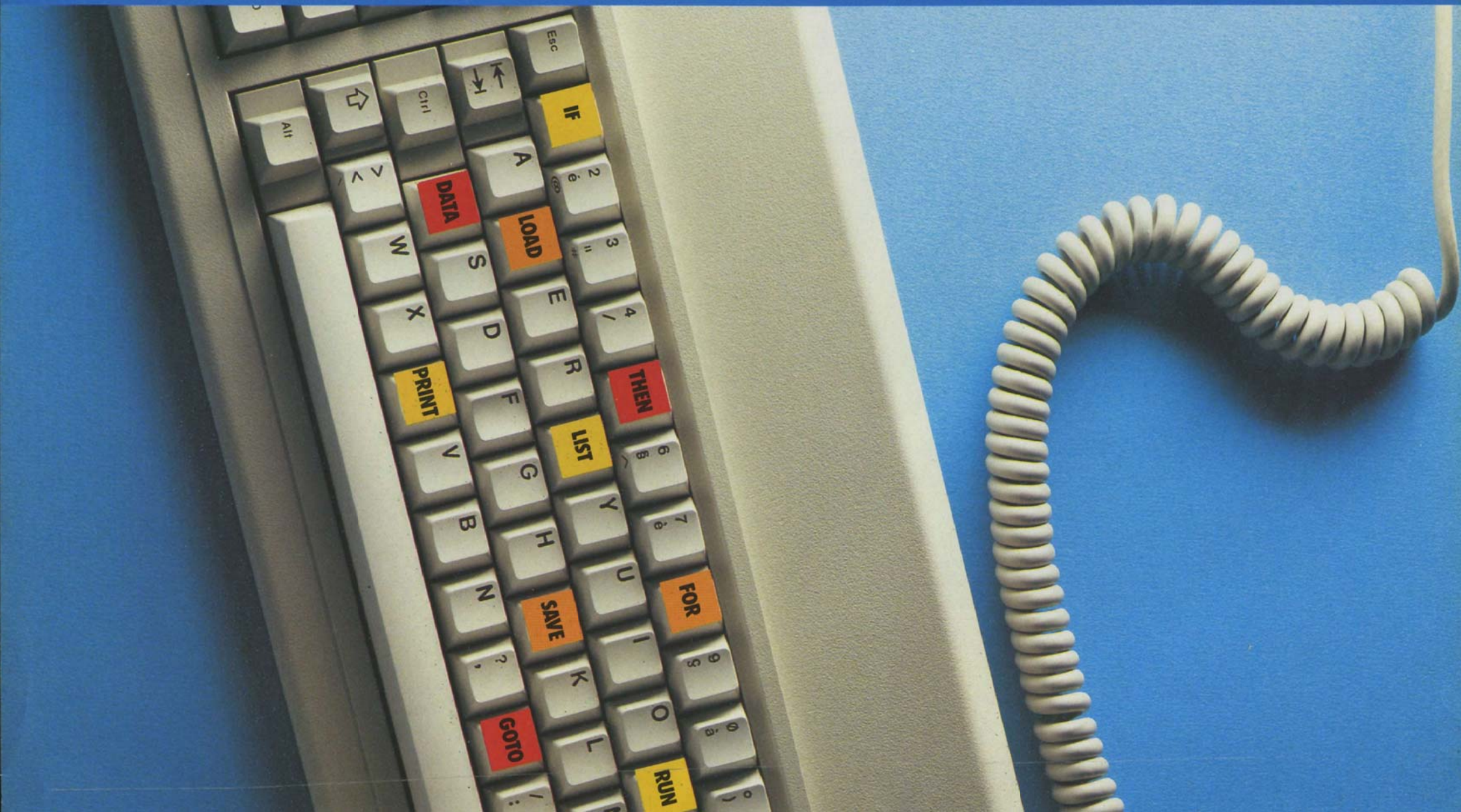
