Popularisé par l'explosion de la micro-informatique, le langage Basic reste le premier amour de tous ceux qui ont découvert l'informatique grâce aux micro-ordinateurs. Considéré depuis toujours avec un brin de condescendance par les programmeurs professionnels, le Basic a su évoluer pour devenir un langage à part entière, c'est-à-dire capable de développer des programmes compétitifs. Découvrir les récentes évolutions du Basic, c'est comprendre les concepts fondamentaux de la programmation. C'est en celà justement que le Basic est précieux, tel un témoin de l'évolution de la micro-informatique.

BASIC, MON AMOUR

CRÉÉ EN 1964, LE BASIC EST DONC un vieux langage. Il n'est pas sans intérêt de remonter jusqu'à cette époque, puisque certaines caractéristiques actuelles du langage y trouvent leur explication. Basic signifie Beginners All Purpose Symbolic Instruction Code, c'est-à-dire langage à tout faire pour les débutants. Un langage pédagogique par conséquent, dont l'unique prétention était de permettre une première expérience de programmeur. Son gros atout par rapport aux langages traditionnels est d'être un langage interprété. Ce terme technique signifie simplement que les ordres d'un programme écrit en Basic sont, les uns après les autres, traduits en langage machine puis immédiatement exécutés. Cette technique a l'avantage de la facilité : pour lancer un programme, il suffit de taper la commande RUN pour obtenir aussitôt l'exécution du premier ordre. Un langage interprété possède une deuxième qualité essentielle pour des débutants, à savoir exécuter un ordre en mode immédiat. Par exemple, pour additionner 2 et 2, il suffit d'écrire : PRINT 2+2. Et l'ordinateur d'imprimer 4, le résultat de l'opération. Depuis la banalisation de la calculette, cette méthode paraît plutôt lourde, mais souvenons-nous, avant le début des années 70, il n'y avait pas de calculette... L'avantage de ce mode d'exécution immédiat est surtout pédagogique : tant qu'on n'est pas très sûr du

Découvrez ou redécouvrez le Basic à travers ce plaidoyer pour le plus controversé et le plus plébiscité des langages de programmation

fonctionnement d'un ordre en Basic, il est possible de l'essayer. La différence entre un ordre à exécuter immédiatement et un autre dont l'exécution sera lancée par l'ordre RUN est faite au moyen d'une convention : les ordres destinés à être insérés dans un programme doivent être précédés d'un numéro de ligne, les ordres interprétés et exécutés immédiatement sont au contraire écrits sans numéro de ligne. Cette facilité de mise en œuvre est encore améliorée par l'existence d'un éditeur plein écran grâce auquel on se déplace sans contrainte dans les ordres affichés pour y faire des corrections. Pourtant une telle aisance comporte un risque pour un programmeur débutant : pouvoir modifier sans limite son programme et le tester ligne à ligne incite à entrer dans un cycle d'essais successifs sans fin, essais qui font perdre de vue le but premier.

La deuxième qualité du Basic est son extrême simplicité : il y avait moins d'une dizaine d'ordres dans le langage d'origine. Il faut bien reconnaître que cette pauvreté le rendait incapable d'aborder des programmes complexes, mais ce n'était pas là le but. Depuis, il s'est considérablement enrichi, et les Basic les plus récents sont aussi riches, sinon plus, que les grands langages classiques de programmation, comme Cobol, Fortran ou Pascal.

Ces différents enrichissements se sont faits progressivement, et les puristes de la théorie des langages peuvent reprocher aux récents Basic un caractère fourre-tout assez prononcé. Mais il reste au Basic une qualité essentielle : il permet, en se servant de quelques ordres seulement, de faire ses débuts dans l'art difficile de la programmation.

Peu gourmand

Un dernier avantage du Basic était sa faible taille en mémoire. A l'heure où la norme pour les ordinateurs tourne autour de 1 Mo de mémoire, il est amusant de se souvenir par exemple que l'un des tout premiers micro-ordinateurs disponibles en France, le MCB Alcyane, se vendait début 1978 avec 12 Ko de mémoire et qu'il était livré avec deux versions de Basic, l'une occupant 2 Ko et l'autre un peu plus de 10 Ko. Basic était alors un langage simple, facile à utiliser et peu gourmand en mémoire. Toutes ces caractéristiques ont fait qu'au moment de l'apparition de la micro-informatique, la totalité des constructeurs ont tout naturellement proposé leurs machines avec le langage Basic.

