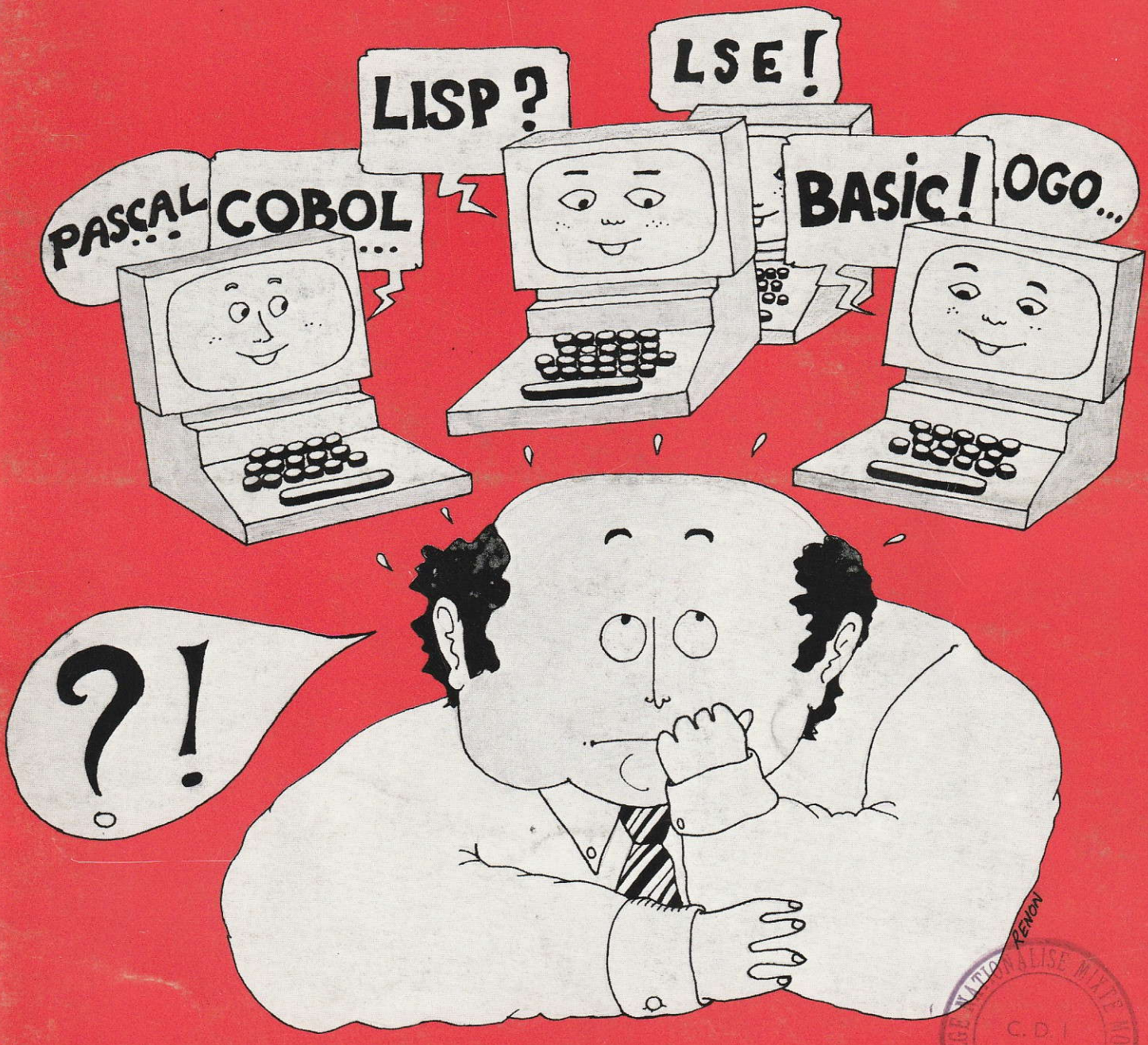
EDUCATION & INFORMATIQUE

N° 7/SEPTEMBRE-OCTOBRE 81/BIMESTRIEL/25 F



DOSSIER :

les langages de programmation

INFORMATIQUE

LES ORDINATEURS, STRUCTURE ET FONCTIONNEMENT DES SYSTÈMES INFORMATIQUES,

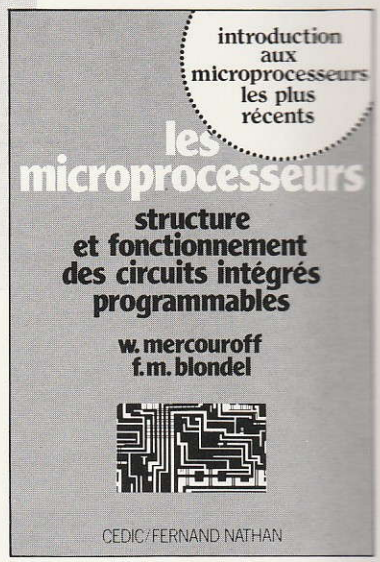
L'ouvrage : Consacré aux aspects matériels de l'informatique, ce livre, par la clarté de son exposé, est accessible à tout lecteur. Il constituera une remarquable mise au point pour les professionnels.

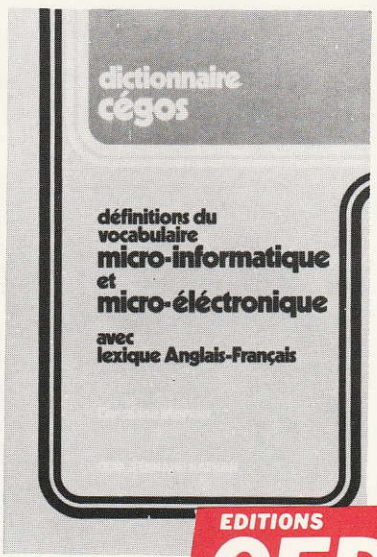
LES MICROPROCESSEURS, STRUCTURE ET FONCTIONNEMENT DES CIRCUITS INTÉGRÉS PROGRAMMABLES

L'ouvrage : Rendu nécessaire par le développement de ce que l'on pourrait appeler une "Nouvelle Electronique", ce livre en décrit l'évolution et s'intéresse aussi aux plus récents des microprocesseurs.
Indispensable au spécialiste, il constitue par la clarté de son exposé une remarquable introduction aux microprocesseurs.

DICTIONNAIRE MICRO-INFORMATIQUE ET MICRO-ÉLECTRONIQUE CEGOS

L'ouvrage : Définissant de façon claire plus de mille termes y compris les plus récents, complété d'un lexique anglais/français et d'un lexique français/anglais, ce livre constitue un ouvrage de référence indispensable aussi bien au professionnel qu'à l'amateur.





EDITIONS CEDIC DIFFUSION FERNAND NATHAN

Eel/9

Les ordinateurs ..... **Réf. 0301** ☐ **55,00 F**

Les microprocesseurs ..... **Réf. 0302** ☐ **45,00 F**

Dictionnaire de micro-informatique et micro-électronique CEGOS ..... **Réf. 0307** ☐ **130,00 F**

Montant : _____ F.

Frais d'envoi : _____ 4,00 F.

Total : _____ F.

Date _____

Ci-joint :
☐ chèque bancaire ☐ postal ☐ mandat

Bon de commande

Nom _____

Prénom _____

Adresse _____

_____ Code postal _____

☐ Enseignant ☐ Entreprise ☐ Particulier

☐ désire recevoir le catalogue
☐ désire recevoir les ouvrages cochés

Signature :

EDITIONS CEDIC

32, Bd St-Germain, 75005 Paris

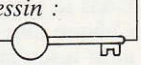
EDUCATION & INFORMATIQUE

programmation

par A. Deledicq

L'objectif des 10 leçons qui vont suivre est d'apprendre à programmer les micro-ordinateurs. Nous essaierons d'éviter à la fois les trop grandes généralités et les petits problèmes techniques associés à une machine particulière.

Les points clefs qui nous apparaissent importants pour une compréhension efficace, sont signalés par le dessin :



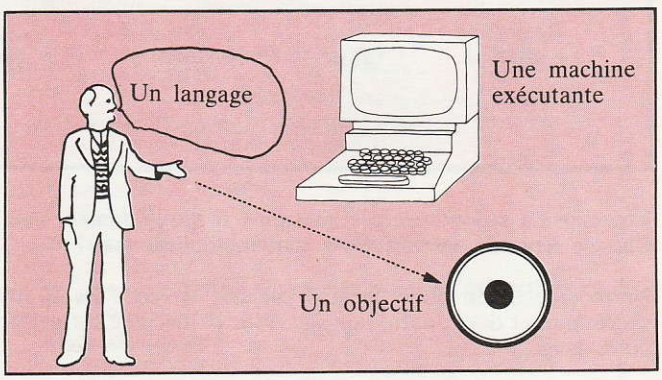
LEÇON I :
Programmeur - Langage - Exécutant

1.1 L'ACTIVITÉ DE PROGRAMMATION

L'activité de programmation suppose la mise en présence de deux « individus » :

- Le programmateur (vous, moi...)
- La machine exécutante (une autre personne, un outil, une machine, un ordinateur...)

Le programme donne des ordres à la machine exécutante dans le but de réaliser un certain objectif. Il faut donc que le programmeur s'astreigne à :



- **Définir un objectif tenant compte des possibilités de réalisation effective de la machine-exécutante**

- Pour cela, on doit connaître la réponse à une question préalable :

« Quel est le **pouvoir-faire** de l'exécutant ?
Écrire sur un ruban ? Déplacer un curseur ? Fournir de l'énergie ? Éclairer des points de couleurs sur un écran de télévision ?...

- **Énoncer une suite d'instructions dans un langage compris par l'exécutant**

- Pour cela, on doit connaître la réponse à une question préalable : « Quelle est la **compétence** de l'exécutant ? c'est-à-dire, à quel type d'ordres est-il capable de réagir par une action déterminée ?
Une phrase en français ? Un appui sur la touche d'un clavier ? Une suite de chiffres en héxadécimal ? La rotation d'un curseur ? Une instruction en PASCAL ?...

① INITIER A L'INFORMATIQUE ?

D'un point de vue éducatif, on peut donc énoncer ceci : TOUT EXERCICE consistant à :

- DÉFINIR LES COMPÉTENCES D'UNE MACHINE EXÉCUTANTE
- DEMANDER L'ÉCRITURE D'UNE SUITE D'ORDRES PERMETTANT D'ATTEINDRE UN OBJECTIF RÉALISABLE PAR CETTE MACHINE EXÉCUTANTE.

constitue un APPRENTISSAGE de L'ACTIVITÉ DE PROGRAMMATION et donc une « INITIATION A L'INFORMATIQUE ».

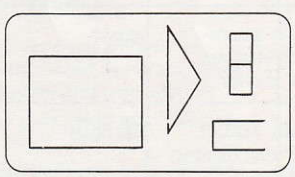
② FAITES LA MACHINE

Pour contrôler qu'un programme aboutit bien à l'objectif désiré, on a intérêt à se mettre à la place de la machine exécutante : on lit le programme pas à pas et on simule l'exécution des ordres successifs. C'est un exercice très profitable au programmeur débutant.

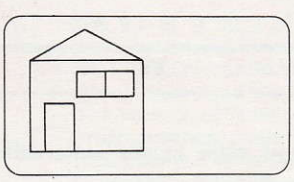
1.2 UN EXEMPLE DE PROGRAMME

La machine exécutante est capable de déplacer des formes numérotées sur un écran quadrillé :

« 1 » est la forme
« 2 » est la forme
« 3 » est la forme



Les instructions qu'elle comprend sont de la forme : « 1XP » où X est l'une des lettres B, H, G, D et P, un entier compris entre 1 et 10. L'instruction 2 D7 par exemple déclenche le déplacement de la forme 2 de 7 carreaux vers la droite « 1 R9 » déclenche la rotation de la pièce 1, dans le sens des aiguilles d'une montre de 9 heures (c'est-à-dire de 3/4 de tour). Cette rotation s'effectue autour du point le plus bas à gauche de la forme.



Voici un programme dont l'objectif est de mettre la « porte » à sa place :

```
3 G7
3 H2
3 R3
```

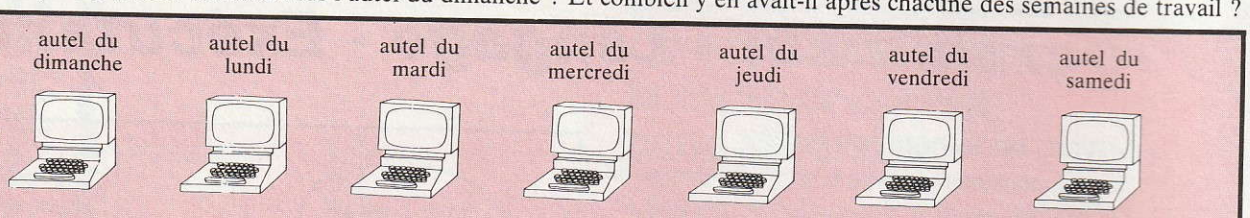
Que fait le programme suivant :

```
1 R9
1 G1
1 H4
2 R9
2 G4
```

Voir le point ②

1.3 UN PROBLÈME : LES SEPT AUTELS DU KAPITAL

Dans le temps du Kapital, il y a 7 autels. Le premier dimanche de l'année, une offrande est déposée sur le premier autel. Chaque jour de la semaine, le peuple dispose sur l'autel correspondant autant d'offrandes qu'il en a sur l'autel du dimanche, ainsi jusqu'au samedi. Et le dimanche suivant, le Grand Prêtre du Kapital réunit toutes les offrandes sur le premier autel. La même règle d'offrande doit être respectée chaque semaine : chaque jour on dépose sur l'autel correspondant autant d'offrandes qu'il y en a sur l'autel du dimanche. Et les semaines se suivent ainsi et le peuple apporte de plus en plus d'offrandes jusqu'aux congés payés à la fin de la 10e semaine.
Combien y a-t-il alors d'offrandes sur l'autel du dimanche ? Et combien y en avait-il après chacune des semaines de travail ?



A la fin de la 2e semaine, le Grand Prêtre a réuni 49 offrandes sur l'autel du dimanche...

Cherchons à répondre à ces questions en programmant une machine exécutante. L'analyse du problème est vite menée : **Chaque semaine, les offrandes sont multipliées par 7**

Notre objectif est donc de faire **calculer et écrire les 10 prèmières puissances de 7.** Précisons maintenant notre machine exécutante et donc, notre langage. Plus la machine est perfectionnée, plus simple est notre programme (car plus évolué est notre langage).

1.4 LE MÊME PROBLÈME : SIX PROGRAMMES

Si la machine exécutant est...	Le programme est écrit en...	Comme ceci...
Un bachelier	français	calcule et écris les 10 premières puissances de 7
Un micro-ordinateur à clavier	basic	1 FOR N = 1 TO 10 2 PRINT 7↑N 3 NEXT N
Un écolier d'une dizaine d'années	français	1. Multiplie 1 par 7 2. Écris le résultat obtenu 3. Multiplie-le par 7 4. Écris le résultat obtenu 5. Recommence encore 8 fois les ordres 3 et 4.
Une calculatrice de poche non programmable	appuyant sur les touches	1 × 7 = × 7 = × 7 = × 7 = × 7 = × 7 = × 7 = × 7 = × 7 = × 7 =
Une calculatrice programmable	une espèce de langage-machine	7 sto 1 1 sto 2 1 sto 3 Lbl 8 Rcl 2 × Rcl 1 = Pause sto 2 1 + Rcl 3 sto 3 − 10 = if × 0 gto 8
Un micro-processeur	un langage assembleur codé en héxadécimal	7 C 20 E 10 4 F 862...

Les 4 derniers exemples doivent être considérés aujourd'hui comme des anecdotes ou de simples informations. En fait, ce sont les deux premiers exemples qui nous intéressent.

Précisons quelques éléments de langage :
- 7 ↑ N est l'écriture en langage BASIC de 7 π (sur une ligne)
- PRINT X est l'ordre en langage BASIC déclenchant l'affichage de X pour la machine

En LSE, on écrit « afficher X »

En BASIC 1 FOR N = 1 TO 10
BLOC 2
3 NEXT N
signifie Exécuter le bloc 2 pour N = 1
Exécuter le bloc 2 pour N = 2
.....
Exécuter le bloc 2 pour N = 10

En LSE, on écrit : 1 FAIRE 3 POUR N = 1 JUSQU'A 10
BLOC 2
3 fin de boucle

« FAIRE 3 POUR... » signifie en fait :
« exécuter les ordres écrits jusqu'à l'ordre n° 3 y compris pour... »

(3) Organisez-vous dans votre langue maternelle

Il vaut toujours mieux
- Commencer par énoncer en français les objectifs et les sous-objectifs d'un problème, même et surtout s'il est compliqué. Suivez donc ce conseil.
- Décrivez d'abord la structure de résolution du problème, en énonçant, en français, les différents points de son développement.

EDUCATION & INFORMATIQUE

SOMMAIRE

RÉDACTEUR EN CHEF :
W. Mercouroff
RÉDACTEUR EN CHEF ADJOINT :
Christian Lafond
COMITÉ DE RÉDACTION, RESPONSABLES DE RUBRIQUES :
J. Arsac - H. Bestougeff - F.-M. Blondel - M. Coutty - P. Muller - J.-M. Salomon
ONT COLLABORÉ A CE NUMÉRO :
Ch. Berthet - H. Bestougeff - F.-M. Blondel - M. Corbel - M. Coutty - A. Deledicq - A. Ducrocq - J. Hebenstreit - J.-L. Laurière - C. Le Roy - B. Levrat - P. Muller - B. Petazzoni - M. Peuchot.
COMITÉ SCIENTIFIQUE :
J. Beghin - A. Danzin - H. Fady - J. Hebenstreit - C. Pair - J. Perriault - J.-C. Simon
SECRÉTARIAT DE RÉDACTION :
M. Corbel
MISE EN PAGE ET ILLUSTRATIONS :
F. Renom
RÉDACTION, ADMINISTRATION, ABONNEMENTS :
9, rue Méchain - 75676 Paris Cedex 14
ABONNEMENTS 1 AN (5 numéros) :
110 F (France) 140 F (Étranger)
PUBLICITÉ :
B. Vedrenne (589.61.85)

Les fiches d'E & I
Nous informons nos lecteurs que deux erreurs se sont glissées dans la fiche « Ordinateur » du N° 6. Il faut lire : — Colonne 1, paragraphe 2 : « Cette machine de Von Neuman est composée **notamment** d'une unité arithmétique... » — Colonne 2, paragraphe 2 (« en ne faisant référence qu'aux possibilités **mathématiques** des machines... »
Merci de bien vouloir nous en excuser...