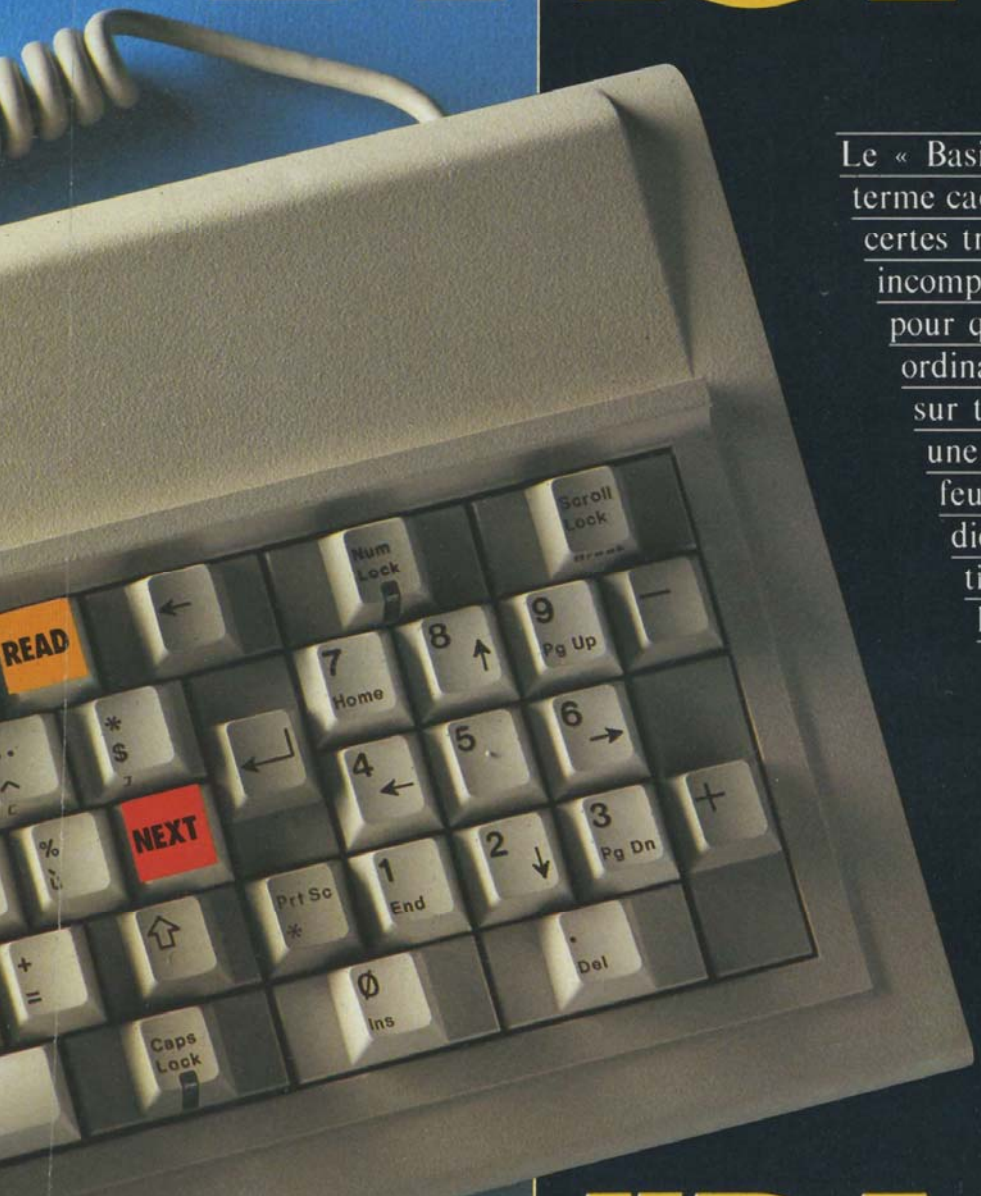
PROGRAMME DU MOIS