Malheureusement, chacun a fait son propre Basic et il n'y a pas de norme à proprement parler. La situation est à peu près claire dans le domaine des ordinateurs professionnels compatibles avec l'IBM PC, où le Basic de Microsoft s'est imposé comme un standard de fait. Ce n'est pas le cas pour les autres machines où chaque constructeur se donne la peine de livrer un Basic qui fasse bien ressortir les particularités de sa machine : chaque Basic est différent des autres. A cause de sa réputation de simplicité et de son absence de normalisation, le Basic garde encore maintenant une réputation de petit langage que les professionnels ne sauraient utiliser sans déchoir. Pourtant, cette opinion, fondée il y a quelques années devient fausse à mesure que le Basic s'enrichit. Nous allons voir quelles en sont les principales améliorations récentes et pourquoi il peut rivaliser maintenant avec les meilleurs langages.

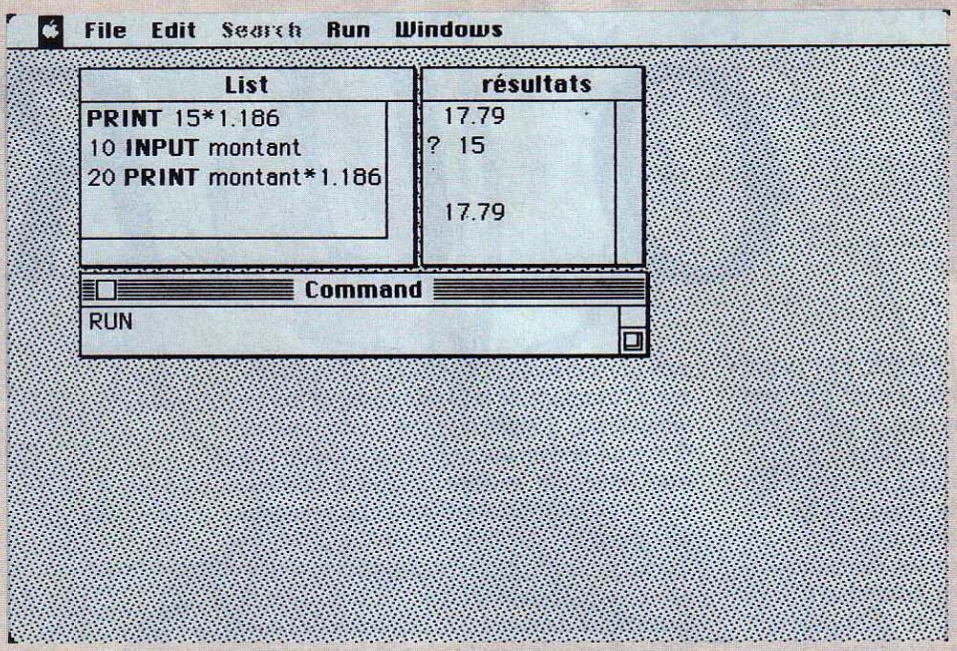
FOR ou WHILE

L'amateur dont le plus long programme ne dépasse pas une centaine de lignes, ne comprend généralement pas ce qui fait la différence entre son art et le métier d'un programmeur professionnel.

La première chose à comprendre est que ce dernier travaille sur un problème qui n'est pas le sien, et dont il n'a pas de connaissance intuitive. En pratique, l'écriture du programme est précédée par une phase d'analyse, pendant laquelle l'analyste décide de qui sera informatisé et de la façon dont ça le sera. Le programmeur, qui se trouve en bout de chaîne dans le processus, travaille donc à partir d'un document écrit, sans contact avec la réalité qu'il informatise. Pour des raisons évidentes d'efficacité, le problème est divisé en multiples sous-problèmes. Il est donc important qu'il puisse disposer lui aussi, dans le langage lui-même, d'ordres lui permettant de structurer sa pensée. En acquérant cette technique, un programmeur amateur gagnera beaucoup de temps dans la réalisation de son programme : il pensera donc avant tout à analyser son problème pour en détacher plusieurs parties distinctes pouvant faire l'objet de mises au point séparées.