ET EN SUPPLÉMENT DÉTACHABLE

- **ET & I PROGRAMMATION** Leçon 1 et 2
  Pour le lecteur qui commencerait la lecture d'E & I au numéro 7, nous avons reproduit ici la leçon 1 de programmation, parue dans le numéro précédent (cela permet d'ailleurs d'en corriger quelques erreurs typographiques rendant sa compréhension difficile).
- **LA VOITURE A MICROPROCESSEURS RÉALISÉE AU LYCÉE TECHNIQUE DIDEROT** (Suite et fin)

Exposez au Premier Marché International

des Simulateurs,
Machines à Enseigner, (EAO)
Machines Intelligentes, (CAO - CFAO - RAP - SAP)
Jeux Pédagogiques d'Entreprise.

**Si vous êtes Concepteur,
Fabricant de systèmes,
Fabricant de composants,
Distributeur, Prestataire de logiciel,
Prestataire de didacticiel
Editeur ou Consultant,
le SIMEP'82 vous concerne.**

**Une occasion unique
de négocier des contrats,
de présenter et vendre en 5 jours
vos matériels et services, aux acheteurs
et utilisateurs du monde entier.**

SIMEP'82

du 22 au 26 Mars 1982 Cannes – France
Palais des Festivals

SIMEP'82

Veuillez m'envoyer, sans engagement, des informations complémentaires sur le SIMEP.

Nom : _____ Fonction : _____

Société : _____ Activité : _____

Adresse : _____ Téléphone : _____

Retournez ce coupon à : MIDEM ORGANISATION, 179 Av. Victor Hugo, 75116 PARIS – France – Tél. : 505.14.03. Télex : 630.547 MID-ORG
U.K. INTERNATIONAL EXHIBITION ORGANISATION LTD, JACK KESSLER, 9 Stafford Street LONDON, W1X 3PE Tél. : 499.23.17. Télex : 25230 MIP/MIDEM LDN
U.S.A. Perard Associates inc c/o JOHN NATHAN, 30 Rockefeller Plaza, Suite 4535 NEW YORK NY 10 112. Tél. : (212) 489.13.60. Télex : 235.309 OVMUR UR

ÉDITORIAL

Le dégel

Début juin, Monsieur Savary, Ministre de l'Éducation Nationale, décidait le « gel » du plan informatique. Gel ne veut pas dire disparition, et comme l'écrivait le Ministre : « Cette décision n'était pas motivée par une réticence à l'égard de l'introduction de l'informatique dans le système éducatif, mais reposait sur la conviction que la concertation avec les différents partenaires, indispensable pour une opération de cette nature n'avait pas été effectuée, que la finalité pédagogique de l'opération avait été insuffisamment prise en compte et, d'une façon plus générale, que les différents aspects du plan n'avaient pas été conçus dans leurs relations mutuelles ».

Aussi le Ministre décidait-il, début juillet, de nous confier une mission de concertation et de réflexion pour étudier comment l'action pourrait être reprise dans de bonnes conditions dès la prochaine rentrée et dans les années ultérieures.

L'exercice, un peu délicat, consistait donc à fixer rapidement quelques principes de manière à pouvoir proposer dès maintenant les mesures à prendre à la rentrée. Il fallait surtout éviter de perdre un an...

Heureusement, on ne partait pas de zéro et il n'y avait pas à pratiquer la politique de la table rase. Heureusement surtout, un très large consensus s'est dégagé des consultations opérées.

La première priorité est la formation. Formation approfondie et on ne peut manquer de faire référence à ce qui s'est passé entre 1970 et 1976 pour des enseignants qui seront ensuite chargés de former leurs collègues, de participer à des recherches, de créer et de diffuser des logiciels. A la rentrée seront ouverts 10 ou 11 centres de formation d'un an, dans des établissements universitaires, répartis sur tout le territoire. Ces centres devraient aussi être des foyers de recherches et de rencontre, en liaison avec les CRDP.

La recherche est d'ailleurs aussi une priorité. Et pour elle comme pour la formation, nous pensons qu'il faut faire largement appel à l'Enseignement supérieur, reprenant là aussi et élargissant les pratiques du passé. Il ne s'agit pas pour autant d'écarter l'INRP dont nous souhaitons que dans les années futures il joue un rôle accru.

Nous pensons aussi qu'il faut sortir de cette espèce de guerre de religion qui s'est élevée depuis quelque temps entre les tenants de l'informatique d'enseignement et ceux de l'enseignement de l'informatique. Il y a là deux pôles, mais entre eux une continuité : modéliser fait-il partie de la démarche informatique ou est-il un objectif de l'enseignement ? Employer une forme de programmation pour développer les capacités logiques est-ce utiliser l'informatique comme outil ou enseigner la discipline ?

En fait, l'informatique est un élément de la culture d'aujourd'hui. Et il est donc important d'étudier dès maintenant son apport à la formation générale de nos élèves. C'est pour procéder à cette étude que seront ouvertes dans 10 ou 12 lycées des options expérimentales en classe de seconde, avec des équipes d'enseignement qui s'y sont préparées.

Beaucoup de questions restent à traiter : production des logiciels, rôle de la télématique... Comme le Ministre l'a indiqué avec force, c'est dans la concertation que l'introduction de l'informatique dans l'enseignement doit être poursuivie.

Yves LE CORRE
Claude PAIR

LES PRINCIPALES MESURES PRISES

FORMATION DES ENSEIGNANTS *— 200 enseignants répartis dans 10 centres universitaires couvrant l'ensemble du territoire, recevront une formation d'un an.*

MICRO-ORDINATEURS *— Le marché 81 portera sur 1 250 ordinateurs et 300 imprimantes. Tous les centres de formation approfondie seront équipés. Les choix des établissements scolaires seront effectués par les directions du Ministère en liaison avec les Recteurs.*

Notons que sur ces deux premiers points, il n'est pas fait mention du niveau d'enseignement ; on peut donc penser que les collèges et les écoles normales seront désormais également concernés par l'informatique.

L'OPTION INFORMATIQUE *— Elle sera mise en place à titre expérimental dans 10 à 12 lycées en classe de seconde. C'est un enseignement de type général, qui n'a pas pour but de former des informaticiens, et qui sera assuré par les équipes de professeurs qui ont suivi l'an dernier une formation complémentaire.*

Cette option qui se veut plus « pratique » que théorique et ouverte sur les autres disciplines devrait, si l'expérience s'avère positive, être prolongée en 1re et Terminale.

On précise au ministère de l'Éducation nationale que Messieurs Pair et Le Corre poursuivent leur mission jusqu'au mois de septembre pour préparer un plan à plus long terme et que des instances de concertation pédagogique, nationales et académiques, seront mises en place en 81/82 pour étudier tous les problèmes de l'informatique scolaire.

La microgénération

Albert Ducrocq

Oui, les jeunes réalisent qu'ils vivront une civilisation de l'ordinateur, vis-à-vis de laquelle ils sont loin de nourrir un sentiment d'effroi. Mieux : ils voient volontiers dans l'informatique un remarquable outil intellectuel. Après le temps de l'accroissement des productivités à l'usine et au bureau, il leur apparaît naturel que des progrès interviennent dans la communication du savoir. Ils sont réceptifs. Toute la question est de savoir comment, dans la pratique, l'ordinateur va faire sa grande entrée dans l'enseignement.

Or la tendance actuelle permet de discerner deux mouvements qui se situent à des niveaux différents et semblent appelés à se développer simultanément, chacun obéissant à la logique de son évolution.

La « voie royale »...

Nous découvrons d'abord ce que l'on pourrait appeler « la voie royale » avec l'emploi systématique d'un micro-ordinateur pour l'ensemble des disciplines, un micro-ordinateur dont on aura appris à manier le clavier aussi aisément que le stylo l'était hier, en utilisant les programmes actuels, en n'hésitant pas à créer ceux qui n'existent pas.

Écrire des programmes ? Non seulement certains jeunes excellent dans cet art, mais les meilleurs didacticiels — nous avons été frappés de le constater — sont souvent conçus par des élèves qui, venant d'assimiler une notion et ayant, toute fraîche dans leur esprit, la démarche intellectuelle qu'elle implique, sauront la décortiquer de façon particulièrement heureuse à l'intention de leurs camarades.

Cette voie royale exige toutefois la possession d'un micro-ordinateur. Et là réside incontestablement un goulot d'étranglement. En dépit de l'abaissement des coûts, l'Éducation nationale ne saurait offrir à court terme un micro-ordinateur à chaque élève, comme elle lui assure la gratuité de ses manuels. Et c'est par ailleurs dans un nombre de foyers relativement réduit que le micro-ordinateur fera son apparition au cours des prochaines années.

Ajoutez à cela des facteurs psychologiques. Au temps des copistes, l'imprimerie avait été présentée comme une invention de Satan. L'informatique, notions-nous ci-dessus, ne fait pas peur aux jeunes. Elle est loin de tous les enthousiasmer pour autant : nombreux sont ceux qui n'en voient pas la nécessité, la trouvant ennuyeuse, ou jugeant fastidieux d'apprendre le langage qui leur permettrait d'utiliser une machine. Cette réticence à faire entrer le micro-ordinateur dans leur vie intellectuelle est particulièrement marquée dans la jeunesse féminine. Alors que dans les classes pilotes où le micro-ordinateur enseigne la musique, le français ou les sciences naturelles, les filles réussissent aussi bien que les garçons, la proportion d'éléments féminins nous est apparue particulièrement faible dans les activités laissant à option l'usage de micro-ordinateurs, comme si l'on assistait à un combat de la poésie contre le rationalisme.

En réalité, l'informatique est au service de l'une comme de l'autre de même que la plume permet d'écrire aussi bien des sonnets qu'une équation. L'effet d'entraînement ne va pas manquer de jouer mais à un rythme qui promet de rester à l'unisson des programmes d'équipement, de sorte qu'à court ou moyen terme, on ne saurait attendre que la masse des élèves se presse sur cette voie royale des précurseurs.

De la calculatrice...

L'autre mouvement, au contraire, revêtant les traits d'une lame de fond, nous paraît appelé à conduire pratiquement toute la jeunesse à adopter l'informatique par étapes et cela presque à son insu, ce second mouvement ayant pris pour point de départ la petite calculatrice.

Le fait est là en effet : tout le monde possède une calculatrice de poche, cet outil faisant aujourd'hui partie de la trousse de l'écolier.

Une calculatrice, objecterez-vous, ce n'est pas un micro-ordinateur. Certes, mais avec elle sont déjà prises les habitudes d'assistance intellectuelle, annonciatrices d'une civilisation de l'informatique.

D'aucuns affirmeront a priori : ces habitudes sont mauvaises. Les jeunes risquent de ne plus savoir calculer demain. Nous ne partageons pas ce point de vue et considérons que la machine va obliger l'écolier à réfléchir beaucoup plus qu'au temps où l'apprentissage de la table de multiplication relevait du psittacisme. Demain, cette table de multiplication, les jeunes la découvriront en effet a posteriori : ils réaliseront pourquoi il est intéressant, quand on doit souvent faire des calculs, de savoir sa table. Et dans cette considération, ils pourront trouver une incitation à l'apprendre. Mettons-nous en effet un instant dans la peau de l'écolier de 8 ans auquel on vient d'offrir une calculette alors que, depuis deux ans, il apprenait tant bien que mal à compter. Sa première réaction est l'émerveillement : « C'est extraordinaire, s'exclame-t-il, je n'ai plus besoin de faire les opérations ! Il n'est plus nécessaire de savoir combien font 7 fois 9. Je n'ai qu'à appuyer sur les touches 7, × et 9 et à lire le résultat ! »

Mais vivons la suite. Lorsqu'à longueur de mois le même écolier est astreint à faire des opérations, il comprend que savoir une fois pour toutes combien font 7 fois 9, c'est beaucoup plus agréable qu'être obligé chaque fois de sortir sa calculatrice, mettre le contact et frapper sur trois touches. Apprendre par cœur que 7 fois 9 font 63 ? Mais *cela se sera fait tout seul* dès l'instant où l'on aura commandé un certain nombre de fois ce calcul. Autrement dit, à l'usage, la calculatrice sera devenue une machine à enseigner la table des multiplications courantes.

L'élève n'aura pas appris méthodiquement comme autrefois sa table des 2, puis celle des 3 et ensuite celle des 4. Mais il aura été conduit dans le désordre à connaître les résultats d'un nombre croissant de multiplications. Bientôt il jugera inutile de recourir à sa machine pour les multiplications usuelles, la réservant pour les autres, pour celles faisant intervenir des nombres compliqués.

Ces dernières, il refusera sans doute obstinément de les effectuer manuellement en appliquant les règles de l'arithmétique. Mais les adultes n'en sont-ils pas déjà là ? Une page a probablement été tournée, le temps est révolu où nous consentions à perdre un temps proche de la minute pour multiplier entre eux deux nombres tels que 365 et 86 400 : les machines ne sont-elles pas capables de faire ce travail en moins de 10 secondes (temps essentiellement requis, cela va sans dire, par la manipulation des touches).

Tel est le sens de l'évolution actuelle : la procédure d'extraction d'une racine carrée sombre dans l'oubli. Les règles de la multiplication de nombres à plusieurs chiffres vont sans doute suivre la même voie. En revanche, les « ficelles » permettant d'obtenir un résultat plus vite qu'avec la machine seront appréciées. Ainsi, paradoxalement, le fait de ne plus apprendre bêtement la table de multiplication par colonnes pourrait demain puissamment favoriser le calcul mental et la gymnastique intellectuelle ; la calculette apparaîtra autant comme

une assistante intellectuelle que comme une stimulante dans la mesure où l'individu sera enclin à rechercher l'astuce qui lui permettrait d'aller plus vite qu'elle. Et sans doute, y a-t-il là dès aujourd'hui une carte maîtresse à jouer sur le plan pédagogique : enseigner les moyens de calcul grâce auxquels les élèves constateront qu'ils peuvent être plus rapides que les machines.

Les opérations que l'on ne peut pas faire de tête exigent parfois de longues séquences de calculs.

Mais là, nous découvrons le second élément extraordinairement positif pour les jeunes que représente la calculette, dans le fait que l'on peut aujourd'hui tabler sur une *continuité* depuis la petite machine à faire les quatre opérations de l'école primaire, jusqu'au micro-ordinateur riche de sa bibliothèque de disquettes en passant par l'étonnante gamme des machines à jouer qui viennent renouveler l'enseignement à la fois parce qu'elles permettent d'apprendre l'histoire ou le latin d'une autre manière — et, à l'adresse des rudiments de ces disciplines, nous tiendrons le même langage que pour la table de multiplication en arithmétique — mais aussi parce que ces machines à jouer viennent remettre en cause le découpage traditionnel du savoir et de la formation. Pourquoi certaines disciplines figuraient-elles hier sur la liste des matières à enseigner alors que d'autres — comme le jeu d'échecs dont les vertus sont pourtant remarquables — en avaient-elles été exclues ?

Au même titre que tombe le cloisonnement qui hier voulait que l'on apprenne par cœur la table de multiplication des 10, mais pas celle des 11, ni celle des 12, la structuration même de l'enseignement est appelée à relever de formules beaucoup plus souples : les jeunes sont au demeurant conscients de souvent se préparer à la vie autant, sinon mieux avec la considération de sujets parascolaires qu'en se limitant aux programmes traditionnels.

Les calculatrices dites scientifiques ? Il est caractéristique qu'elles soient adoptées par des sujets de plus en plus jeunes et qu'à travers elles, nous soyons témoins de la même démarche intellectuelle : l'exécution d'une opération en fait comprendre la nature comme, autrefois, dans l'aventure biologique la fonction avait précédé l'organe. Il n'est pour s'en convaincre qu'à observer ce garçon de 12 ans appuyant sur la touche « log » puis frappant 1, 10, 100, 1000... Il lit 0, 1, 2, 3... S'enhardit-il à frapper des nombres compris entre 100 et 1000 ? Il trouve le 2 suivi de décimales. Nous l'entendrons alors s'exclamer, tel Archimède disant eureka : « Le logarithme, mais c'est la quantité de chiffres que le nombre va avoir... »

... au microordinateur

Non moins significative apparaît l'adoption de la calculatrice à programme par des sujets toujours plus jeunes, avec là également un puissant intérêt pédagogique, dans la mesure où elle conduit à découvrir la notion de formule et à assimiler l'algèbre. Pour calculer la surface d'un cercle, quoi de plus simple, ayant désigné le rayon par A, que de programmer la multiplication de A par lui-même, la multiplication du résultat par pi et l'affichage du résultat. Ayant compris la démarche en quelques minutes, le jeune de s'extasier : « C'est fantastique, il me suffit d'introduire le rayon et j'ai directement la surface du cercle ».

De fil en aiguille, imaginez des programmes aussi compliqués que vous voulez, utilisez la Sharp 1211 pour faire la jonction avec le monde des micro-ordinateurs et les jeunes auront été amenés sans s'en rendre compte à converser avec le micro-ordinateur dont alors l'emploi leur sera devenu tout naturel. Sic itur ad astra...

Tandis que des éclaireurs empruntent la voie royale, il nous apparaît ainsi que le gros de la troupe écolière et estudiantine va cheminer sur cette seconde voie, ouverte par la filière des calculatrices dites de poche.

La jeunesse nourrissait hier la passion du vélo, puis celle de la mobylette, et ensuite de la voiture, considérées comme représentant autant d'étapes vers une domination matérielle du monde. A l'âge de sa domination intellectuelle avec l'informatique, nous constatons que calculette simple, machine programmable et micro-ordinateur font revivre la même démarche, avec cette différence qu'au lieu d'obéir à la loi du doublement tous les quinze ans, comme la circulation routière, la circulation de l'information sur les autoroutes insaturables de l'esprit double à l'heure actuelle quasiment tous les ans.