UN LOGICIEL DE CONVERSION ENTRE BASICS

BASIC'S



Le « Basic » n'a qu'un seul inconvénient : ce terme cache en fait une multitude de langages, certes très ressemblants, mais le plus souvent incompatibles entre eux... Il suffit d'un rien pour que le programme Basic de tel micro-ordinateur familial ne soit pas transposable sur tel autre. « Basic's » va vous apporter une aide très précieuse : pas de manuel à feuilleter ni de tableaux complexes à étudier ! Les correspondances entre instructions, d'un micro à l'autre, s'affichent à l'écran. Des menus permettent de découvrir les ordinateurs traités par le programme, d'effectuer des comparaisons, d'éditer sur imprimante les instructions correspondantes, etc. Le logiciel signale aussi les lacunes de tel ou tel Basic. Simplicité et rapidité d'utilisation en font l'outil modèle pour réconcilier tous les micros et... tous les Basic.

"BASIC'S"

POUR ADAPTER TOUS VOS PROGRAMMES

Un programme exclusif

UN PROGRAMME COMPATIBLE AVEC TOUS LES MICRO-ORDINATEURS

Les tableaux d'équivalences entre différents Basic permettent de transcrire les programmes d'un ordinateur à un autre. Mais ce genre de documents présente des inconvénients. Ils sont nécessairement volumineux, d'où une consultation fastidieuse. De plus, de telles tables de conversion, sur papier, n'ont qu'une seule entrée : la liste des instructions d'un Basic d'un système donné. Cela suppose donc que l'on ait déjà une idée des correspondances, si l'on désire retrouver une instruction d'un autre Basic.

De plus, une cellule dans un tableau, c'est peu lorsqu'il s'agit de définir une équivalence approchante. Quelquefois, plusieurs lignes de programmation sont nécessaires.

La solution la plus attractive consiste bien à recourir à un outil de conversion plus élaboré. C'est tout l'objet de « Basic's ». Il permet, par exemple, de traduire les instructions d'un programme pour Exelvision 100 en Basic pour Thomson MO5 ou Commodore 64. Un tel logiciel est plus complexe qu'il n'y paraît et mérite quelques commentaires.

Pas de Basic standard

La profession informatique se préoccupe régulièrement de définir des normes de standardisation, même si commercialement tous les constructeurs n'y trouvent pas toujours leur intérêt. Le recours au langage ADA au Département de la Défense aux Etats-Unis en est un bon exemple : il s'agissait de se mettre d'accord sur un seul langage, alors qu'il en existe plus de 400 différents, dont certains très spécifiques donc très coûteux.

Cette multiplicité est synonyme de dynamisme. Il n'est pas concevable, compte tenu des évolutions spectaculaires de la technologie des circuits intégrés, que l'on s'en tienne à des systèmes ou à des langages figés, même s'ils ont été érigés en standard.

Une autre question se pose. Peut-on affirmer que tel ou tel standard est bon ou mauvais ? Impossible de ne prendre en compte que les critères techniques. Un standard en soi est souhaitable parce qu'il permet le développement d'une large bibliothèque de logiciels, et donc la portabilité d'une machine à une autre. Il

```
LISTE DES MACHINES
------------------

1:EXL 100
2:MSX
3:TRS 80/VGS
4:APPLE II
5:AMSTRAD CPC
6:COMMODORE 64
7:MO 5
8:TO 7/7.70
9:CANON X-07
10:ORIC ATMOS
11:ZX SPECTRUM
12:SQUALE
13:IBM PC/COMPATIBLES
```

Fig. 1. – Les treize micro-ordinateurs pris en compte dans le programme.

reste qu'un système qui a réussi à s'imposer commercialement est rarement dépourvu de défauts.

En réalité, les standards bien établis sont rares en micro-informatique. On peut citer le codage ASCII des caractères (American Standard Code Information Interchange), ou les modes de communication IEEE 488 ou encore Série 232 C.