Dans les premiers Basic, les outils de structuration étaient pratiquement inexistants. Le seul outil était la boucle FOR, servant à répéter un groupe d'ordres plusieurs fois jusqu'au prochain ordre NEXT, tout en faisant varier la valeur d'une variable de contrôle. Ainsi si l'on veut dans une chaîne de caractères, remplacer toutes les lettres majuscules par des mi-

LA TVA EN EXÉCUTION IMMÉDIATE OU EN PROGRAMME



UNE CARACTÉRISTIQUE DU BASIC EST qu'il existe 2 modes de fonctionnement du langage. Le premier est le mode immédiat dans lequel l'ordre est exécuté dès la fin de la frappe. Ainsi, pour un calcul de TVA, on écrit :
PRINT 15*1.186
et l'ordinateur imprime immédiatement la réponse : 17.79.

On se sert ici de l'ordinateur comme d'une calculette, et il faut retaper le taux de TVA pour chaque nouveau calcul. Au contraire, pour faire la même chose dans un mode de programmation, on écrirait :
10 INPUT MONTANT
20 PRINT MONTANT*1.186

Les deux ordres numérotés sont conservés dans la mémoire de l'ordinateur et forment le programme. Pour le lancer, il suffit de taper la commande RUN.

L'interpréteur Basic exécute le premier ordre, INPUT, qui est un ordre de saisie au clavier. La machine attend la saisie du montant, et quand c'est fait, passe à l'ordre suivant, le calcul et l'impression du résultat.

La différence entre les deux solutions est que l'on peut faire fonctionner plusieurs fois le programme sans avoir à retaper le taux de TVA.

C'est toute la différence entre une calculette et un ordinateur.

nuscules, on pourra par exemple commencer à mettre au point les quelques ordres nécessaire à l'opération sur un seul caractère, puis entourer ce petit programme d'une boucle qui réalisera cette opération pour tous les caractères de la chaîne. Ce qui compte dans cet exemple, c'est moins le programme lui-même que le mécanisme mental : pour faire une opération sur plusieurs caractères, on met d'abord au point sur un seul, puis on teste la petite séquence d'ordre, l'entourant ensuite du couple FOR NEXT.

Répéter une séquence d'ordres sous le contrôle d'une variable n'est pas la seule structure possible. La plupart des Basic ont maintenant l'ordre WHILE, en français « tant que », qui permet de répéter une série d'ordres tant qu'une condition logique est satisfaite. Ainsi, le mathématicien qui calcule la valeur d'une fonction par approximation fera fonctionner son programme tant que la précision désirée n'est pas obtenue. L'expression logique qui conditionne la sortie de la boucle peut être complexe. Un programme de vérification d'une zone fonctionne tant que le caractère RETOUR n'est pas rencontré et tant que la taille maximum de la zone n'est pas dépassée. Les deux ordres FOR et WHILE jouent des rôles similaires. On peut d'ailleurs regretter que le Basic n'adopte pas la solution qui a été retenue dans certains autres langages, comme le PL1, où les deux ordres ont été fusionnés. Das ce cas, on pourrait répéter un paquet d'ordres en itérant une variable, tout en conditionnant la sortie de la boucle par une expression logique. Le programmeur débutant aura intérêt à user systématiquement de l'ordre WHILE qui amène à une meilleure structuration, puisqu'il oblige à se poser la question des conditions de sortie de boucle.

La construction d'outils

Si le choix des ordres de structuration est affaire de style personnel, l'emploi de sous-programmes est, lui, une affaire d'analyse et d'organisation du travail. Revenons à notre programmeur professionnel. Sauf dans le cas de tous petits projets, il ne travaille pas seul. Pour des raisons de délais, plusieurs programmeurs collaborent et doivent donc pouvoir éventuellement finir un programme commencé par un autre. De plus, surtout pour des programmes de gestion, on s'aperçoit bien vite que l'on fait « toujours la même chose ». On peut distinguer trois actions principales : la vérification des données, le traitement et la mise en fichiers, puis l'impression.