A. Ducrocq

LAUSANNE - WCCE 81 - LAUSANNE - WCCE 81 - LAUSANNE WCCE 81 - LAUSANNE WCCE 81

La 3e Conférence mondiale informatique et Enseignement organisée par l'I.F.I.P. s'est déroulée à Lausanne du 27 au 31 juillet 1981. Plus de 1 200 participants venant de plus de 60 pays, 100 communications présentées, une grande exposition de matériels et de logiciels...

La Rédaction d'E & I a, bien entendu, suivi pour vous cette conférence.

Nous en rendrons compte de la façon la plus complète possible dans le dossier :

SPÉCIAL LAUSANNE - WCCE 81

qui paraîtra dans,

NOTRE PROCHAIN NUMÉRO

Brève histoire des langages

J. Hebenstreit

La notion de langage de programmation perd progressivement de son mystère, et si les sigles comme FORTRAN, BASIC, APL, PLI, ALGOL, etc., restent encore peu connus du grand public, tout le monde, ou presque, sait, aujourd'hui, au moins par ouï-dire, que les langages de programmation servent à communiquer avec les ordinateurs ou, plus simplement, à écrire les programmes.

Tous les ordinateurs fonctionnent en binaire, ce qui signigie que les informations à faire traiter par un ordinateur doivent être codées avec des 0 et des 1.

Les tout premiers ordinateurs n'acceptaient que des informations écrites en binaire, c'est-à-dire que le programmeur devait écrire son programme en binaire ! Pour obtenir l'addition de 21 et 6, il fallait écrire :

10 110	010101	000110

Pour simplifier le travail du programmeur, on inventa, dans un premier stade, les *langages d'assemblage,* déja nettement plus proches des habitudes humaines puisque dans un tel langage, l'instruction précédente s'écrira :

ADD 21,6

la machine étant munie d'un programme de traduction qui aura pour tâche de traduire ADD21,6 en code binaire qui est le *seul* exécutable par la machine.

En règle générale, en langage d'assemblage, chaque instruction correspond à une instruction de la machine, c'est-à-dire à une des opérations élémentaires que l'ordinateur est capable d'exécuter.

Le langage d'assemblage, malgré sa complexité (de 60 à 300 instructions différentes selon les machines) est encore, aujourd'hui, utilisé par les professionnels de l'Informatique, dans certains cas particuliers.

Par contre, cette même complexité rend son usage difficile pour traiter les problèmes usuels car on aboutit à des programmes très longs et inutilement détaillés. La longueur des programmes, liée au peu d'efficacité de chaque instruction, a, en effet, pour conséquence de rendre les programmes peu lisibles, donc difficiles à mettre au point, et à modifier.

C'est vers les années 50 qu'apparaît le premier langage un peu évolué :

FORTRAN

Vers les années 60, l'Administration américaine a demandé à un groupe d'experts de définir un langage adopté à ces problèmes ; ce fut la naissance du langage COBOL.

Parallèlement, un groupe d'universitaires européens, peu satisfait des lacunes du langage FORTRAN, définit un nouveau langage destiné au calcul scientifique : ALGOL 60 (Algorithm Oriented Language). Ce langage fut peu utilisé mais il eut une riche descendance.

Jusque-là, les programmes devaient nécessairement être recopiés sur des cartes perforées. Le paquet de cartes perforées représentant le programme était placé sur un « lecteur de cartes perforées » qui lisait la totalité du paquet. Il fallait ensuite attendre l'exécution du programme et la sortie des résultats sur une imprimante. L'utilisateur n'avait donc aucun moyen d'intervenir dans le déroulement du programme.

Le développement de l'utilisation des ordinateurs en temps partagé allait faire naître un type de langage particulier, au moins par sa mise en œuvre, à savoir les langages de type « interactif ».

En effet, un ordinateur en temps partagé partage, comme son nom l'indique, son temps de traitement entre un certain nombre de terminaux reliés à l'ordinateur. L'utilisateur d'un tel système, assis devant son terminal (machine à écrire ou écran de tube cathodique muni, d'un clavier), peut entrer son programme directement au clavier. Il était normal, dans ces conditions, de lui permettre d'intervenir dans le déroulement de son programme, de l'arrêter, de le corriger, de le reexécuter, etc. Mais ceci suppose des langages de type particulier. Le premier en date des langages interactifs ne fut pas BASIC (Beginner All-purpose Scientific Instruction Code - 1964) comme on le dit trop souvent, mais un langage utilisant un *vocabulaire français* et inventé en France à la SEA, en 1959 : le langage PAF (Programmation Automatique des Formules), qui était utilisé sur les ordinateurs CAB 500 fabriqués par la SEA (Société d'Electronique et d'Automatisme).

C'est dans la lignée du PAF et en s'inspirant largement d'ALGOL

que fut créé, en 1971, au service Informatique de l'ESE, le langage LSE qui fut ensuite étendu à deux reprises ; d'abord par l'adjonction d'instructions graphiques, puis par l'adjonction de commandes supplémentaires lui donnant les caractéristiques d'un langage « temps réel ».

Entre temps s'était en effet développée l'utilisation des ordinateurs pour le contrôle de processus (pilotage de colonnes à distiller, de fours de cimenteries, de chaînes d'assemblage, etc.), ce qui posa de nouveaux problèmes et aboutit à la création de langages « temps réels » comme RTL, CORAL, PEARL, SIMULA, ADA, etc., destinés à la programmation de ces problèmes particuliers.

Parmi le millier de langage de programmation apparus au cours des 20 dernières années, deux d'entre eux méritent une mention particulière. L'un est APL. L'autre est LISP. Ce dernier langage, très utilisé en intelligence artificielle, est, en quelque sorte, l'ancêtre d'une nouvelle famille de langages dits « applicatifs » par opposition à tous les autres qui sont dits « procéduraux ». Les langages applicatifs sont encore peu répandus car ils nécessitent pour leur exécution des ordinateurs très puissants, mais leurs caractéristiques en font des candidats sérieux pour l'avenir. Deux de ces langages ont déjà fait une apparition remarquée sur le marché, ce sont LOGO et SMALLTALK.

Programmation et langages

Un langage de programmation à plusieurs fonctions :

— il doit permettre de spécifier les « objets » que l'on veut faire manipuler par l'ordinateur

— il doit permettre de spécifier les « opérations » que l'on veut manipuler par l'ordinateur

— il doit permettre de spécifier l'ordre dans lequel on veut effectuer ces opérations

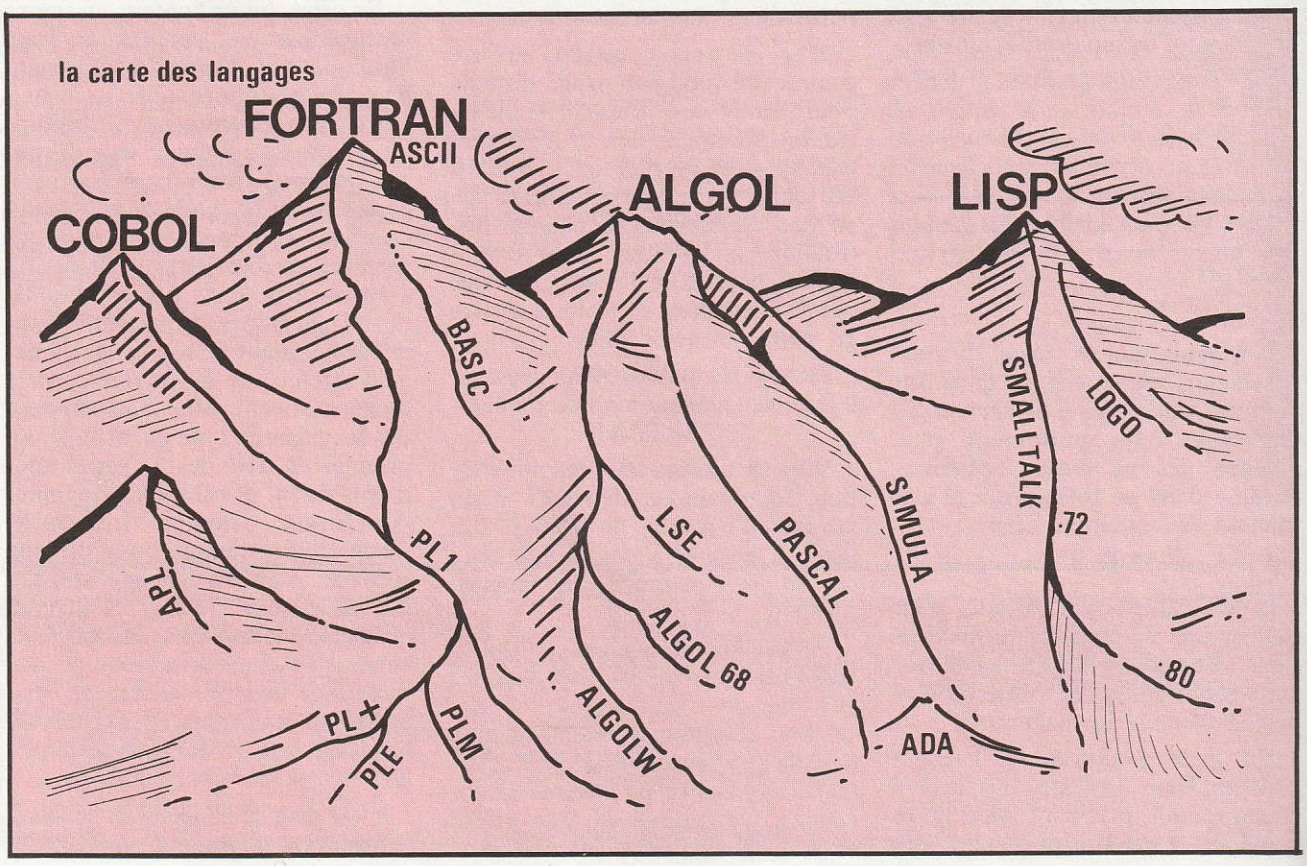
— il doit, obligatoirement, être conçu de manière à pouvoir être traduit de manière automatique dans le langage de la machine (binaire), ce qui suppose que chaque opération spécifiée dans le langage doit pouvoir être décomposée automatiquement (par l'ordinateur lui-même) en une liste d'opérations élémentaires que l'ordinateur est capable d'effectuer par construction.

C'est cette variété de fonctions qui explique la variété des langages de programmation.

Les objets et les opérations

On dira qu'un langage est de type mathématique si les objets qu'il peut manipuler sont des nombres, des listes de nombres ou des tableaux de nombres. Les opérations sur les nombres sont les opérations arithmétiques usuelles (+, −, x, /) ; c'est le cas de FORTRAN et d'ALGOL. Par contre, dans ces langages, les listes et tableaux de nombres ne sont pas des « objets », contrairement à ce qui se passe, par exemple, en APL, où il existe une opération d'addition deux à deux des nombres de 2 tableaux de nombres. On peut, en ALGOL ou FORTRAN, faire ces sommes mais il faut, pour cela, écrire plusieurs lignes de programme. En LSE et dans certaines versions sophistiquées de BASIC, les « chaînes de caractères » sont des objets et il existe une opération de recherche d'une sous-chaîne dans une chaîne (exemple : Trouver si le mot « chat » figure dans la phrase « La mère Michel a perdu son chat »). En FORTRAN et en Mini-BASIC, il est impossible de définir une chaîne de caractère.

Dans les langages dits « graphiques », les « objets » sont les graphismes dessinés sur l'écran d'une console et on dispose, dans ces langages, d'opérations telles



que l'homothétie, la rotation, la translation, l'effacement de l'image ou d'une partie de l'image affichée.

Dans les langages dits « langages d'auteurs » et destinés à l'EAO comme COURSE WRITER, TUTOR, PILOT, etc., les « objets » sont des textes à afficher sur un écran, des réponses à analyser, etc.

Par contre, pour les langages destinés à la commande des machines-outils à commande numérique comme APT, les « objets » sont des déplacements, en trois dimensions, de la pièce à usiner où de l'outil ou encore le changement d'outil, la mise en marche ou l'arrêt du liquide de refroidissement de la pièce, etc.

Ce qui précède permet de comprendre pourquoi il est impossible d'aboutir à un langage « universel ». En effet, un langage permettant de définir tous les « objets » possibles et imaginables et l'ensemble de toutes les opérations possibles sur ces objets hétéroclites, serait d'une complexité qui rendrait son usage pratiquement impossible.

Par ailleurs, la programmation en « langage naturel » qui a longtemps été l'un des thèmes favoris de l'« intelligentsia artificielle » (selon le néologisme du Prof. Weitzenbaum du MIT), est une notion ambiguë car qu'est-ce qu'un langage naturel ? Est-ce que U = RI fait partie du langage naturel ? S'il n'en fait pas partie, la phrase « la différence de potentiel entre deux points d'un circuit est égale à la résistance de ce circuit, multipliée par l'intensité du courant qui la traverse » en fait-elle partie ?

Si le langage naturel est le langage habituel, à l'exclusion de tout signe ou symbole tels qu'ils sont utilisés en sciences, la programmation va rapidement devenir un cauchemar, à cause des interminables définitions qu'il va falloir donner pour paraphraser les notations habituelles. Ce serait, par ailleurs, un recul considérable par rapport à la situation actuelle tant il est vrai que de nombreux progrès n'ont pu être faits que grâce à l'invention d'une notation adéquate.

Si, par contre, on entend par langage naturel un langage de programmation plus proche de la manière humaine de penser que les actuels langages, images à peine déguisées du mode de fonctionnement des ordinateurs, alors comme on l'a déjà dit, les langages applicatifs ont de bonnes chances de triompher demain.

Conclusion

Tous les langages actuels, à de rares exceptions près, sont des langages algorithmiques, c'est-à-dire des langages permettant de décrire de manière rigoureuse la liste des opérations successives formant un algorithme.

Cependant des tendances nouvelles apparaissent qui risquent de bouleverser plus ou moins rapidement la pratique actuelle de la programmation.

— Les « générateurs de programme » qui fabriquent le programme à partir de sa description.

— Des générateurs interactifs de programme où la machine pose des questions à l'utilisateur sur ce qu'il veut faire et fabrique le programme correspondant à partir des réponses de l'utilisateur (NO-CODE de General Automation).

— Des langages de très haut niveau basés pour l'essentiel sur des opérations de type « théorie des ensembles » dont les langages applicatifs sont de bons exemples.

— etc.

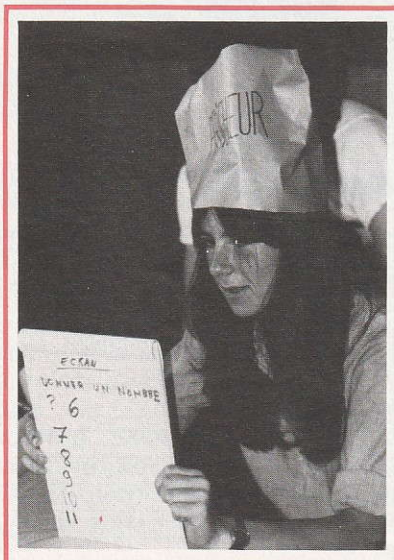
Il est peu probable que ces « nouveaux langages » remplacent les langages de programmation actuels qui sont, dans l'ensemble, assez bien adaptés aux besoins des professionnels de la programmation. Il est par contre probable que ces « nouveaux langages » se développeront rapidement pour des nouvelles classes d'utilisateurs auxquelles ils permettront de communiquer avec les ordinateurs sans passer par les contraintes des langages actuels.

Une telle évolution était d'ailleurs prédite dès 1979 par Terry Winograd du MIT dans un article intitulé « Au-delà de la programmation » :

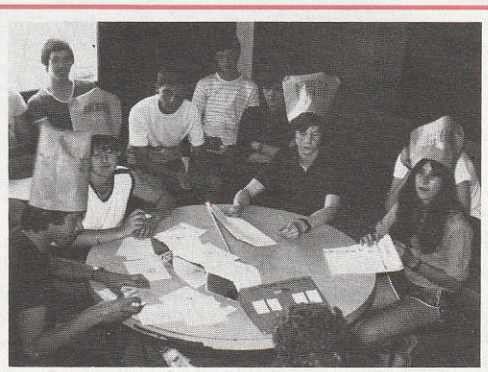
« L'utilisation fondamentale de la programmation ne consiste pas dans la création de séquences d'instructions pour accomplir des tâches mais dans l'expression et la manipulation de descriptions de processus de traitement et des objets qui sont impliqués. La programmation dans le futur s'orientera de plus en plus vers la description de comportements.