Ces standards ont pu être définis car ils reposent à la fois sur le matériel et sur le logiciel. La situation devient beaucoup plus floue lorssqu'il s'agit exclusivement de logiciel. Les systèmes d'exploitation CP/M ou MS-DOS ne sont pas des standards au sens strict mais le sont devenus *de facto*. Le PC-DOS est une bonne illustration de pseudo-standard. Car aucune norme n'a jamais été définie à son sujet, n'étant qu'une variante de MS-DOS. En clair, l'institution d'un standard tient essentiellement à sa diffusion commerciale.

Le langage Basic, bien que souvent dénigré par les programmeurs professionnels, s'est imposé en micro-informatique, du moins jusqu'ici. Car, les développeurs de logiciels, qui redoutent l'hermétisme de l'Assembleur, se tournent désormais vers Pascal et ses multiples variantes, ou le langage C.

Pour l'histoire, Basic a été créé aux Etats-Unis, le 1er mai 1964 par deux enseignants, John Kemeny et Thomas Kurtz, du Dartmouth College de Hanover, dans le New Hampshire. Ils destinaient ce B-A-S-I-C (Beginners All Purpose Symbolic Instruction Code) à l'apprentissage, en quelques heures, de la notion de programmation pour des applications de physique.

Comme toute langue, un langage informatique évolue. Entre la version originelle du Dartmouth College et celles que nous connaissons aujourd'hui, plus d'un dialecte a vu le jour.

Certes, de nombreux micro-ordinateurs emploient une forme à peu près standardisée : MBasic ou Basic Microsoft (du nom du célèbre éditeur). Or de multiples versions du MBasic ont été créées, pour répondre à des commandes de constructeurs (comme Thomson) ; on parle fréquemment des versions 2.0, 3.0 ou même 5.0. Il est aussi souvent question du Basic 80 : c'est toujours le MBasic développé par Microsoft mais sous CP/M. A destination des micro-ordinateurs professionnels (16 bits), Microsoft a également développé GBasic et GWBasic, qui offre des possibilités graphiques particulières.

Autour d'un noyau d'instructions existant dans le Basic de Dartmouth, des fonctions supplémentaires sont rajoutées, permettant souvent de mieux structurer les programmes ou de tirer parti des capacités graphiques ou sonores des nouvelles générations de micro-ordinateurs.

Un autre « standard » est souvent évoqué : le Basic ANSI, offrant des possibilités graphiques et que l'on rencontre, par exemple, sur les micro-ordinateurs Hewlett Packard.

On dénombre une bonne centaine de versions Basic dans le monde. Le paradoxe veut que, à l'inverse de ce qui se passe dans l'informatique « gros systè-

```
LISTE DES MACHINES
FONCTIONS CALCUL
LOG (  EXPRESSION NUMERIQUE  )

EXL 100:LOGARITHME DECIMAL
MSX:LOGARITHME NEPERIEN
TRS 80/VGS:LOGARITHME NEPERIEN
APPLE II:LOGARITHME NEPERIEN
AMSTRAD CPC:LOGARITHME NEPERIEN
COMMODORE 64:LOGARITHME NEPERIEN
MO 5:LOGARITHME NEPERIEN
TO 7/7.70:LOGARITHME NEPERIEN
CANON X-07:LOGARITHME NEPERIEN
ORIC ATMOS:LOGARITHME DECIMAL
SQUALE:LOGARITHME NEPERIEN
IBM PC/COMPATIBLES:LOGARITHME NEPERIEN
```

Fig. 2. – L'EXL 100 et l'Atmos se démarquent des autres : la fonction LOG (logarithme) correspond, chez eux, au logarithme « décimal ».