Chaque programmeur astucieux, chaque société de services se constitue au fil des ans une bibliothèque de petits outils, dont l'emploi est généralisé. Il n'y a pas cinquante manières de vérifier qu'une date saisie à l'écran est valide, et il est parfaitement inutile de reprogrammer à chaque fois cette opération, moins triviale qu'il n'y paraît. La manière la plus pratique de donner une forme définitive à ces outils de base est de les organiser en sous-programmes. Rien de plus simple en apparence, il suffit d'utiliser le couple GOSUB - RETURN, qui sont les ordres d'appel et de fin de sous-programmes présents dans tous les Basic. Malheureusement, cette structure primaire est parfaitement impropre à la construction de bibliothèques de sous-programmes. En effet, les variables peuvent être modifiées en dehors de l'utilisation d'un sous-programme. Il convient donc de fixer conventionnellement et à l'avance les noms de variables du programme principal et ceux qui ne serviront que dans les sous-programmes. Cela oblige, pour mettre au point des outils communs entre programmeurs, une concertation précise. Ce n'est pas du tout la même chose de dire à un programmeur : « Fais un sous-programme qui vérifie une date », que d'être obligé de lui préciser à l'avance le nom de la variable autorisée et surtout le nom des variables interdites.

Dans la pratique, on se rend compte avec les ordres classiques GOSUB et RETURN qu'il est quasiment impossible d'intégrer plus de quelques sous-programmes dans le même programme, et que même dans ce cas, il arrivera un jour qu'une variable serve par erreur dans deux sous-programmes différents. Ce jour-là, une séquence mille fois utilisée refusera de fonctionner normalement, et il faudra beaucoup de temps pour identifier la variable fautive. Ce fonctionnement limité du couple d'ordres GOSUB-RETURN est le principal obstacle à l'utilisation professionnelle des anciennes versions du Basic. Pour un non-professionnel, la manipulation des faux sous-programmes GOSUB et RETURN est encore plus hasardeuse, car travaillant moins régulièrement sur son programme, il oublira fatalement les noms des variables.

Une situation en évolution

A moins de s'astreindre à une discipline incroyablement contraignante, il est donc impossible de mettre au point un programme Basic dépassant quelques centaines de lignes et surtout illusoire de vouloir en reprendre des parties pour les améliorer ou les intégrer

LA FACTORIELLE ENTRE UN FAUX ET UN VRAI SOUS-PROGRAMME

List

```
10 FOR I=7 TO 10: GOSUB 100: PRINT I;R;:NEXT I STOP
100 REM Début du sous-programme
110 R=1:FOR J=1 TO I:R=R*J:NEXT J
120 RETURN
```

List

```
FOR I=7 TO 10: CALL FACT(I,R): PRINT I;R;:NEXT I:STOP
SUB FACT(N,F) STATIC
F=1: FOR I=1 TO N : F=F*I : NEXT i
END SUB
```

résultats

```
 7   5040   8   40320   9   362880   10   3628800
```

LES DEUX PROGRAMMES FONT EXACTEMENT la même chose : ils calculent et affichent les factorielles de 7 à 10.

Dans le premier cas, on utilise les ordres GOSUB et RETURN. Comme les valeurs des variables peuvent être modifiées aussi bien dans le programme principal que dans le sous-programme, il faut faire extrêmement attention pour ne pas employer deux fois la même variable.

Dans le deuxième cas, on utilise les ordres CALL et SUB, qui définissent un vrai sous-programme. Les valeurs des variables sont indépendantes, ainsi peut-on écrire sans problème deux variables I distinctes. A l'inverse, il n'est pas nécessaire que les noms des variables servant d'argument soient les mêmes au moment de l'appel et dans la déclaration du sous-programme. Celui-ci ayant un nom, les numéros de ligne deviennent inutiles.

AVEC INKEY ET WHILE, LE PROGRAMME PERD PATIENCE

List

```
REM début de programme
WHILE A$="" AND T<1000
A$=INKEY$:T=T+1
WEND
REM suite du programme
PRINT A$,T
```