La machine doit être perçue comme un mécanisme avec lequel nous inter-agissons et non comme une abstraction mathématique qui peut être complètement caractérisée en termes de résultats. »

J. Hebenstreit



QUE FAIT CETTE JEUNE FILLE ?
1 UNE RECETTE DE CUISINE
2 SON PROCHAIN LOTO
3 UN MESSAGE CODÉ
4 AUTRE...
(Photo A.N.S.T.J)

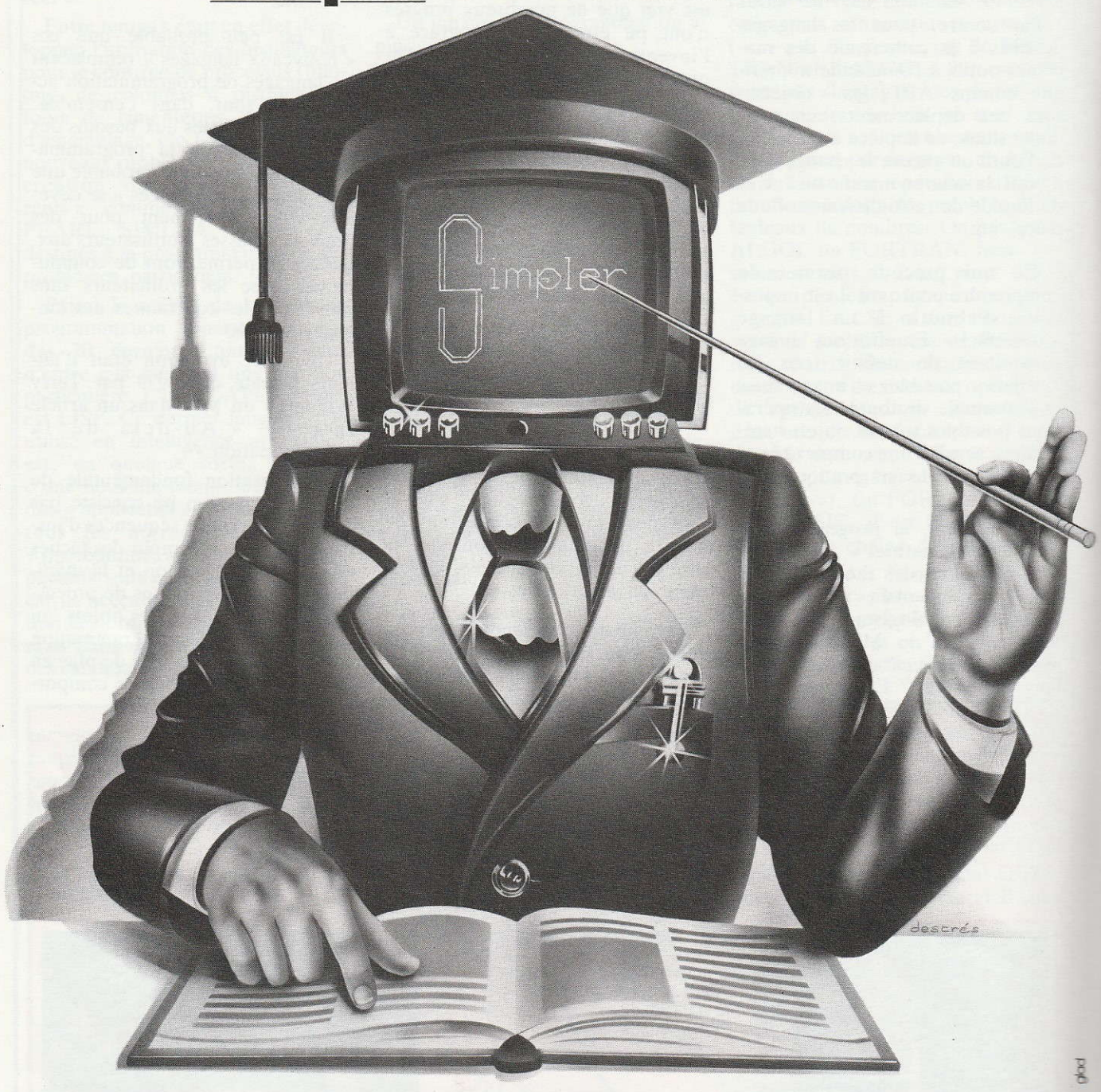


Elle joue le rôle de « l'afficheur » dans le jeu de l'ordinateur. Chacun remplit ainsi les fonctions d'une partie d'un système informatique.

Ayant bien compris le fonctionnement, on n'en apprécie ensuite que mieux les possibilités des langages évolués.

Ce jeu fut la 1re activité du camp informatique d'été organisé par l'Association Nationale Sciences et Techniques Jeunesse (A.N.S.T.J) *qui a connu cette année un grand succès. Photo A.N.S.T.J*

Enseignement assisté par ordinateur : Simpler ouvre la 3e voie.



Jusqu'à présent, en EAO, vous n'aviez guère le choix. D'un côté, le grand système évolué, très performant mais très coûteux, qui faisait appel à un gros ordinateur. Et de l'autre, le micro-ordinateur individuel, limité dans son prix... comme dans ses possibilités.

Maintenant AS & I distribue Simpler en France. Simpler, toutes les capacités d'un grand système, avec un coût au poste de travail équivalent à celui d'un micro. Simpler est issu de l'Université de l'Illinois qui depuis 1959 est le foyer du concept d'EAO et le prestigieux développeur des outils les plus en vue dans ce domaine.

Simpler dispose du langage Tutor pour l'écriture des didacticiels et ne se limite pas à n'être qu'un magnifique instrument d'EAO. Il est de plus ouvert vers toutes les applications de l'entreprise ou de l'organisation.

Le produit Simpler est exploité sur matériel

MODCOMP

as&i

Demande de documentation Simpler
à envoyer à AS & I - 8, rue Charles-Pathé
94300 Vincennes - Tél. : 374.59.49
ou à Modcomp - 17, rue des Solets
94513 Rungis Cedex 115
Tél. : 687.33.82.

Nom_____
Fonction_____
Société_____
Adresse_____
Tél._____

LES ACTES DE NAISSANCE DE QUELQUES LANGAGES

Ch. Berthet

Il existe à l'heure actuelle plusieurs centaines de langages de programmation. En pratique, moins d'une dizaine d'entre eux sont très utilisés tandis que quelques autres connaissent un relatif succès :

- ***FORTRAN** est le langage le plus utilisé pour les applications scientifiques numériques.*
- ***ALGOL** a joué un rôle important dans l'évolution des langages mais n'a jamais été utilisé.*
- ***COBOL** est le plus important des langages en gestion.*
- ***PL/1** se veut un langage universel.*
- ***BASIC** et LSE sont destinés à un usage interactif.*
- ***APL** est très différent des autres langages et connaît un certain succès.*
- ***PASCAL** est un langage récent conçu pour faciliter la programmation structurée.*
- ***ADA** qui n'existe encore que sur le papier est plus particulièrement destiné à traiter les problèmes dits de « temps réel » (Commande de processus, etc.)*

D'autres langages comme le LISP, C, FORTH, CPL1, ne sont actuellement utilisés que par une minorité de professionnels tandis que les langages applicatifs comme SMALLTALK et LOGO (sous-ensemble de SMALLTALK), commencent à faire une timide apparition.

FORTRAN

1954 — Père : John Backus (IBM)

Avant 1954, toute la programmation se faisait en langage machines ou en langage d'assemblage. C'était lourd, long, pénible et cher. A cette époque, la naissance de FORTRAN a constitué l'une des plus importantes révolutions de l'histoire de l'informatique. FORTRAN (pour FORmula TRANslation), langage de traduction de formules algébriques en langage machine) n'a pas été accepté facilement. On prétendait qu'un compilateur serait toujours moins bon que les bons programmeurs en langage machine. Actuellement, FORTRAN est le plus populaire des langages de programmation. Il est presque universellement utilisé en calcul numérique. Monopole d'IBM jusqu'en 1961, on le trouve maintenant dans tous les systèmes d'exploitation.

ALGOL

Pères : John Backus, Mc Carthy, Perlis, Bauer, Naur, Vauquois et d'autres...

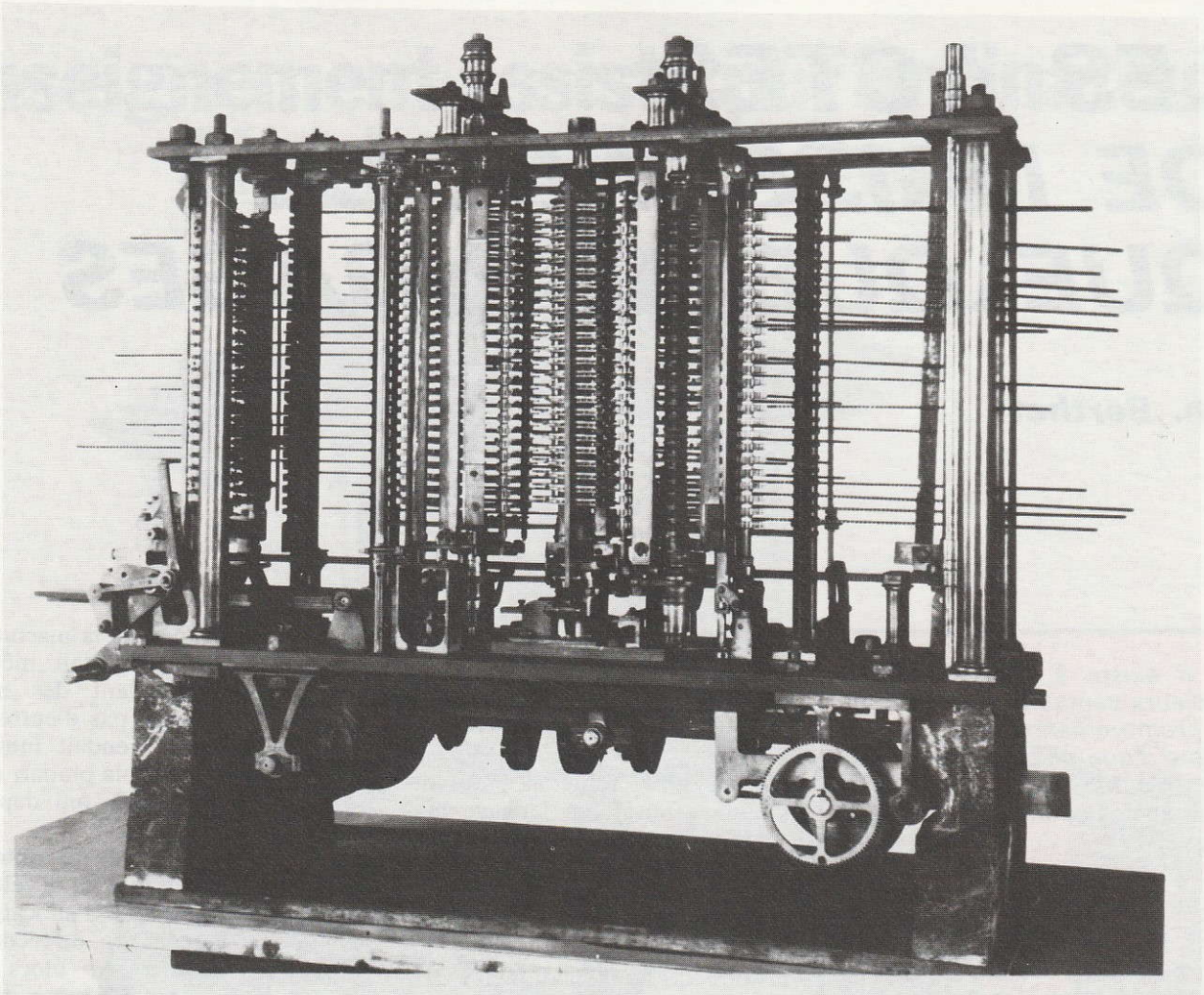
Le langage ALGOL est né d'efforts américains et européens en réaction contre FORTRAN, coupable à la fois d'être trop lié à la technologie d'un constructeur, et de manquer des qualités académiques les plus élémentaires : simplicité, régularité, commodité d'emploi, indépendance de la machine.

Le langage initial était même tellement indépendant des machines, qu'il n'avait pas d'entrées-sorties, ce qui le rendait inutilisable. Bien entendu, la plupart des réalisations d'ALGOL ont depuis corrigé ce défaut.

Il reste que les travaux autour d'ALGOL ont conduit au développement des concepts d'allocation dynamique de mémoire, de récursivité, de structure par blocs et procédures, qui ont été repris dans les langages modernes.

COBOL

La nécessité de répondre à l'immense marché des utilisations des ordinateurs en gestion, a très tôt poussé les constructeurs à développer des langages appropriés, très riches au niveau des spécifications d'entrées-sorties, mais assez limités en calcul numérique. On assistait donc, à la fin des années 50, à une floraison de langages voisins mais incompatibles entre eux : COMTRAN, FLOW-MATIC, FACT, AIMACO, etc. Le Ministère de la Défense Américain (D.O.D.), pour mettre de l'ordre dans cette BABEL, a imposé la définition d'un langage de gestion commun (ou COmmon Business Oriented Language), en abrégé, COBOL. On peut se demander si la structure très rigide et hiérarchisée de COBOL, le vocabulaire même du langage (Section, Division, etc.), ne viennent pas de ses origines militaires...



Partie analytique de la machine de Charles Babbage. Dès 1833 l'ingénieur anglais Charles Babbage proposait la constuction d'une mémoire mécanique destinée à conserver non seulement les nombres nécessaires au calcul, mais aussi les instructions demandées pour effectuer les calculs, ce qui était nouveau. La capacité était de 1 000 nombres, chacun des 50 chiffres décimaux. Photo IBM.

PL/1

C'est la nécessité d'adapter FORTRAN aux possibilités nouvelles de la technologie des années 60 qui a conduit le Share Fortran Committee à PL/1. Il est en effet apparu très vite qu'il fallait en réalité concevoir un nouveau langage et non pas simplement étendre FORTRAN. PL/1 associe ainsi les concepts les plus utiles d'ALGOL (structure générale du langage), de FORTRAN (formules, boucles itératives, fonctions et subroutines) et de COBOL (entrées/sorties), en ajoutant d'intéressantes spécifications de traitement des chaînes de caractères et quelques instructions nouvelles, particulièrement utiles (par exemple, l'instruction ON).

PL/1 est un succès ; les compilateurs actuels sont parfaitement au point et font maintenant de ce langage un outil supérieur à FORTRAN, ALGOL et COBOL.

BASIC, APL, LSE

BASIC, LSE et à sa façon, APL, sont des langages appropriés à cette manière de travailler.

- BASIC (1964 - Pères : Kemeny et Kurtz) est une création conjointe de Dartmouth College et de General Electric.
- APL (1965 - Père : Ken Iverson d'IBM) est un langage tout-à-fait remarquable. Mais son extrême concision se révèle finalement un handicap : il est très difficile de relire un programme APL un tant soit peu complexe — c'est-à-dire servant tant soit peu à quelque chose, et qui ne soit pas le calcul de la douze millième décimale de Pi ou quelque algorithme inédit du jeu de Nim, encore que dans ce dernier cas pourquoi pas — et maintenir en état une bibliothèque de programmes APL est chose ardue.
- LSE (1970 - Pères : J. Hebenstreit et Y. Noyelle, de l'École Supérieure d'Électricité) a été défini à la demande du Ministère de l'Éducation Nationale Français. Il est destiné à l'enseignement et se révèle plutôt meilleur que BASIC comme langage pour microordinateur. C'est un langage puissant et facile à utiliser.

PASCAL

1970 — Père : M. Wirth

Ce langage est d'origine suisse. C'est un descendant direct d'AL-

GOL auquel on a ajouté un certain nombre de possibilités correspondant à la conception moderne de la programmation. Sa structure cependant interdit son utilisation en mode interactif.

SMALLTALK 80

1980 — Père : A. Kay ; Mère : A. Goldberg

Ce langage est un développement interne depuis 1970 à la XEROX CORPORATION et fera sa première apparition publique sur plusieurs microordinateurs en 1981.

Dérivé de LISP et de SIMULA (lui-même dérivé d'ALGOL), c'est un langage applicatif, donc de conception extrêmement originale, qui risque à terme, de bouleverser complètement nos actuelles techniques et méthodes de programmation.

ADA

ADA, ainsi nommé en mémoire de Charles Babbage, inventeur au XIX[e] siècle d'une des premières machines à calculer (l'amie de Babbage s'appelait Ada Lovelage) est le dernier-né. De conception française, il résulte d'un appel d'offres du ministère de la Défense Américain, encore une fois.

Il s'agissait, en s'inspirant de PL/1 et quelques autres, de concevoir un langage très sûr, lisible, et facilitant la maintenance, donc, au total, moins cher que les autres. On verra à l'usage, si ce pari aura été gagné.

En un quart de siècle, les ordinateurs ont connu une évolution technologique foudroyante. Il existe une aussi grande distance entre les machines actuelles et leurs ainées des années 50 qu'entre le supersonique « Concorde » et le Bréguet « Point d'interrogation » de Costes et Bellonte de 1930.

Parallèlement, de même que les règles de base du pilotage des avions sont restées identiques, celles de la programmation des ordinateurs n'ont guère varié. Les langages ont très peu évolué ces vingt dernières années et la programmation réelle reste en définitive assez différente d'un secteur d'application à l'autre.

Il n'y a pas en effet d'informatique spécifique d'un domaine déterminé. L'informatique « de gestion » n'existe pas plus que l'informatique « médicale », « documentaire » ou « juridique ». Les outils sont les mêmes et la science qui les a créés est unique, autonome et originale.

Bien programmer implique d'abord quelque chose à dire, et dès lors, de bien connaître son domaine d'application. Cela implique aussi la maîtrise de la technique informatique, qui, malgré la commodité d'utilisation des langages actuels reste complexe. C'est pourquoi la programmation demeure — et demeurera longtemps encore — une affaire de professionnels.

Ch. Berthet

2 programmes en 5 langages

F.-M. Blondel

1/ Calcul de factorielle de n

Nous avons choisi un exemple de programme très court : le calcul de la factorielle d'un nombre entier n notée n !, est définie par le produit :

n x (n−1) x (n−2) x ... x 3 x 2 x 1

Comme ce calcul peut être utilisé à plusieurs endroits d'un programme donné, il est préférable de le définir comme un sous-programme (ou une procédure, ou une fonction suivant la terminologie du langage utilisé).

1 *Sous-programme en BASIC*

```
100 REM FACTORIELLE DE N
110 F=N
120 N=N-1
130 IF N=1 THEN 160
140 F=F*N
150 GOTO 120
160 RETURN
```

2 *Procédure en LSE*

```
100 PROCEDURE &FACT (N) LOCAL N
110 RESULTAT SI N=1 ALORS 1 SINON N*FACT(N-1)
```

3 *Fonction en APL*

```
∇ R ← FACT N
[1] → ZERO X2 N = 0
[2] R ← N X FACT N−1
[3] → 0
[4] ∇ ZERO = R ← 1
```

Notons toutefois que cette définition est inutile en APL, car il existe un opérateur factorielle, ce qui s'écrirait : !N.

4 *Fonction en LISP*

```
( DE FACT (N)
    ( COND
        ( ( ZEROP N) 1
        ( T  ( TIMES  N ( FACT (SUB1 N ))))))
```

5 *Fonction en PASCAL*

```
FUNCTION FACT  (N: INTEGER): INTEGER;
BEGIN
IF N=1 THEN FACT:=1
.      ELSE FACT:=N*FACT(N-1)
END;
```

Nota : Parmi ces différentes rédactions, seul le sous-programme en BASIC se présente de façon un peu différente. En effet, les autres langages permettent la récursivité, c'est-à-dire la possibilité pour une fonction de s'appeler elle-même dans sa définition.

2/ Pour parler « Verlan »...