LES OUTILS DE CONVERSION D'UN BASIC A UN AUTRE SONT QUASIMENT INEXISTANTS

```
APPLE II:
IF CONDITION LOGIQUE THEN INSTRUCTIONS

AMSTRAD CPC:
IF CONDITION LOGIQUE THEN INSTRUCTIONS ELSE INSTRUCTIONS

COMMODORE 64:
IF CONDITION LOGIQUE THEN INSTRUCTIONS

MO 5:
IF CONDITION LOGIQUE THEN INSTRUCTIONS ELSE INSTRUCTIONS

TO 7/7.70:
IF CONDITION LOGIQUE THEN INSTRUCTIONS ELSE INSTRUCTIONS

ORIC ATMOS:
IF CONDITION LOGIQUE THEN INSTRUCTIONS ELSE INSTRUCTIONS

ZX SPECTRUM:
IF CONDITION LOGIQUE THEN INSTRUCTIONS
```

Fig. 3. – Le verdict du programme « Basic's », pour ce qui concerne les équivalences de l'instruction IF THEN ELSE : elles n'existent pas sur Apple, Commodore 64, ZX Spectrum.

mes », les outils de conversions d'un Basic à un autre sont quasiment inexistants.

Sur les ordinateurs de grande taille, il existe depuis bien longtemps des traducteurs de programme ou **compilateurs,** qui jouent le rôle d'intermédiaire entre le codage machine et le programmeur. Les compilateurs se sont très largement répandus car ils permettaient notamment de transférer un programme de gestion implanté en langage machine sur un ordinateur IBM vers un DEC : si ce programme était rédigé en Cobol, la totalité du programme source pouvait être transposée et tout matériel disposant d'un compilateur Cobol pouvait alors le récupérer.

En Basic, outre les compilateurs, se sont également développés les interpréteurs. L'interpréteur transforme l'instruction ou la ligne de programmation **au moment où le microprocesseur va l'exécuter.** Schématiquement, le processus est le suivant : lecture de la première ligne du programme, transformation en binaire, exécution, lecture de la deuxième ligne de programme... Cette procédure de traduction consiste à passer du programme « source » (écrit en langage évolué) au programme « objet » uniquement binaire. Inconvénient essentiel : la lenteur puisque par exemple dans une boucle, l'interpréteur doit traduire toutes les instructions lors de chaque passage.

Le compilateur scinde ces fonctions en deux parties distinctes, exécutées séparément. Une fois le programme saisi, la compilation est lancée. Le programme « source » devient « objet » et il est prêt à être exécuté. Les instructions, toutes préalablement transformées en binaire, seront exécutables sans modification.

Concrètement, l'interpréteur permet une mise au point du programme très aisée pour le débutant mais il n'existe pas de test sur la qualité des instructions utilisées. Par exemple, si, en Basic, on utilise GOTO 125 et que cette ligne 125 n'existe pas, le message d'erreur ne s'affichera qu'au moment du branchement à la ligne 125, à l'exécution du programme. Le compilateur décèlera cette erreur et l'indiquera avant. Il permet donc une mise au point générale, signalant en une fois toutes les erreurs de ce type.

Un programme compilé sera plus rapide à l'exécution et donc recommandé à un programmeur chevronné. L'interpréteur (permettant une mise au point aisée ou même un contrôle de syntaxe, ligne par ligne) convient mieux au débutant.

En fait, le système idéal serait de disposer d'une version interprétée pour la mise au point du programme et d'une version compilée pour l'exécution finale, et de pouvoir passer, sans interruption, de l'une à l'autre.

La famille des traducteurs

Avec la multiplication des microprocesseurs, sont apparus les **cross-compilateurs.** Leur rôle consiste à compiler un programme rédigé en langage évolué mais, au lieu de générer un programme destiné à l'ordinateur sur lequel il est utilisé, il génère un programme en langage machine qui sera exécutable sur un autre matériel. Ils permettent donc de récupérer les logiciels d'une machine donnée sur une autre. Ainsi, par exemple, le système d'exploitation du Macintosh a été développé sur Lisa. De même, on peut utiliser un TIPC de Texas Instruments pour créer des programmes destinés à un Exelvision... Ceci grâce à un Cross-Assembleur.