L'ORDRE INKEY TRANSMET LE DERNIER caractère tapé au clavier. Sa grande différence avec le classique INPUT, c'est qu'il n'arrête pas le déroulement du programme. En combinaison avec l'ordre WHILE, on peut construire toutes sortes de programmes de contrôle du clavier. Ici par exemple, le programme attend environ 10 s que l'utilisateur se serve du clavier, puis continue son déroulement. On obtient cette temporisation en tournant dans la boucle 1 000 fois, à moins qu'un caractère ne soit saisi, auquel cas on sort immédiatement. Ce type de dispositif peut servir dans les programmes de jeux par exemple, où l'ordinateur peut continuer sa recherche tant que le joueur n'intervient pas sur la machine. On remarque de plus que ce petit programme ne contient pas d'ordre de branchement GO TO, ce qui est la marque d'une bonne utilisation des ordres de structuration.

dans un autre programme. Depuis quelques mois, la situation est en train de changer. Les deux derniers Basic de Microsoft, le Quick Basic pour IBM PC et le Basic 2.0 pour Macintosh, proposent l'un et l'autre une vraie structure de sous-programmes. A la différence de l'ordre GOSUB qui fonctionne avec un numéro de ligne, l'ordre CALL permet d'appeler un sous-programme par son nom. Le début est indiqué par un ordre SUB contenant le nom du sous-programme et la liste des variables, et la fin est indiquée par l'ordre END SUB. Le fonctionnement apparent est exactement le même dans les deux cas mais il existe pourtant une différence capitale. Dans le cas d'un vrai sous-programme, la portée des variables est limitée à l'intérieur même de ce sous-programme. Ainsi si on réutilise le même nom de variable dans deux sous-programmes différents, il ne se passe rien de fâcheux car le Basic considère que ce sont deux variables différentes.

Le seul moyen d'introduire et de sortir des valeurs est de passer par les variables déclarées au début. Pour reprendre notre exemple, l'analyste pourra demander au programmeur de faire un sous-programme nommé VERIFDATE avec deux arguments. Une variable chaine de caractères contiendra la date à vérifier et une variable numérique indiquera à la sortie si la date est valide ou non.

Avec ces seules indications, notre programmeur peut se mettre au travail, sans se préoccuper du nom des variables figurant par ailleurs dans les autres parties du programme. Qui plus est, les autres programmeurs travaillant avec lui pourront considérer le problème des vérifications de date comme réglé, et se mettre à programmer en écrivant des appels au sous-programme VERIFDATE, comme s'il existait déjà. Pour l'amateur isolé, le sous-programme constitue la fin de presque tous ses problèmes de programmation. Face à un problème donné, il lui suffira de le séparer en plusieurs parties qu'il nommera et pour lesquelles il décidera à l'avance quelles sont les informations échangées. Passé cette étape, il écrira le squelette de son programme comme une suite d'ordres CALL. Ensuite, il recommencera avec chacun des sous-programmes.

On reconnaîtra le bon programmeur au respect des quelques règles suivantes : chaque sous-programme commencera par des lignes de remarque indiquant son mode d'emploi et la signification des variables ; un programme principal ou un sous-programme ne devraient jamais dépasser quelques dizaines de lignes, l'idéal étant une vingtaine, la taille d'un écran.

En s'efforçant de programmer avec ces règles simples, on aura la surprise de constater que la quasi-totalité de la programmation courante en gestion se résume à une centaine de sous-programmes d'une vingtaine de lignes chacun. Comme tout bon ouvrier, le programmeur se constituera peu à peu sa boite à outils. Il y gagnera en rapidité, et pourra de plus, changer pratiquement sans problème de langage, l'appel de sous-programmes étant justement un ordre commun à tous les langages. Il est probable qu'au moment où sa boite à outils sera bien au point, notre programmeur sera déjà promu analyste-programmeur.

Interruption

Dans les anciennes versions du Basic, le seul type d'interruption était le cas d'une erreur de fonctionnement du programme découverte en cours d'exécution. Partant du principe, un peu pessimiste, que cette erreur peut survenir à n'importe quel moment, le Basic la considère comme un événement extérieur venant perturber le déroulement normal du programme ; celui-ci est alors interrompu par un ordre spécial, ON ERROR, un sous-programme essayant de gérer l'erreur au mieux. Le traitement de base d'une telle erreur consiste à afficher le type d'erreur et le numéro de ligne où elle s'est produite. Là encore, ce n'est pas le traitement qui compte, mais l'idée originale. Quand on regarde un ordinateur moderne, on se rend compte en effet que, de plus en plus, des éléments extérieurs peuvent venir à tout moment agir sur le déroulement du programme. On en vient à généraliser le principe de ces interruptions. Avec le Basic 2.0 de Microsoft pour le Macintosh, il n'existe pas moins de cinq événe-