Voici un autre exemple de programme simple : l'inversion d'un texte. Il a été écrit dans différents langages sous la forme d'un sous-programme :

1 *Sous-programme en BASIC*

```
100  REM  INVERSION DE LA CHAINE C$
101  REM  AVEC RESULTAT DANS R$
102  R$=""
103  FOR I=1 TO LEN(C$)
104  R$=MID$(C$,I,1)!R$
105  NEXT I
106  RETURN
```

2 *Procédure en LSE*

```
100  * INVERSION DE LA CHAINE TEXTE
101  PROCEDURE &INV(TEXTE) LOCAL INVER
102  CHAINE INVER ; INVER=''
103  FAIRE 105 POUR I = 1 JUSQUA LGR(TEXTE)
104  INVER = SCH(TEXTE,I,1)!INVER
105
106  RESULTAT INVER
```

3 *Fonction APL*

Notre programmeur APL est parti en vacances. Il nous a signalé l'existence d'une fonction du langage qui correspond à nos besoins. Il s'agit de la notation 0. Pour obtenir l'inverse de TEXTE, écrire 0 TEXTE.

4 *Fonction LISP*

```
(DE  REVERSE (L1  L2)
     (IF  (NULL L1)  L2
          (REVERSE (CDR L1) (CONS (CAR L1)L2))))
```

Cette fonction lisp est définie pour une liste quelconque. Son application à une chaîne de caractères se fera au moment de l'appel, en transformant la chaîne en une liste, puis le résultat en une chaîne :

```
(IMPLODE (REVERSE (EXPLODE  TEXTE)))
```

5 *Fonction en PASCAL*

```
FONCTION  INV (C:STRING) : STRING;

VAR  R:STRING;
     I:INTEGER;

  BEGIN
  R :=' '
  FOR I :=1 TO LENGTH(C) DO
      R :=CONCAT( COPY (C,I,1)R)
INV:=R
END;
```

Les langages applicatifs le LISP

H. Bestougeff

Les exemples donnés précédemment montrent à quel point l'écriture des programmes en LISP est différente.

H. Bestougeff tente ici de vous introduire dans ce monde nouveau des langages « applicatifs », une première exploration qui vous demandera, sans nul doute, un effort continu.

Les langages tels qu'ALGOL, PASCAL, BASIC, LSE, appartiennent à une famille qui se caractérise par la notion d'*instruction*. Un programme écrit dans un de ces langages est une suite d'ordres (instructions) qui décrit une séquence de transformations d'une situation initiale à une situation finale.

De façon simple, on peut dire qu'une situation est définie à chaque instant par l'ensemble des valeurs des variables du programme et que ce sont les *instructions d'affectation* qui vont modifier une situation.

Il s'ensuit que l'utilisation de tels langages oblige à une analyse détaillée de la suite des situations désirées. Les transformations précisées par les instructions, réalisent le passage d'une situation à la suivante, d'où le conseil souvent donné au débutant de « faire tourner son programme à la main » pour vérifier pas à pas, à partir d'une situation initiale, que les transformations choisies conduisent au résultat cherché, en suivant l'évolution des valeurs des différentes variables.

A ce niveau, on s'attache plus à la méthode pour obtenir le résultat que le résultat lui-même. Si le programme est complexe, il est souvent difficile, lors d'une simple lecture, de voir ce qu'il est supposé faire, *car la véritable compréhension passe par l'explicitation des situations obtenues et non par l'examen individuel des instructions.*

Les langages applicatifs sont au contraire construits de façon à pouvoir définir le résultat cherché sans se préoccuper a priori, de la manière dont ce résultat sera calculé. Ils sont fondés sur la notion de *fonction*. Il s'agit alors de définir des fonctions qui, appliquées à des arguments, fourniront le résultat cherché. Il n'y a plus à ce moment, ni instructions d'affectation, ni suites d'instructions. Un « programme », dans un langage applicatif, est une *expression* qui a une structure générale de type *opérateur, opérande* où l'opérateur représente la fonction et l'opérande, la valeur de l'argument. L'exécution d'un programme correspond à l'évaluation de l'expression.

Il est clair que l'opérateur et l'opérande peuvent être eux-mêmes des expressions, ce qui permet une construction hiérarchique d'expressions complexes. Le point important est que la valeur d'une expression est déterminée uniquement par la valeur des parties constituantes. Ainsi, si une même expression apparaît deux fois dans le même contexte, elle a la même valeur pour les deux occurrences ; de ce fait, un programme complexe peut être décomposé en expressions plus simples directement compréhensibles.

Nous allons maintenant examiner plus en détail la façon de définir les fonctions et de construire les expressions.

Une fonction f est définie lorsqu'on se donne un ensemble source A, un ensemble but B et une règle qui associe à chaque élément de A un élément de B appelé valeur de la fonction f. Ceci peut être noté par le schéma suivant : $A \xrightarrow{f} B$. Par exemple, l'addition de deux nombres entiers est une fonction de nom « add » qui, à un couple de valeur N x N, associe une valeur de l'ensemble N :

$$N \times N \xRightarrow{add} N$$

La façon d'opérer cette association n'est pas explicitée ici. Seuls sont spécifiés les ensembles source et but et le nom.

De manière générale, si x est un élément de A et y un élément de B, on peut dire que y est le résultat de l'application de f à x, que l'on écrit f(x) ou fx.

f carré

Nous venons ainsi d'écrire une expression simple, composée d'une partie opérateur et d'une partie opérande. Par extension, afin d'indiquer le résultat d'une application, on construit une expression à partir

de deux expressions (entourées ou non de parenthèses), en mettant la partie opérateur à gauche et la partie opérande à droite. L'expression résultante est dite *composée par application* (d'où le nom de langage applicatif).

Ainsi les expressions arithmétiques peuvent être écrites sous la forme opérateur-opérande :

ⓐ $(x) + \frac{y \times z}{x + z}$

ⓑ add (x, div (mult (y, z), add (x, z)))

L'expression ⓑ est une expression complexe qui se décompose d'abord en opérateur add et opérande (x, div (mult (y, z), add (x, z)).

La partie opérande est de nouveau décomposable jusqu'au niveau d'une expression simple.

Cette expression complexe peut être vue comme une fonction de x, y, z, construite à partir de fonctions élémentaires connues en utilisant la *composition* de fonctions.

Cependant, la composition n'est pas suffisante pour obtenir toutes les « fonctions intéressantes » et nous devons introduire deux autres mécanismes de définition de nouvelles fonctions à partir de fonctions primitives ou de fonctions précédemment définies : ce sont l'expression conditionnelle et la récursivité.

Mais auparavant, nous allons faire quelques remarques utiles pour la suite :

— Si, dans la définition d'une fonction, l'ensemble but est réduit aux valeurs de vérité, {vrai, faux} : $a \rightarrow$ { vrai, faux } nous dirons que f est un *prédicat*

L'écriture $f \in (A \rightarrow B)$ signifie que f est un élément de *l'ensemble des fonctions* qui, appliqué à un élément de A, donne un élément de B.

Puisque $(A \rightarrow B)$ désigne un ensemble (de fonctions), on peut essayer de définir des fonctions (dites d'ordre supérieur) qui ont pour ensemble source et but, un ensemble de fonctions. Le résultat de l'évaluation est alors *une fonction*. C'est là une des puissances du langage applicatif.

Exemple : Soit la fonction « élever au carré », notée $N \overset{carré}{\Rightarrow} N$, nous pouvons définir une fonction « élever à la puissance quatrième » à partir d'une fonction g telle que : $(N \rightarrow N) \xrightarrow{g} (N \rightarrow N)$, qui, en particulier, transforme la fonction « carré » en fonction « quatre » ;
g (carré) = puissance quatrième.

Expression conditionnelle

Pour la définir, prenons un exemple simple, celui de la fonction valeur absolue. On rappelle que :

$$x = \begin{cases} x & \text{si } x \rangle 0 \\ -x & \text{si } x \langle 0 \end{cases}$$

ce que nous noterons

abs(x) = (neg (x) → moins (x),x)

où neg est un prédicat ayant pour valeur « VRAI » si $x \langle 0$ et « FAUX » dans l'autre cas.

La forme générale de l'expression conditionnelle est donc :
$(P_1 \rightarrow e_1, P_2 \rightarrow e_2 \ldots e_n)$
ce qui peut se traduire par :
Valeur de l'expression = si p_1 alors e_1
Sinon P_2 alors e_2... etc.
ce qui est différent du classique IF...THEN...ELSE, instruction conditionnelle (et non une expression) déclanchant une transformation de la situation.

La récursivité

La définition récursive d'une fonction se caractérise de façon externe, par le fait que le nom de la fonction apparaît dans l'expression même de définition, mais avec une forme d'arguments différente. Ce phénomène est analogue à celui des procédures récursives mais il faut bien souligner que les procédures ne sont pas en général des fonctions (même si certains langages permettent la définition de procédures particulières de types de fonctions). Le rôle que jouent les variables locales et globales est déterminant. Dans les langages applicatifs, la notion de variable n'est pas la même. Les variables ne changent pas de valeur dans le temps ; elles ont, comme en mathématiques d'une certaine façon, une valeur unique pour chaque évaluation d'une expression.

Prenons comme exemple, le calcul du PGCD de deux nombres positifs m et n par l'algorithme d'Euclide, en donnant d'une part, l'expression d'une fonction récursive, et d'autre part, une procédure non récursive en LSE.

a) PGCD (m, n).

(zéro (res (m, n) → n, PGCD (n, res (m, n))

Zéro est un produit et **res** une fonction qui s'interprètent comme

zéro (x) x égal à zéro

res (x, y) reste de la division de x par y

b) 100 PROCÉDURE PGCD (M, N, P) LOCAL R

110 R ← M − N * ENT (M/N)

120 SI R = O ALORS ALLER EN 140

130 M ← N ; N ← R ; *ALLER EN 110*

140 P ← N ; RETOUR

X	CAR X	CDR X	Y	CONS (X Y)	ATOM (CAR X)	EQ X (CAR X)
(A B)	A	B	B	((A B) B)	T	F
(A B C)	A	(B C)	((D E))	((A B C) (D E))	T	F
(A B) (C D))	(A B)	((C D))	((A))	(((A B) (C D)) (A))	F	Indéfini

REMARQUE : La fonction EQ n'est pas définie si au moins un des deux arguments n'est pas un atome. La définition de nouvelles fonctions à partir de ces deux primitives utilise les mécanismes introduits précédemment, à savoir : la composition, l'expression conditionnelle et la récursivité. Cependant, les formes syntaxiques qui expriment les mécanismes en LISP sont définies de façon à être également des S-expressions (ou liste). Cette approche permet d'avoir une fonction d'évaluation unique qui lance l'évaluation de toutes les expressions.

Le langage LISP

Afin d'illustrer les notions un peu abstraites évoquées précédemment, voyons quelques aspects du langage LISP introduit par Mac Carthy en 1960. Notons que, dans les versions actuelles, le LISP n'est pas purement applicatif.

Dans tous les exemples pris jusqu'à maintenant, le domaine des fonctions était celui des nombres entiers (noté N). Or, nous avons précisé que les ensembles, source et but des fonctions, pouvaient être quelconques, et donc en particulier, des ensembles d'objets structurés, (tableaux, chaînes, etc.) que l'on manipulerait globalement.

Les fonctions LISP utilisent un domaine particulier qui est celui de *l'Ensemble des S-expressions* (« S » signifiant « symbolique ».)

Comment est construit cet ensemble ?

L'élément de base est *l'atome*, qui est considéré comme invisible. Nous ne considérerons ici que les atomes dits symboliques, qui sont, soit des symboles isolés (lettre notamment), soit des chaînes de caractères : A, B, LANGAGE, INFORMATIQUE, ÉDUCATION...

L'ensemble des S-expressions est alors défini comme suit :

— Un atome est une S-expression.

— Une séquence d'S-expressions entourée de parenthèses est une S-expression (cette séquence est également appelée liste).

Exemple de S-expressions :

(ÉDUCATION ET INFORMATIQUE) Séquence de 3 atomes

(LA REVUE) (ÉDUCATION ET INFORMATIQUE)
Séquence de 2 S-expressions
Séquence de S-expressions

? ( )

→ NIL (NIL est une liste).

Pour manipuler ces S-expressions, on introduit 5 fonctions qui ont pour ensemble source et but, des ensembles S-expressions. Elles sont notées CAR, CDR, CONS, ATOM, EQ.

Intuitivement (voir tableau), CAR donne la tête d'une liste, CDR, une liste où la première S-expression de la liste donnée est omise, CONS construit une liste à partir de deux listes, ATOM et EQ sont deux prédicats définis ainsi :

ATOM X est vrai, si et seulement si X est un atome
→ EQ X Y est vrai si et seulement si X et Y sont deux atomes identiques.

En résumé, pour qualifier les langages applicatifs, on peut insister sur trois points :

*1 — **L'aspect « constructif »***
Les nouvelles fonctions sont construites au fur et à mesure, à partir des fonctions primitives. Ceci implique un choix judicieux de ces fonctions primitives pour chaque application. En effet, si la composition et l'expression conditionnelle sont faciles à manipuler pour les néophytes, il faut admettre que la dextérité, en ce qui concerne la récursivité, ne s'acquiert pas aussi rapidement. Il faut donc être capable de créer, dans un domaine donné, des langages applicatifs qui possèdent des fonctions de base telles que l'appel à la récursivité ne soit pas nécessaire pour les cas usuels.

2 — La facilité de désigner et de manipuler des objets complexes globalement, liée à la puissance de la définition des fonctions (notamment celles d'ordre supérieur).

3 — La lisibilité des programmes réalisant des traitements importants due au fait que la signification d'une partie d'un programme est indépendante de la signification de l'ensemble.

L'inconvénient majeur concerne le manque de performances et l'encombrement mémoire de ce type de programmes. La taille, la puissance et le coût élevé des machines expliquent, au niveau historique, le faible développement de ce type de langages. Mais, à l'heure actuelle où le prix de l'écriture d'une instruction reviendra bientôt aussi cher que l'ensemble du matériel, l'enjeu est différent et l'objectif est de s'orienter vers des outils qui rendent les utilisateurs « performants ».

H. Bestougeff

EXEMPLE : La fonction EQ s'intéresse à l'identité de deux atomes, la définition LISP suivante introduit une fonction qui teste l'égalité de deux listes quelconques.

DÉFINITION D'UNE FONCTION EN LISP

```
(EQUAL (LAMBDA (X Y)
(COND

((ATOM X) (EQ X Y))

((ATOM Y») NIL)

(EQUAL (CAR X) (CAR Y))
(EQUAL

(CDR X) (CAR Y)) (T NIL))))
```

COMMENTAIRES

EQUAL est le nom de la fonction LAMBDA. (X Y) indique que l'on va définir une fonction de deux listes X et Y.
L'expression conditionnelle est notée :
(COND (X1 Y1) (X2 Y2)... (T YN)).
Dans chaque parenthèse, la première liste telle que X1, est une expression propositionnelle qui a pour valeur « vrai » (noté T) ou « faux » (noté NIL). La deuxième liste est une expression quelconque. On remarquera que, dans la dernière parenthèse, la première liste est remplacée par la valeur T.

Un descendant de LISP aux multiples possibilités éducatives : LOGO

F.-M. Blondel

Appartenant à la famille de LISP, le langage LOGO est une des applications éducatives des idées de l'intelligence artificielle. Conçu par ses auteurs dont le principal est Seymour Papert, pour fournir aux enfants les moyens de résoudre eux-mêmes des problèmes avec l'ordinateur, il reprend une bonne partie des caractéristiques de LISP tout en adoptant une syntaxe et un vocabulaire qui le rendent plus proche du langage naturel. Son emploi est prévu en conjonction avec des matériels originaux dont le principal est la « tortue » : un véhicule commandé par ordinateur, et qui peut se déplacer en avant ou en arrière, pivoter à droite ou à gauche, laisser une trace au sol grâce à un stylo, et savoir s'il a rencontré un obstacle grâce à ses palpeurs.

La programmation en LOGO se fait en définissant des procédures à partir des « primitives » du langage. Les procédures ainsi définies deviennent alors de nouvelles primitives. Voici par exemple une procédure qui permet de dessiner un triangle avec la « tortue » :

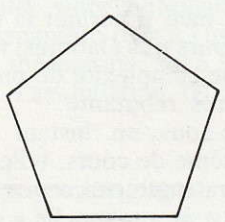
```
POUR TRIANGLE
1  AVANCE  100
2  DROITE  120
3  AVANCE  100
4  DROITE  120
5  AVANCE  100
FIN
```

et une procédure qui permet de dessiner tous les polygones réguliers :

```
POUR  POLY  :COTE  :ANGLE
1  AVANCE   :COTE
2  DROITE   :ANGLE
3  POLY     :COTE  :ANGLE
FIN
```

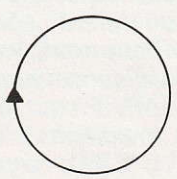
Pour dessiner un pentagone, il suffit d'écrire :

```
POLY  100  72
```



Pour dessiner un cercle, limite de polygone, il suffit d'écrire :

```
POLY  1 1
```



Écrite de cette manière, la procédure POLY ne s'arrête pas. Il faudrait lui ajouter un test sur le nombre de côtés dessinés ou sur la quantité totale de tours effectués par la tortue.

Les langages pour les enseignants

Maurice Peuchot

L'Enseignement Assisté par Ordinateur est né en grande partie des théories de l'enseignement programmé des psychologues SKINNER, prônant un découpage de tout apprentissage en unités d'informations (items), s'enchaînant de façon séquentielle, et CROWDER, qui introduisit des graphes plus élaborés dans lesquels le cheminement de l'élève dépendait des réponses qu'il fournissait.
Progressivement, la notion d'EAO a dépassé ce strict cadre. Les informaticiens ont mis au point des « langages » destinés à faciliter le travail de l'enseignant pour l'élaboration des programmes. Quels sont ces langages ? Que permettent-ils de faire ?
Maurice PEUCHOT est une des personnes qui a le plus travaillé sur ce sujet. Auteur, en particulier d'une méthode, la Conception Modulaire pour l'Enseignement Assisté par Ordinateur (CMEAO), il nous donne ici un aperçu des langages existants, de leurs possibilités et des développements qu'ils devraient connaître. Bien évidemment, les outils dont on dispose pour s'exprimer, influent sur le contenu des messages ; les langages d'EAO, avec les contraintes qu'ils imposent, n'échappent pas à cette loi. Aussi est-ce à une réflexion plus générale sur l'EAO que nous invite ici l'auteur.