Mais les compilateurs ont un inconvénient majeur : ils sont longs à réaliser et donc coûteux à mettre au point. Il est exclu d'en réaliser un lors de chaque amélioration d'un langage. Pour cette raison sont nés les **préprocesseurs.** Leur rôle est, à partir d'un programme source utilisant la nouvelle version d'un langage évolué, de fonctionner conjointement avec le compilateur de l'ancienne version pour générer le code objet. Un préprocesseur est le plus souvent utilisé pour étendre les possibilités d'un langage (par exemple afin de disposer d'un Basic n'utilisant pas les numéros de ligne).

Le coût nécessaire au développement des compilateurs est devenu dissuasif, car les informaticiens doivent tout « réapprendre » lorsqu'un nouveau langage se présente...

Ainsi, ADA qui est à l'heure actuelle le plus moderne des langages évolués (le plus puissant, mais aussi le plus complexe) n'a disposé de compilateurs que six ans après sa définition !

Seule solution pour les créateurs d'ADA et ceux qui veulent rentabiliser son coût de conception : les **vérificateurs.** Ces logiciels simulent un compilateur. Ils ne génèrent pas de code exécutable, mais analysent un programme source afin de savoir s'il est « compilable », s'il contient des erreurs de logique. Un vérificateur permet aux informaticiens de préparer des logiciels sources et de s'habituer au langage concerné en attendant la disponibilité effective du compilateur. Le plus connu est LINT s'exécutant sous le système d'exploitation Unix : il permet de traiter un programme source rédigé en langage C.

Enfin, ultime merveille de créativité informatique, les **compilateurs de compilateurs.** Ils ont été créés afin de diminuer le temps de développement des compilateurs. Leur rôle : à partir de la syntaxe du langage évolué, ils génèrent... un compilateur de ce langage. Seul problème des compilateurs de compilateurs : on n'a pas encore inventé le « compilateur de compilateurs de compilateurs »... De plus, ils sont limités aux langages d'une structure particulière. On peut inclure dans cette famille les logiciels « générateurs d'application » comme « The Last One », qui, à partir d'une description des tâches désirées, génèrent le programme correspondant dans un langage donné. Leur rôle se limite souvent à la gestion de fichiers ou de données numériques. De là à concevoir un générateur de programmes d'intelligence artificielle, il y a un énorme pas à franchir.

Le programme

Le programme que nous vous proposons ce mois-ci ne réalise pas de traduction automatique, au sens strict. Toutefois, la structure de données qu'il met en œuvre pourrait être récupérée pour en faire un traducteur automatique ; il prendrait par exemple un programme Basic sur EXL 100 et le transformerait en une version pour Thomson MO5. Mais cette transformation prendrait un certain temps, certainement supérieur à celui nécessaire pour se familiariser avec les Basic des micro-ordinateurs concernés. Ce traducteur, fort volumineux, serait capable de transposer tous les programmes n'utilisant pas les ordres PEEK et POKE, ni des instructions trop spécifiques à la machine. Comme nous l'avons vu, Basic's permet d'établir les correspondances entre les instructions Basic des micro-ordinateurs les plus répandus.

Un premier menu demande de choisir un système parmi treize : celui pour lequel le listing du programme est publié par exemple dans *Soft & Micro* mais élaboré pour un micro différent du vôtre... Le programme affiche par catégories la liste des instructions contenues dans son dictionnaire : contrôle du clavier, de l'écran, chaînes de caractères, etc. Lorsqu'une instruction a été choisie, le programme affiche les correspondances dans les autres Basic.

Basic's offre l'avantage d'une autre utilisation : il permet, en effet, d'afficher le contenu de tout son dictionnnaire d'instructions (tous Basic confondus), et, pour chacune d'elle, peut éditer la liste des micro-ordinateurs qui en disposent. D'autre part, il explique, en deux ou trois mots, le rôle des instructions.