PROGRAMMER PAR INTERRUPTION

List

```
REM début de programme
ON TIMER (10) GOSUB Efface
TIMER ON
FOR i=1 TO 1000 :PRINT i:NEXT
STOP
REM Traitement de l'interruption
Efface:CLS:RETURN
```

LA GÉNÉRALISATION DE LA NOTION D'interruption modifie complètement la manière de programmer. Dans ce petit programme, l'écran est effacé toutes les 10 secondes. L'option ON TIMER interrompt en effet le programme toutes les 10 secondes et le branche vers le sous-programme Efface. Comme tout bon chronomètre, il peut être déclenché par l'ordre TIMER ON, suspendu par TIMER STOP et remis à zéro par TIMER OFF.

ments extérieurs qui peuvent venir interrompre le cours normal du programme : l'option ON MENU est activée quand un élément de menu est sélectionné ; ON MOUSE permet de détecter la pression sur le bouton de la souris ; ON DIALOG prend en compte une action dans une des fenêtres de l'écran ; ON TIMER est activé quand un temps donné est écoulé ; enfin l'option ON BREAK gère les tentatives d'arrêt du programme en cours.

Avec ces diverses options, il devient possible de changer complètement l'idée de base de la programmation. On considère que la machine ne fait rien en temps normal, et tout le programme consiste à attendre des événements extérieurs pour les gérer. Avec l'option ON MENU en particulier, le programme principal ne contiendra plus que la définition des fenêtres des menus, et c'est à partir du choix de l'utilisateur que l'on se branchera vers telle ou telle option. Cette nouvelle manière de programmer renforce encore la nécessité de considérer son programme comme une succession de petites tâches indépendantes. Il est maintenant certain en effet, à cause de la généralisation de la souris et des techniques de menus déroulants et de fenêtres que toutes les applications à venir seront programmées de la sorte. Belle revanche du Basic qui est actuellement le seul langage à intégrer dès aujourd'hui ces nouveaux concepts.

Comme il est dit au début de cet article, l'une des qualités fondamentales du Basic est d'être un langage interprété. Si c'est effectivement un avantage pour la phase de mise au point du programme, cela devient un inconvénient une fois cette mise au point effectuée. En effet, la phase d'interprétation de chaque ordre est faite juste avant l'exécution de chacun d'eux : si le programme passe mille fois sur le même ordre, la phase d'interprétation est refaite mille fois, ce qui ralentit considérablement la vitesse d'exécution du programme. L'idée, pour remédier à cet inconvénient, est de faire, en une seule fois, la traduction des ordres Basic en langage machine, directement exécutable. Cette opération est faite par un compilateur, qui génère, à partir d'un programme écrit en Basic, une suite d'ordres en langage machine. Jusqu'à présent, les compilateurs Basic, à cause de leur prix, étaient strictement réservés aux professionnels. Microsoft vient de rompre cette séparation, en proposant, pour les ordinateurs IBM PC et les compatibles, le Quick Basic, capable de compiler tout programme écrit en GW Basic – le langage livré avec la quasi-totalité des ordinateurs compatibles avec l'IBM PC.

Le Basic du futur

Nous voici donc en droit de nous demander pourquoi le Basic n'est pas plus souvent employé par les programmeurs professionnels. Il faut savoir que de très nombreux programmes commercialisés sont effectivement développés en Basic, mais leurs auteurs n'osent pas l'avouer, de peur justement de ne pas paraitre professionnels. Mais il existe de vraies raisons pour se tourner vers un autre langage. Il manque, en particulier, dans le Basic de Microsoft, l'intégration d'une gestion de fichiers complète, qui intègre l'accès séquentiel, l'accès direct par numéro d'enregistrement, mais aussi l'accès séquentiel indexé par le contenu d'une zone. Il ne faut pas aller chercher ailleurs la raison de l'immense succès de dBase II, qui offre la possibilité de travailler véritablement sur les fichiers. Dans l'idéal, un bon Basic devrait intégrer, comme celui du Macintosh, la gestion des menus déroulants, des fenêtres et de la souris. Dans le monde du MS-DOS, on attend avec impatience de Microsoft, une version du Basic qui permettrait de se servir des outils mis au point pour le logiciel intégrateur Windows. Et bien entendu, un compilateur bon marché ferait du Basic le langage idéal de développement. On peut rêver.