Il y a déjà une vingtaine d'années qu'est née l'idée d'utiliser l'ordinateur aux fins d'enseigner. Le projet était à cette époque d'autant plus ambitieux que l'état de l'informatique ne s'y prêtait guère : très peu de possibilités matérielles, en particulier des « terminaux » mal adaptés à ce genre de travail. Quant aux langages de programmation eux-mêmes, leur maîtrise faisait, davantage à cette époque que maintenant, l'objet d'une spécialité rare. En d'autres termes, si le concept d'EAO était né, il restait à définir l'ensemble des moyens à mettre en œuvre afin de le matérialiser.
On sait qu'au cours de ces années, la réponse trouvée à certaines questions fit immédiatement surgir de nouveaux problèmes, *tant il est naturel que chaque perfectionnement, loin de satisfaire l'utilisateur, le rend plus exigeant.*
De plus, certaines de ces exigeances sont apparemment contradictoires. Par exemple, il importe de pouvoir disposer d'un clavier riche, utilisant une typographie classique, mais ce clavier riche est lui-même d'emploi malaisé, principalement pour l'élève qui n'est pas habituellement un dactylographe expérimenté... Le cas des langages d'EAO constitue un autre exemple d'exigence contradictoire : ou bien un tel langage est simple, auquel cas il est vite maîtrisé mais aux dépens de ses possibilités pédagogiques, ou bien il permet la réalisation de cours très élaborés, mais au prix d'une complexité de programmation très rebutante.
Arrêtons-nous un instant sur la notion même de cours, telle quelle est généralemnt perçue en EAO. Trop souvent, on appelle « cours » une succession « d'écrans » à la mise en page plus ou moins heureuse, aux croquis plus ou moins approximatifs, à la typographie plus ou moins fidèle. De temps à autre, une question simple est posée à l'élève. Il est aisé de découvrir pour de tels cours, l'option pédagogique sous-jacente : l'enseignement programmé de type Skinnerien ou Crowderien. De plus, et quand bien même cette approche pédagogique ne serait-elle pas du goût de l'auteur du cours, la nature même du langage de programmation utilisé infléchit la démarche pédagogique plus ou moins vers le behaviorisme.
Cela est essentiellement dû à la notion même d'exercice en EAO classique. Nous développerons plus loin cet important aspect des choses. Enfin, nous ne justifierons pas la nécessité d'inventer des langages de programmation dévolus à l'EAO. Ils ont été créés en vue d'une application de l'informatique bien précise, et les raisons en sont connues.

Caractères communs aux langages d'EAO

Il existe un grand nombre de langages d'EAO. Cela tient probablement au fait que là où l'enseignant espère une aide précieuse dans l'exercice de son métier, d'autres, en créant des langages, imaginent la nature de l'aide que l'EAO apporte à l'enseignant, sans pour autant appréhender nécessairement les besoins de l'élève... Mais, quel que soit le nombre de ces langages, il existe un noyau commun, imposé par la nature même de cette application. Tout langage d'EAO doit au moins permettre, de présenter de l'information pédagogique à l'élève, d'analyser ses réponses, d'accéder à un enchaînement de notions dépendant de son comportement, de comptabiliser en vue de statistiques, ses succès et échecs.

Chaque langage d'EAO offre en effet ces possibilités. Certains se distinguent des autres par des possibilités supplémentaires : graphiques, jeux de caractères spéciaux, couleur, son, couplage avec un projecteur de diapositives, un magnétophone ou un magnétoscope. Peu d'entre eux par contre, comportent les instructions arithmétiques nécessaires à la simulation. Bien entendu, chaque langage a son propre jeu d'instructions permettant de rédiger des programmes pour l'ordinateur. En effet, et bien qu'il soit fréquent d'entendre proclamer que l'emploi de ces langages ne constitue pas une activité de programmation, il n'est que de lire les programmes écrits dans ces langages pour être convaincu que ce sont bel et bien d'authentiques langages de programmation dont la maîtrise requiert autant d'efforts qu'en nécessite la pratique de FORTRAN ou de COBOL.

Ces langages ont de plus une faiblesse. Elle se situe au niveau des exercices, lesquels sont, vus de l'élève, peu motivants. *Car, ce qui motive l'élève, ce n'est pas ce qui se passe dans la machine mais ce qui lui est permis de faire.*

Certes, certains exercices sont très spectaculaires, mais la participation directe, active de l'élève y est très faible. On justifie souvent l'EAO par des arguments du genre : l'élève travaille à son propre rythme, il n'est pas assujetti à un horaire ni à un lieu précis d'étude (décentralisation des terminaux). Sans doute ! Mais on pourrait en dire tout autant du livre !...

Or, ce qui distingue l'EAO de l'enseignement livresque est précisément *l'interactivité*. Si un élève commet une erreur, le livre ne réagit pas ; l'ordinateur dès lors qu'il est correctement programmé, si ! Et c'est bien dans les exercices que se manifeste le plus le caractère nécessairement interactif de l'enseignement. Paradoxalement, il semble qu'on ait négligé les exercices en EAO...

Ainsi, on dit fréquemment que l'erreur est source d'enrichissement. C'est vrai, à condition que l'erreur soit détectée certes, mais aussi commentée en fonction de sa cause, laquelle doit être identifiée. Il a été donné à l'auteur d'assister à des démonstrations époustouflantes mais pour lesquelles le rôle de l'élève confinait à une activité de presse-bouton. Certes, apprendre dans certaines circonstances à appuyer sur le bouton adéquat peut être utile, voire nécessaire, mais cela s'apparente plus à l'imprégnation de mécanismes qu'à l'acquisition et l'assimilation de connaissances.

• *Principales instructions PILOT*

Le code opération utilisé pour présenter une ligne de texte est : T:
Par exemple : T: BONJOUR
A l'exécution, le mot BONJOUR apparaîtra sur l'écran.
Si S contient Pierre, alors :
T: BONJOUR S, COMMENT VAS-TU ? fera apparaître sur l'écran : BONJOUR PIERRE, COMMENT VAS-TU ?
L'instruction T peut-être conditionnelle : T (N > 3) : BRAVO ! provoquera l'affichage du message BRAVO ! si et seulement si la variable N est égale, à l'instant où cette commande est exécutée, à au moins 4.
Si l'argument de T: n'est pas une information mais une question, alors il faut autoriser l'élève à répondre et à saisir sa réponse. Cela est réalisé par l'instruction A : ACCEPT. A n'a pas d'argument en général.
L'ordinateur compare la réponse de l'élève au moyen de l'instruction M: (MATCH) dont l'argument constitue la bonne réponse.
Exemple :
T: 2 + 2 = ?
A:
M: 4
TY: BIEN
JY: SUITE
TN: NON RECOMMENCEZ
JN: 9 A
** SUITE*
Etc.
TY: est une instruction conditionnelle. Si la réponse de l'élève coincide avec 4 (YES), alors « BIEN » est affiché, puis un saut (JUMP) est effectué à l'étiquette SUITE. J est elle-même conditionnelle.
TN: signifie : imprime « NON, RECOMMENCEZ » si la réponse de l'élève n'a pas coincidé (NO) avec 4.
On pourrait admettre la réponse « QUATRE »
M: 4 ! QUATRE « ! » signifiant « OU »
De même le symbole & signifie « ET »
JN: A signifie : Si le résultat de la comparaison en M: 4 est NON, revenir à A : (pour attendre une autre réponse de l'élève).
En plus de ces instructions, il en existe d'autres telles que :
C: (Fonction de calcul)
U: (Appel d'un sous-programme)
L: (Enchaînement de leçons)
W: (Pause)
D: (Dimension : au sens FORTRAN)
X: (Exécution d'un choix de caractères, considéré comme une instruction PILOT très puissante)
Etc.

Principaux langages

On sait que presque tous les grands constructeurs ont, soit créé leurs propres systèmes d'EAO, soit les ont fait développer par des universités. Les systèmes ASET, IIS, PLATO, etc., sont aujourd'hui bien connus ainsi que les langages d'EAO comme COURSEWRITER (IBM) ou TUTOR (CDC). ASET a été développé par UNIVAC, IIS par IBM et PLATO par l'université de l'Illinois en collaboration avec Control Data. A côté de ces langages, l'auteur travaille pour sa part depuis quinze ans à un système d'EAO auquel la notion même de langage de programmation est absente, le CMEAO, dont

la description ici nous ferait sortir du sujet. Puisqu'il n'est pas possible de décrire, même brièvement, les qualités et défauts des systèmes énoncés ci-dessus, nous avons choisi un langage d'EAO moins connu mais qui, outre ses qualités propres et certaines, présente l'avantage d'apparaître presque comme la synthèse des principaux langages d'EAO : le système PILOT.
Le système PILOT fut développé en 1968 par le Dr. John A. STARKWEATHER et ses collaborateurs, à l'université de Californie (San Francisco). Chemin faisant, nous retrouverons dans PILOT, les qualités et défauts des systèmes susnommés. Afin d'éclairer le lecteur, faisons une brève incursion au sein de ce système :

Les exercices en EAO

Peu importe le travail de programmation. Ce qui importe avant tout, c'est ce que peut faire l'élève confronté à un exercice, et ce que cet exercice lui apporte effectivement.

Nous avons déjà évoqué l'enseignement programmé. Ce qui caractérise cette démarche pédagogique est le refus de confronter l'élève à des difficultés quelconques. On place l'élève sur des rails et il se laisse guider. De temps à autre, un exercice simple lui est proposé, non pour le faire réfléchir, mais pour renforcer la connaissance (ou l'automatisme ?) fraichement acquise. Si cette approche « anti-socratique » peut se justifier par la nature du support (le livre) et par l'impossibilité de procéder différement (pis-aller) il ne saurait être question de l'ériger en panacée. *Dès lors qu'un support nouveau est disponible, il importe de définir des approches nouvelles.* Simuler sur ordinateur des méthodes utilisées à l'aide d'autres moyens ne contribue qu'à faire jouer à l'ordinateur un rôle d'amplificateur des défauts de ces méthodes. Et puis, peut-on sérieusement envisager d'utiliser un ordinateur comme machine à tourner automatiquement les pages d'un livre, ce livre fut-il rédigé selon les méthodes de l'enseignement programmé ou fut-il illustré de la plus belle façon qui soit ?...
Pour en revenir aux langages de l'EAO, on peut dire qu'un tel procédé ne vaut que par son aptitude à définir des exercices pédagogiquement valables. Bien sûr, cette notion d'exercices pédagogiquement valables est affaire de goût personnel. Les inconditionnels de Skinner s'accommodent fort bien d'exercices pour lesquels l'élève doit répondre par OUI ou par NON, ou dans le meilleur des cas, par le choix d'une réponse parmi d'autres, proposées dans le cadre d'une Question à Choix Multiple (QCM)

Quand bien même une telle option pédagogique serait-elle parfois valable, il n'en demeure pas moins que *figer l'EAO dans cette voie étroite ne pourrait que provoquer la désaffection vis-à-vis d'un moyen d'enseignement riche en potentialités mais qui aurait déçu.* L'auteur sait fort bien que l'EAO autorise d'autres types d'exercices tels que ceux simulant des phénomènes de physique. Dans ce cas, la question posée à l'élève ne s'exprime plus uniquement par un message, mais est renforcée par un graphique, un schéma animé ou non.
Un fois de plus, concentrons notre attention sur l'élève. Que fait-il ? Que *peut-il* faire, confronté à un tel exercice ? Est-on certain de ne pas être amené de façon plus ou moins directe à une réponse ou à une cascade de réponses du type « VRAI/FAUX » ou à une Question à Choix Multiple ?

Bref, qu'offrent en réalité les langages d'EAO pour créer des exercices significatifs ? Examinez soigneusement leurs codes opérations disponibles. Vous serez admiratif pour ceux concernant la création de graphiques, de jeux de caractères, éventuellement d'effets sonores. Ou bien serez-vous intéressé par les possibilités d'obtenir des statistiques concernant le comportement de l'élève, ou bien encore par certaines activités pédagogiques de ces systèmes : dialogues entre utilisateurs via les terminaux ou autres malices cachées dans la machine ?

Il y a plus et mieux à faire. Par exemple, analyser une réponse d'élève en fonction du contexte particulier dans lequel cette réponse est formulée, et ce, dans le cadre d'un exercice particulier. *Ceci entraîne la possibilité d'analyser, plus qu'une simple réponse, un raisonnement tout entier.* Mais il est bien évident que pour en arriver là, il ne faut pas se contenter des langages d'EAO existants.

De plus, la notion d'exercice évoque des réponses. Et pourquoi pas des questions ? Dès lors que l'élève est autorisé à poser des questions, le dialogue devient plus vivant, l'élève se sent plus motivé et plus grand est son profit. On sait que tout cela est possible en CMEAO mais ce sera peut-être l'objet d'un autre article.

AUTRES CARACTERISTIQUES DE PILOT

- ***Instructions « graphiques »***
PILOT permet la création de graphiques de la façon la plus simple qui soit : en promenant le curseur sur l'écran. Son parcours détermine un graphique auquel on donne un nom. Il suffit de le mémoriser et de le rappeler par son nom au moyen de l'instruction GX : Si on a appelé « MAISON » un dessin représentant une maison, pour faire apparaître ce dessin sur l'écran, il suffit d'écrire dans le programme l'instruction :
GX : MAISON
Précisons que ces graphiques peuvent être en couleur (6 couleurs).

- ***Caractères spéciaux***
Chaque utilisateur peut se définir son (ou ses) propre(s) jeu(x) de caractères. Pour se faire, il suffit d'appeler l'éditeur de caractères. 35 matrices de points apparaissent sur l'écran, permettant ainsi de créer un alphabet de 35 lettres ou symboles. A chacun de ces symboles est affecté au choix de l'utilisateur, une touche du clavier. C'est ainsi qu'en quelques instants, il est possible de se doter de l'alphabet arabe, cyrillique, katakana, etc., ou bien d'enrichir l'alphabet mathématique habituellement trop pauvre sur les claviers usuels.

- ***Le son***
PILOT possède un autre jeu très complet d'instructions permettant de créer des effets sonores (alarme, mélodie). Mais d'un point de vue pédagogique, l'intérêt est moindre, sauf peut-être pour l'enseignement du solfège.

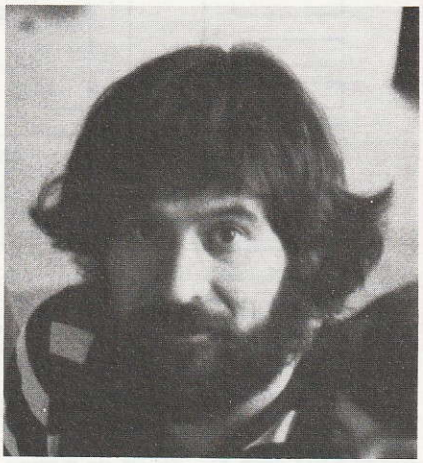
PRATIQUE DES LANGAGES D'EAO

Tout d'abord, une constatation : plus un langage d'EAO est riche et moins cette richesse est exploitée.

(Suite p. 33)

RUBRIQUE CLUB

La voiture a microprocesseurs



Alain Labreuille

Après avoir décrit les conditions dans lesquelles INFOCLUB du Lycée Technique DIDEROT avait mené à bien la réalisation de son bolide doté d'un microprocesseur 6802 (& I N° 6), nous avons demandé à Alain LABREUILLE, l'animateur de l'équipe, de décrire, une Alpha T33 de TURINS'CAR et les modifications qu'il y avait apportées.

UNE VOITURE ROBOT, MAIS LAQUELLE... ?

Même si certains avaient opté pour une solution analogique (et nous leur tirons le chapeau bien bas), la réalisation d'un véhicule robot à microprocesseur tenait suffisamment du « pari impossible » pour pousser les amateurs de microinformatique à relever le défi. Le club du lycée s'est donc procuré une carte microprocessur (le 6802, quel arôme !) pour lequel nous disposons d'un système de développement complet : éditeur/assembleur, prom-programmeur/débug. résident. Le matériel réuni, il ne manquait plus que le principal, des idées et du temps pour mettre en œuvre notre projet. Mais commençons par le commencement...

LA MÉCANIQUE

La modification majeure est l'utilisation de 2 moteurs (un pour chaque roue) mécaniquement indépendants, des MABUSHI 550 qui, alimentés en série, absorbent le même courant et développent donc le même couple moteur. Cette méthode (une idée de Yann, membre du club) assure le fonctionnement « différentiel du train arrière ».

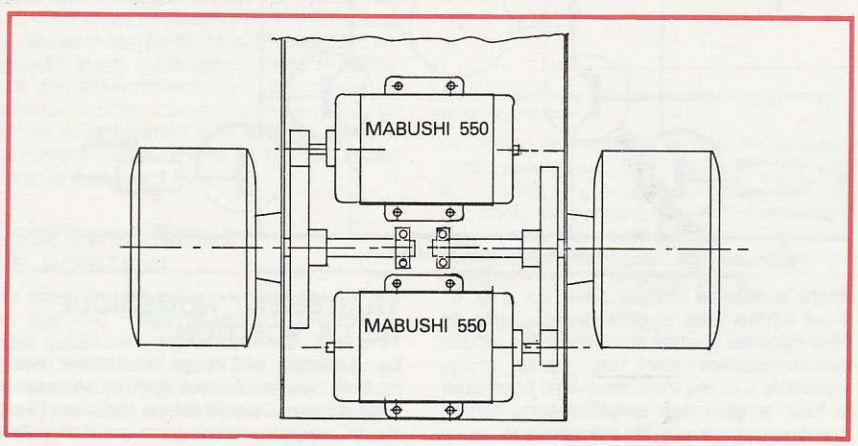


Fig. 1 — Train arrière à moteurs indépendants, reliés en série.

