chaque évaluation est suivie d'un test : si le résultat est nul, le nombre saisi (NUM) est divisible par d. On ne le testera pas avec les diviseurs suivants, qui, cependant, répondent peut-être à la question (si NUM = 165, par exemple, il est divisible par 3, mais également par 5 et par 11). Cette façon de dérouler le programme n'est donc donnée qu'à titre indicatif.

PROCEDURE DE CONTROLE DE DIVISIBILITE (PREMIERE VERSION)