Seymour DINNEMATIN

DES BASIC, ENCORE DES BASIC

Tenir un catalogue des différents Basic disponibles est une tâche sans fin : voici plutôt quelques produits marquants.

IBM PC et compatibles

Le Basic livré avec la machine se nomme Basica chez IBM et GW Basic pour les compatibles. Dans les deux cas, c'est un produit de Microsoft. Avec les dernières versions, il est possible d'accéder à la plupart des fonctions du système d'exploitation par la fonction Shell.

Quick Basic permet de compiler un programme écrit en Basica ou GW Basic. Il ajoute à ces langages un certain nombre de fonctions, comme les sous-programmes que l'on peut compiler séparément et les fonctions multilignes. Quick Basic est un produit Microsoft qui coûte 1 190 F HT.

Mem-DOS est un système de développement qui inclut un Basic, un gestionnaire d'écran et un séquentiel indexé. Il coûte 3 500 F HT et est développé par la société Memsoft.

PC-Master est un ensemble d'une cinquantaine d'ordres qui viennent compléter le Basic de chez Microsoft dans le domaine de la gestion de fichiers et la gestion d'écran. On peut utiliser PC-Master avec le compilateur Quick-Basic. PC-Master est distribué par Micro Application et coûte 800 F HT.

Top Hat, distribué par les Ciments français au prix de 9 000 F HT est un générateur de programmes Basic qui prend à sa charge la gestion d'écran et de fichiers. Le programme généré peut être modifié puis compilé par Quick Basic.

High Screen 2 est un logiciel de gestion d'écran dont les commandes sont exécutées par un petit programme résident en mémoire. Il peut fonctionner avec n'importe quel langage dont le Basic. Il est distribué pac PC-Soft au prix de 4 900 F HT.

Macintosh.

Le Basic 2.02 de Microsoft est l'un des plus avancés du point de vue des fonctions. Le Macintosh est l'une des seules machines qui ne soit pas livrée avec un Basic : il coûte 1 790 FHT, et permet d'accéder à pratiquement toutes les fonctions internes du Macintosh. Un compilateur est prévu dans quelques mois.

Traçorg de la société Syst'infor est un logiciel qui permet de dessiner l'organigramme d'un programme avant la phase de programmation. Il coûte 500 F HT.

Atari

L'Atari ST provoque un grand intérêt de la part des producteurs de Basic.

Mem-DOS sera donné, à partir de septembre avec l'Atari 1040 ST à la suite d'un accord mondial entre Atari et la société niçoise Memsoft. Outre ses possibilités propres, il ouvre aux acheteurs de l'Atari ST la bibliothèque des logiciels verticaux déjà développés avec ce Basic.

Fast Basic est un Basic compilé écrit par la société Philon et distribué en France par Run Informatique au prix de 895 F HT.

Softworks Basic est un autre Basic compilé pour Atari ST, il est également distribué par Run Informatique, au prix de 750 F HT.

Amiga

Sur l'Amiga, Microsoft à développé un Basic qui prend en compte les possibilités de la machine. En particulier, il possède des ordres pour l'utilisation de la synthèse de parole, ainsi qu'un étonnant ordre ON COLLISION qui prend en compte l'interruption générée par le co-processeur graphique en cas de collision de deux objets animés sur l'écran.

Amstrad

Les machines Amstrad bénéficient d'un excellent Basic développé par la société anglaise Locomotive Software. Fidèle à son image professionnelle, le Basic du PCW inclut Jetsam, une bonne gestion de fichiers très complète, mais oublie les possibilités graphiques de la machine.

Thomson

Les machines Thomson sont distribuées depuis qu'elles existent avec un bon Basic de chez Microsoft, pas tout à fait compatible, hélas, avec le GW Basic de l'IBM PC.

MSX

La norme MSX étant à l'origine un produit de Microsoft, le Basic qui accompagne cette norme est très complet, en particulier pour la gestion des interruptions générées par les manettes de jeux, le clavier ou les rencontres de motifs animés sur l'écran.