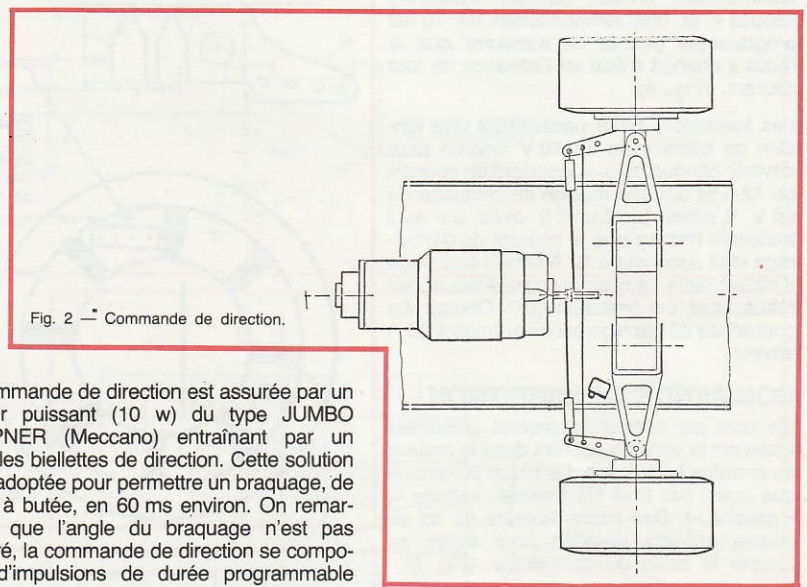


Fig. 2 — Commande de direction.

La commande de direction est assurée par un moteur puissant (10 w) du type JUMBO GRUPNER (Meccano) entraînant par un câble les biellettes de direction. Cette solution a été adoptée pour permettre un braquage, de butée à butée, en 60 ms environ. On remarquera que l'angle du braquage n'est pas mesuré, la commande de direction se composant d'impulsions de durée programmable dans un sens ou dans l'autre, à la manière d'un moteur pas-à-pas.
On remarquera sur le dessin les deux ressorts de rappel soudés au câble moteur diamètre 3 à 4 mm genre meccano (difficile à trouver) et des microrupteurs dont on verra l'action plus tard.

L'autre modification sur les commandes de direction fut d'augmenter l'angle de braquage maximum pour « passer » dans les épingles les plus serrées prévues au règlement (fig. 1 et 2).

CARTE MICROPROCESSEUR

Notre choix s'est porté sur le 6802 mais toutes les cartes sur le marché pourraient être utilisées (MK2, KIM, EMR, etc.).
On y trouve, outre le 6802, piloté par un quartz 4MHZ, 2K de mémoire vive (type 2114) pour stocker les informations, 2 supports REPROM (type 2716) permettant d'exploiter 4K de programme, un compteur programmable 6840 et 2 PIA 6821. Bref, du très classique (fig. 3).

COMMANDE DES MOTEURS DE PROPULSION

On remarquera la simplicité de l'interface de puissance (VN 64GA SILICONIX). Un relais assure la commutation AV/AR. L'interface communique avec le processeur par trois fils seulement (masse, « marche/arrêt » et « avant/arrière ») ATTENTION, pour éviter la destruction des contacts du relais, la

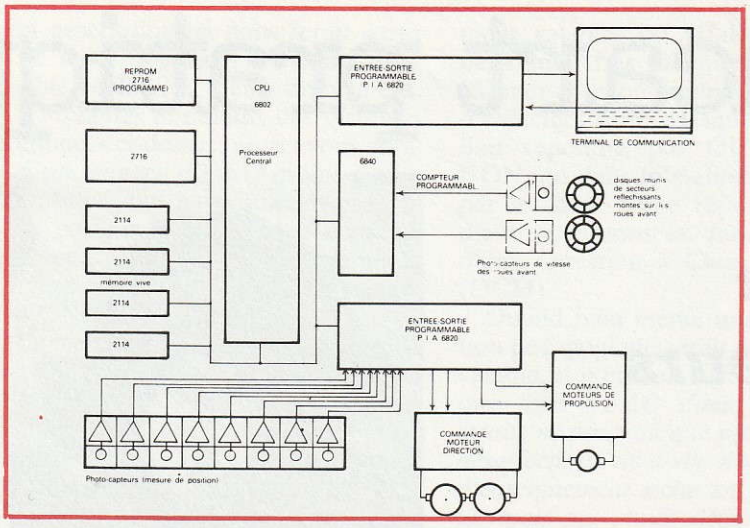


Fig. 3 — Carte microprocesseur et ses périphériques.

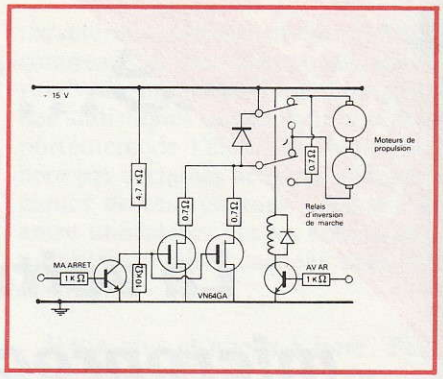


Fig. 4 — Commande des moteurs de propulsion.

Un des PIAS assure les communications avec le système de développement. Ceci permet de tester les programmes directement sur la voiture. Le microprocesseur communique avec des circuits de mesure (photocapteurs pour la position et la vitesse) ainsi que des circuits de commande de puissance (Commande moteurs de propulsion et direction).

commutation AV/AR se fait « moteurs coupés » et une temporisation de 10 ms programmée permet de s'assurer que le relais a changé d'état en l'absence de tout courant. (Fig. 4).

Les transistors MOS nécessitent une tension de commande de 10 V environ pour devenir conducteurs, supportent un courant de 12 A et ont une tension de claquage de 60 V. L'essai (destructif !) avec un seul transistor montra que le courant de démarrage était supérieur à 12 A et qu'il était sage d'utiliser deux transistors en parallèle et des résistances de limitation (0.7 Ohms). Le courant de démarrage est ainsi limité à 20 A environ.

COMMANDE DE DIRECTION

Un pont de transistors permet d'inverser aisément le sens du courant dans le moteur qu'entraîne la direction. Ce circuit communique aussi par trois fils (masse, « droite », « gauche »). Des micro-rupteurs de fin de course ont été installés pour éviter de rompre le cable de commande. (Fig. 5).

DIRECTION DE VITESSE, MESURE DU RAYON DE COURBURE DE LA PISTE

Chaque roue avant est munie d'un générateur d'impulsion. Le système permet de mesurer la vitesse du véhicule entre 1 cm/s et ...mach 2 ! (Fig. 6).

La mesure du rayon de courbure des virages est sensiblement proportionnelle à (V1 + V2)/(V1 − V2) (qui est la variation relative de la vitesse des roues internes et externes). En effet, chacun sait que dans un virage à droite, la roue intérieure parcourt moins de chemin que la roue extérieure, c'est-à-dire que pour des chemins comparables, elles ont des vitesses différentes. (Fig. 7).

DIRECTION DE POSITION

Comment savoir où est la piste ? Après bien des difficultés, nous avons choisi d'éclairer franchement la piste avec des ampoules VARTA alimentation 1,5 V. Le choix de huit capteurs convient à la structure du microprocesseur (8 bits) et l'électronique associée, un amplificateur opérationnel par phototransistor, permet de régler individuellement le gain de chaque détecteur (Fig. 8). Pour vérifier que la détection de piste se fera dans de bonnes conditions, les diodes luminescentes sont un « luxe indispensable » et œuvrent sans égal pour aider à fixer le gain des amplificateurs. Sinon, comment savoir que les informations transmises au microprocesseur correspondent bien à l'état de la piste ?

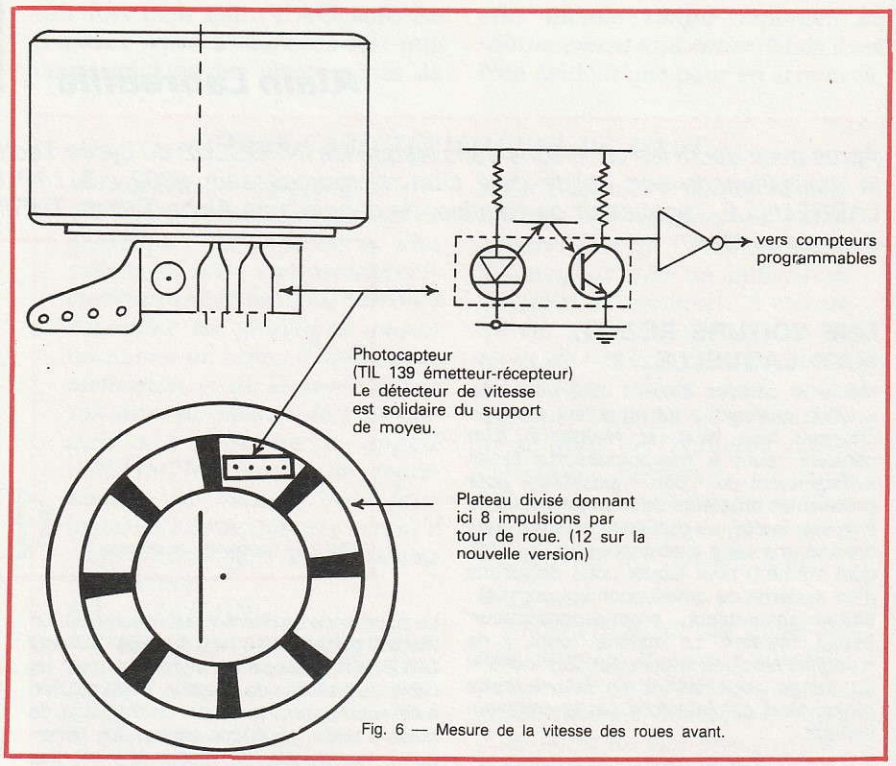


Fig. 6 — Mesure de la vitesse des roues avant.

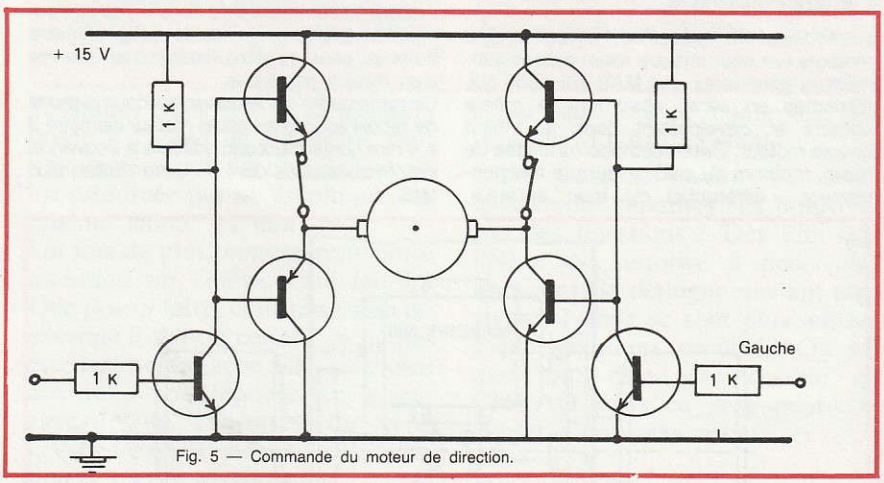


Fig. 5 — Commande du moteur de direction.

TRAITEMENT NUMERIQUE DE LA POSITION

Le traitement de l'image consiste à mettre en mémoire les formes lumineuses observées à intervalles de temps réguliers (5 ms sur la dernière version) ce qui donne une tâche oblongue dont un programme extrait l'élément intéressant, à savoir, la position

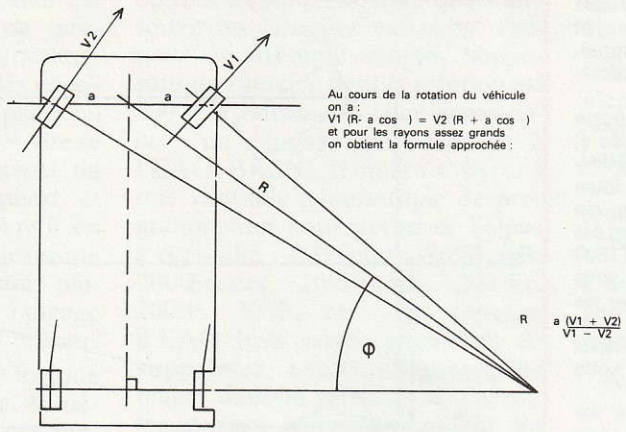


Fig. 7 — Mesure du rayon de courbure de la trajectoire.

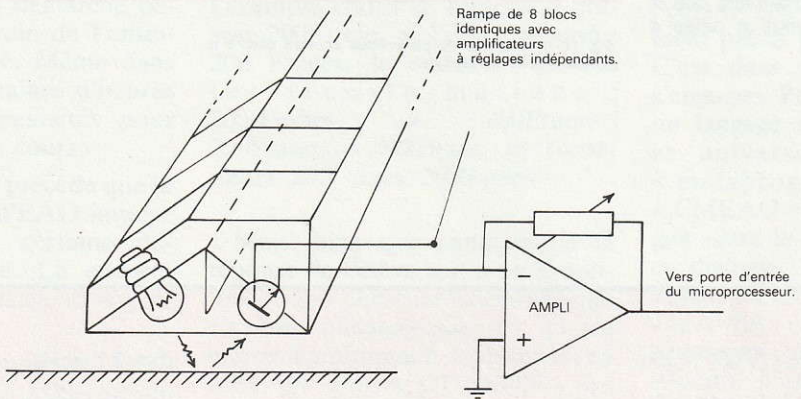


Fig. 8 — Rampe de photodétecteurs permettant de suivre la piste.

estimée du véhicule (Fig. 9). Pour les amateurs de mathématiques, nous utilisons l'intersection de l'axe principal d'inertie D avec l'axe OX. Pour les amateurs d'informatique, ce calcul est effectué en moins de 1ms à 1 % près et donne par la même occasion, l'angle de l'axe D par rapport à OY. La position ainsi estimée est utilisée par la suite dans les programmes d'asservissement.

ASSERVISSEMENT DE LA VITESSE

Nous l'avons choisi « proportionnel », c'est-à-dire que l'énergie fournie au moteur est proportionnelle à l'écart Vo — Vm (Vm étant la vitesse mesurée et Vo la vitesse désirée, appelée aussi « consigne » dans la littérature des asservissements). Une minute de réflexion montre que si Vo — Vm = 0, on fournit effectivement une énergie positive au moteur... dans l'espoir de corriger l'écart dans le bon sens !

ASSERVISSEMENT DE POSITION

Du point de vue théorique, le véhicule est un système « naturellement » instable. Il n'est cependant pas impossible de stabiliser le véhicule et pour être précis, il est nécessaire de définir correctement le système par des « variables » adaptées. Le choix de ces variables n'est pas unique, mais en voici un qui conduit à des résultats exploitables (fig. 10).

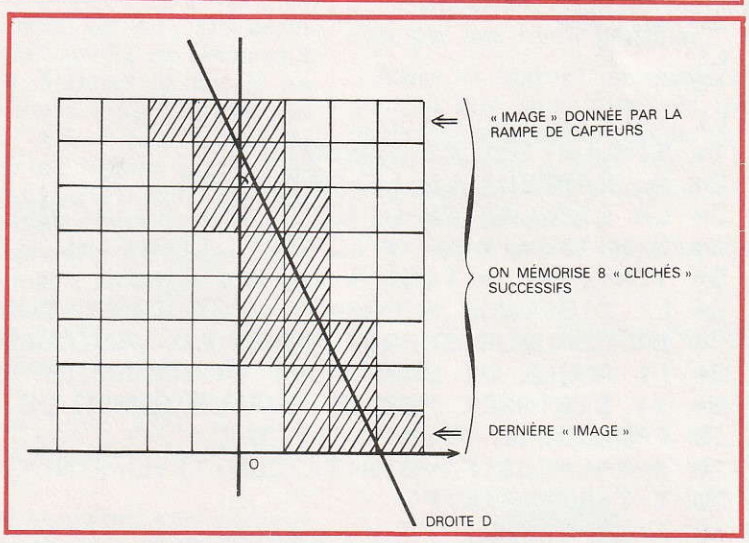


Fig. 9 — Utilisation de « l'image » pour estimer au mieux la position du véhicule.

l'évolution du système se déduit des positions successives du véhicule. Comme X_{n+1}, W_{n+1}, nouvelle position, à X_n et W_n, ancienne position, par le truchement de U_n, commande de direction.
Après quelques lignes de calcul et en simplifiant (W et U petits) on a :

$$X_{n+1} = X_n + 1.\ (W_n + U_n)$$

$$W_{n+1} = \dot{W}_n = \frac{1}{L}.\ U_n$$

le jeu consistant à définir U_n pour que la trajectoire de la voiture soit digne et majestueuse.

Une solution consiste à utiliser : $U_n = a.X_n + b.W_n$. Et a et b ? Comment les déterminer ?
Pour les débutants, voici une proposition : $U_n = — W_n — X_n/1$
On pourra vérifier que cette commande est composée de 2 termes dont la signification physique est simple, et que X_{n+1} devient nul ainsi que W_{n+p} où p n'est pas trop grand...

QUELQUES MOIS PLUS TARD

Voici donc un véhicule qui roule sans trop s'écarter du pointillé central. Il est même capable en roulant doucement, de compter

les pointillés, de mesurer la longueur des lignes droites, la courbure des virages... et finalement, de faire un relevé topographique du circuit et de conserver ces informations en mémoire.
Il reste au calculateur une tâche intéressante : celle de prévoir la vitesse et la position optimales permettant de minimiser le temps de parcours des deux tours chronométrés. Sans trop entrer dans les détails ces calculs sont faisables. Ils ont été en partie testés à l'INFOCLUB sur le T1600 du Lycée. Il ne vous reste plus qu'à vous lancer dans votre réalisation pour friser les exploits... ou les catastrophes !
O temps réel, suspends ton vol et nous laisse quelques jours pour finir cette programmation ! Nom d'un octet !

Pour construire votre voiture-micro, vous pouvez aussi consulter la revue Microsystème, 43, rue de Dunkerque — 75010 Paris.

La rubrique Vie des Clubs de E & I attend vos idées, vos critiques, vos essais, vos réussites autour de la voiture-micro que le lycée technique Diderot vous à aidés à construire. Ecrivez-nous...