Ce sera parfois l'occasion de découvrir des particularités que vous ignoriez. Ainsi, comment savoir, *a priori*, si « A$=KEY$ », qu'utilisent l'EXL 100 et l'Oric Atmos, aura le même effet sur les deux systèmes ? Notons d'ailleurs que sur l'EXL 100, cela correspond à l'attente d'une touche, tandis que sur l'Atmos, cela signifie « touche saisie au vol » (sans attente). Ce genre de constatation vous évitera de graves erreurs. En ce sens Basic's peut se substituer aux notices des constructeurs.

Encadré 1

UN EXEMPLE DE TRANSPOSITION DIFFICILE

La transposition de trois lignes Basic peut arrêter bien des « spécialistes ». L'exemple qui suit écrit un « 0 » en rouge sur trois micro-ordinateurs différents à peu près au milieu de l'écran. Cherchez les points communs.

Commodore 64
```
10 POKE 1444,79
20 POKE 55716,2
```
EXL 100
```
10 CALL COLOR(« 0R »)
20 LOCATE (10,20)
30 PRINT « O »
```
MO 5
```
10 COLOR 1
20 LOCATE 20,10
30 PRINT « O »
```

Quelques options utiles ont été ajoutées aux fonctions de traduction. Ainsi, le logiciel peut éditer ses résultats sur une imprimante. Il est également possible, pour savoir si la transposition sera aisée ou non, de consulter des tableaux récapitulatifs afin de savoir quelles instructions sont disponibles sur quels micro-ordinateurs. Cette vue d'ensemble est essentielle avant d'entamer tout travail de transposition, car elle détermine si la tâche est démesurée ou non : dans certains cas, le temps de traduction peut dépasser celui de développement du programme lui-même. Alors, autant reconcevoir entièrement le programme !

Basic's a, bien sûr, ses limites. Elles sont dues à une prolifération des multiples Basic. Certes, on a coutume de se référer au « modèle » Microsoft, qui se dégage de l'ensemble (suivis par le MSX, le TRS 80, puis l'Apple II, l'IBM PC, le MO5...). Mais au sein même de ce « standard » existent des variantes : le MO5 ne dispose pas de la fonction ATN ; l'Apple II est dépourvu du « IF THEN ELSE », etc.

Les points critiques pour la conversion d'un Basic à un autre concernent le plus souvent les instructions qualifiées de « super-puissantes » par les constructeurs. Ainsi, dès que l'on aborde le domaine du graphisme sur écran, chaque Basic se distingue de ses concurrents. Une simple commande de tracé de points peut prendre une cinquantaine de formes différentes. La syntaxe suffit aussi parfois à rendre l'adaptation très délicate : les fonctions de travail sur les chaînes de caractères en sont l'exemple parfait. Elles sont plus ou moins nombreuses, plus ou moins différentes suivant le système utilisé. Ainsi, pour obtenir un sous-ensemble de la chaîne, certains Basic n'utilisent pas MID$, mais une forme équivalente et pourtant totalement différente d'aspect (c'est le cas des micro-ordinateurs Sinclair ou Hewlett Packard). Certains acceptent des paramètres nuls, d'autres provoquent une erreur.

Les instructions plus simples comme IF THEN ne sont pas forcément mieux loties : IF THEN ELSE n'est pas toujours disponible, certains Basic n'acceptent derrière THEN qu'un numéro de ligne, d'autres nécessitent soit GOTO soit LET pour effectuer un calcul... Le standard « de fait » qui aurait dû s'imposer même à ce niveau d'instruction part en fumée.

Le plus complexe demeure tout de même l'adressage des périphériques (imprimante, écran, disquettes, RS 232...). Il est impossible – ou presque – de trouver deux matériels disposant des mêmes périphériques, ou des mêmes commandes. C'est donc un travail considérable qui attend l'amateur de transposition lorsque le programme original utilise des particularités un peu trop spécifiques d'une machine. **S&M**

J.-L. Luczak F. Pierot