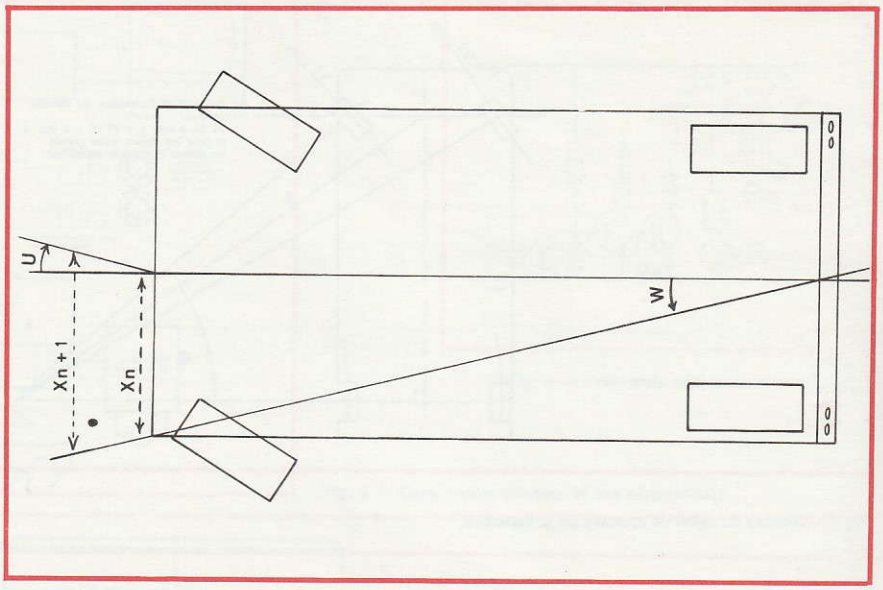


Fig. 10 — Exemple de paramètres pouvant décrire le comportement du véhicule.

```
SIMULATION DE VOITURE                          6-11-81        PAGE 1


AP $SIMVH
LI
????
LI 1
1* SIMULATION DE VEHICULE
2* A. LABREUILLE LE 26/1/1981
3* LA SIMULATION UTILISE DES EQUATIONS RECURRENTES
4* X(N+1) = X(N  + L.TG(T).COS(F)+L.SIN(F)
5* T(N+1)=/ = T(N) + (L/D).SIN(F)
6* L: DISTANCE SEPARANT DEUX CALCULS(DEPLACEMENT DU NEZ)
7* D: LONGEUR(EMPATTEMENT) DU VEHICULE
8* F: ANGLE DE BRAQUAGE; T: ANGLE DE VEHICULE AVEC LA ROUTE
9* X: DISTANCE MESURE SUR LA RAMPE DE CAPTEURS
10 PROCEDURE &SIM(X,T,F,D,L)
20 X-X+L*COS(F)*SIN(T)/COS(T)+L*SIN(F)
30 T-T+L/D*SIN(F)
40 F--T-X/L
50 SI ABS(F))26*3.14/180 ALORS F-F/ABS(F)*26*3.14/180
60 RETOUR
100 L-0.01;D-0.4;F-0
110 X-0.1
120 T-0
150 FAIRE 200 TANT QUE ABS(X))0.001 OU ABS(T))3.14/180
160 AFFICHER"/,2X,F2.3,2X,F2.3,2X,F2.3"X,T*180/3.14,F*180/3.14
170 &SIM(X,T,F,D,L)
180 &SIM(X,T,F,D,L)
200
210 TERMINER
```

Cette richesse est mal utilisée car elle nécessite l'écriture de programmes complexes, programmes habituellement écrits par des enseignants dont le métier n'est pas celui de programmeur. Aussi, très vite se limite-t-on au noyau principal du langage, celui qu'on acquiert et maîtrise rapidement. Quoiqu'il en soit, l'étude et la maîtrise suffisante d'un langage d'EAO requiert plusieurs semaines. Mais, un langage étant maîtrisé, il faut bien l'utiliser.

C'est ici qu'intervient la logique propre de ce langage dans la démarche pédagogique de l'enseignement. S'il n'est possible de définir des exercices convenables qu'au prix d'une programmation longue et complexe, l'enseignement verra très vite infléchir sa démarche pédagogique vers la voie de l'enseignement programmé. Même dans ce cas, plusieurs dizaines d'heures de travail sont nécessaires pour créer une heure de cours.

Il résulte de ce qui précède que le choix d'un langage d'EAO impose dans les faits une certaine démarche pédagogique. Là encore, cette option se manifeste au niveau des exercices.

Programmer un exercice en EAO relève d'une attitude différente de celle généralement adoptée en enseignement traditionnel. Dans cette dernière forme d'enseignement, le maître choisit un exercice, le donne aux élèves, « ramasse les copies » et corrige ces copies une à une. En EAO, il est nécessaire d'anticiper les réponses des élèves, bonnes ou mauvaises, afin de les sanctionner par un commentaire approprié. C'est précisément là qu'il faut faire appel aux plus grandes qualités d'un langage d'EAO, et qu'il est nécessaire de fournir le plus gros effort de programmation. De plus, il est naturellement indispensable d'imaginer des exercices adaptés à l'EAO, c'est-à-dire en fait, relevant des possibilités de l'EAO.

Ces limites sont d'ordre théorique et d'ordre pratique. S'il est vrai qu'en théorie « on peut tout faire », il n'en demeure pas moins qu'il s'agit de déterminer si la somme de travail consacré à une tâche présente suffisamment d'intérêt, bref, si le « jeu en vaut la chandelle ».

Là encore, tout dépend du langage utilisé. C'est ainsi que tel exercice sera simple à créer dans un langage A, difficile dans un langage B, impossible dans un langage C, ou tout-à-fait irréalisable quels que soient les langages existants. Prenons un exemple simple. Supposons un exercice dont la solution est 200 F. Choisissons pour commencer, un langage non dévolu à l'EAO, BASIC. Il faudra se livrer à une véritable gymnastique de programmation pour autoriser l'élève à répondre : 200 francs, 200francs, 200 Francs, 200Francs, 200 Fr, 200 F, 200F, etc. Un langage d'EAO bien conçu permettra de supprimer éventuellement les blancs dans la réponse de l'élève, transformer automatiquement les majuscules en minuscules, autoriser la comparaison sur le début du mot seulement (préfixe).

Dans ces conditions, l'analyse de l'exemple choisi se ramène à prévoir 200f ; car, si l'élève répond : 200 Francs, le système réalisera les transformations : 200 Francs → 200Francs ; 200Francs → 200francs, et reconnaîtra 200f dans 200francs.

Mais, bien que l'analyse de la réponse de l'élève soit ainsi simplifiée, il n'en demeure pas moins que l'intérêt pédagogique est plutôt mince. La réponse à analyser est en effet du type « OUI/NON » ou « VRAI/FAUX ».

Soit l'exercice suivant : Un négociant achète 100 kg de pommes à raison de 3 francs le kg. Il les revend 5 francs le kg. Quel est son bénéfice ? Mis à part le fait de provoquer une réponse exacte, l'intérêt pédagogique d'un tel exercice serait considérablement accru si l'élève, répondant 200 francs, recevait en retour le message : EXPLIQUEZ-MOI COMMENT VOUS ÊTES ARRIVÉ A CE RÉSULTAT. L'élève peut répondre par exemple : « le commerçant a acheté ses fruits 300 f, il les a vendus 500 f d'où le bénéfice annoncé ».

Si de plus, le système peut analyser une telle réponse, alors l'intérêt pédagogique est indubitable.
Qu'il soit permis à l'auteur de préciser que non seulement ce type d'exercices (et bien d'autres plus élaborés encore !) est parfaitement réalisable en CMEAO, mais que sa création ne requiert qu'une vingtaine de minutes et ce, sans aucune connaissance d'un quelconque langage de programmation. Bien entendu, cet exercice admet des réponses différentes de celles données en exemple. On peut également détecter le cas échéant les fautes de calcul et certaines erreurs de raisonnement.

Il est bien évident que l'EAO n'en est qu'à ses débuts. Aussi pourrait-on espérer voir évoluer les langages spécialisés de façon à ce que leurs performances en profitent. Telle n'est cependant pas l'opinion de l'auteur.

L'avenir de l'EAO, ses succès, sa généralisation, passent par la disparition totale de ces langages.

Aujourd'hui, il est déjà possible de se passer de ces langages pour créer, modifier, enchaîner des leçons comportant des exercices de grande valeur pédagogique. Il est également possible de créer « à vue », en déplaçant le curseur sur l'écran, des graphiques suffisamment prévis pour être significatifs. C'est dans cette voie qu'il faut s'engager. Peut-être en définissant un langage système intermédiaire et universel dans lequel les « métaprogrammes » du type « CMEAO » traduiraient le dialogue entre le maître et la machine, ce dialogue s'exprimant dans une langue naturelle quelconque. Cette approche aurait de nombreux avantages : transportabilité d'un système à un autre, corrections, modifications des leçons sans avoir à « comprendre » un programme écrit par une tierce personne.

Enfin, ce langage intermédiaire pourrait être aussi complexe que nécessite l'application puisque personne n'aurait à s'en servir, le programmeur étant précisément le « métaprogramme ».

Amis enseignants, le débat est ouvert...

Maurice Peuchot

Vous avez dit « algorithme » ?

J. Hebenstreit

Holmes regarda d'un air pensif des hiéroglyphes affichés sur l'écran de la console sur laquelle il était penché depuis que j'étais rentré, ferma l'interrupteur, se renversa dans son fauteuil, tira lentement sur sa pipe et dit soudain :

— Non, mon cher Watson, quoi qu'en dise votre nièce, la recette du porc aux pruneaux n'est pas un algorithme.

Je tressaillis comme chaque fois qu'une de ses phrases s'insérait dans le fil de mes pensées avec la même précision que si j'avais pensé à haute voix.

— Ne sursautez pas mon cher Watson. C'est tout à fait élémentaire. Je savais qu'avant de venir ici vous iriez voir votre nièce. Il était facile de prévoir que celle-ci, connaissant votre gourmandise naturelle et pour vous remercier de lui avoir acheté un micro-ordinateur, s'empresserait de mettre dans la mémoire de son micro votre recette préférée, afin de l'afficher triomphalement sur l'écran cathodique. De là à déduire qu'en me voyant travailler sur une console vous repenseriez avec attendrissement à votre conversation avec la petite Sophie, il n'y avait qu'un pas... que j'ai franchi lorsque je vous ai vu sourire d'un air satisfait.

— Tout ceci ne me dit pas pourquoi la recette du porc aux pruneaux, n'est pas un algorithme car après tout, si la recette est suffisamment détaillée, n'importe qui peut y arriver ; or c'est bien ce qui caractérise un algorithme.

Holmes poussa un profond soupir.

— Écoutez Watson, je comprends que vous ayiez pu être impressionné par le jargon technique de la petite Sophie ; malheureusement, il ne suffit pas de posséder un micro et d'avoir suivi un stage de 3 jours, même payé au prix fort, pour devenir compétent en informatique. La notion d'algorithme est une notion difficile qui ne souffre aucun à peu près. Il y a de multiples raisons qui font qu'une recette de cuisine n'est pas un algorithme.

Par exemple, avez-vous jamais entendu dire à quelqu'un qu'il allait résoudre le problème du porc aux pruneaux ? De plus quand deux cuisiniers font le même plat, le résultat n'est qu'approximativement le même selon la qualité de la viande, le talent du cuisinier, etc.

Que penseriez-vous d'une addition dont le résultat dépendrait du talent du calculateur ou de la taille des chiffres ? Ce qui caractérise un algorithme, c'est sa rigueur. Un algorithme est, par définition, une suite de consignes parfaitement définies et en nombre fini qui, appliquées à des données tranforment celles-ci en ???????

Les règles qui permettent d'additionner deux nombres forment un excellent exemple d'algorithme.

— Alors il n'existe d'algorithmes qu'en mathématiques ?

— Si vous voulez dire qu'un algorithme est un objet mathématique et plus précisément de la logique mathématique, vous avez raison, mais il existe quantité d'algorithmes qui s'appliquent à des problèmes mathématiques ou qui ne sont pas considérés comme tels.

Il y a, par exemple, les algorithmes de classement alphabétique, les algorithmes qui permettent de transposer des thèmes musicaux d'une tonalité dans une autre, etc.

De manière générale, on ne peut parler d'algorithme que pour des problèmes dont les données sont des *symboles* ayant des propriétés parfaitement définies. C'est le cas des nombres, des lettres ou de

l'ensemble des signes conventionnels utilisés en notation musicale.

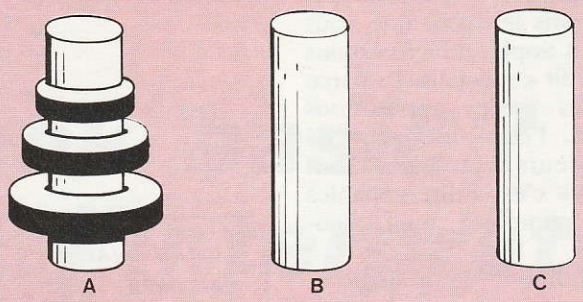
— Votre définition me semble singulièrement restrictive car il existe bien un algorithme pour le jeu des Tours de Hanoï, or les anneaux de ce jeu ne sont pas des symboles que je sache.

— Excellente objection Watson mais qui tombe d'elle-même si j'ajoute que les symboles dont j'ai parlé peuvent parfaitement représenter des objets réels. Je dirai même plus : il n'existe d'algorithmes pour des problèmes portant sur des objets réels que si ceux-ci peuvent être représentés par des symboles qui feront, eux, l'objet des consignes de l'algorithme. De manière plus concrète, si vous jouez aux Tours de Hanoï avec 3 anneaux et que vous trouvez, par hasard, la solution, vous pouvez noter la succession des coups joués et écrire la solution sous la forme : prendre le premier anneau et le transférer sur la deuxième tige, etc., mais il vous faudra tout recommencer si vous passez à 4 anneaux, 5 anneaux, etc., et si vous n'avez pas essayé de jouer avec 6 anneaux, vous ne pourrez ni donner la solution ni même affirmer qu'il existe une solution. Vous aurez donné des recettes ou des modes d'emploi ou des méthodes, mais pas un algorithme. Je peux, par contre, poser le problème en disant que j'ai 3 tiges A, B et C et que les anneaux initialement en A doivent aller en C. Je dis ensuite que, en jouant avec 1 seul anneau, je sais le faire passer soit en B soit en C, ce qui est évident. Je suppose maintenant que je sais faire passer n anneaux de A soit vers B soit vers C. Si j'ai n + 1 anneaux en A, je sais, par hypothèse, faire passer n anneaux vers B. Une fois que c'est fait, je fais passer le n + 1^{e} anneau de A vers C. Il ne me reste plus qu'à finir en faisant passer les anneaux de B vers C, ce que je sais faire puisque, par hypothèse, je sais les faire passer soit vers A soit vers C.

J'ai non seulement montré que le problème avait une solution quel que soit le nombre d'anneaux, mais j'ai, de plus, donné une méthode valable quel que soit ce nombre : j'ai défini un algorithme et vous remarquerez que pour l'établir je n'ai pas eu besoin de disposer d'anneaux ni de faire des essais.

J'ai raisonné sur des représentations abstraites des anneaux ; c'est exactement cela que j'ai voulu dire tout à l'heure en parlant de symboles représentant des objets réels.

Le jeu des Tours de Hanoï



Le jeu consiste à faire passer les anneaux d'une tige sur une autre tige en utilisant une tige intermédiaire de façon qu'un anneau ne se trouve jamais au-dessus d'un autre d'un diamètre inférieur.

— Excusez-moi mon cher Holmes, mais si je veux bien admettre que les recettes de cuisine ne sont pas des algorithmes, je pense que vous allez trop loin. Vous avez parlé, tout à l'heure, d'un algorithme de transposition musicale, mais si vous me donnez un tel algorithme je ne saurai pas l'exécuter puisque je ne connais rien à la notation musicale et donc votre algorithme n'est pas un algorithme pour moi !

— Félicitations, mon vieux, vous faites des progrès ! Il est vrai que ma définition est insuffisante et qu'il faut aller plus loin. J'ai dit qu'un algorithme était une suite de consignes à exécuter dans un ordre prescrit pour obtenir la solution d'un problème, mais la description de ce qu'*on* doit faire pour obtenir la solution. Et votre objection revient à dire : mais qui est « on » ? Je dois dire que pendant très longtemps on ne s'est pas vraiment posé le problème. Le mathématicien Al-Kwarizmi qui vivait au 9^{e} siècle et qui a inventé la notion d'algorithme (le mot « algorithme » dérive du nom de ce mathématicien) faisait l'hypothèse implicite que les « exécuteurs » de ses algorithmes étaient ses collègues, ce qui simplifiait le problème. Ce n'est guère qu'au 20^{e} siècle et à la suite des travaux du mathématicien britannique A.-M. Turing que l'on a cherché à préciser ce point. Si l'on convient d'appeler « processeur » un dispositif physique quelconque capable, par construction, d'exécuter certaines consignes, alors on dira que l'ensemble de ces consignes définit la « compétence » du processeur.

— Si je comprends bien, une calculatrice « 4 opérations » est un processeur dont la compétence est limitée aux quatre opérations de l'arithmétique.

— Exact. Et pour ce processeur, un algorithme de classement alphabétique n'est pas un algorithme puisque non exécutable.

— Est-ce que cela signifie qu'avant d'écrire un algorithme, il faut commencer par s'assurer qu'il existe un processeur dont la compétence permet l'exécution de cet algorithme ?

— Vous brûlez, mon cher Watson, encore que le processeur en question n'ait pas vraiment besoin d'exister physiquement. Il suffit de montrer qu'il est réalisable et de faire comme s'il existait.

— Bien, admettons, mais vous rendez-vous compte qu'avec les conditions que vous imposez, on va passer son temps à définir des millions et des millions de processeurs pour tous les algorithmes qui existent. C'est complètement dément.

— Calmez-vous, Watson. Les choses sont fort heureusement plus simples que ce que vous croyez. Le résultat le plus stupéfiant de la théorie des algorithmes c'est qu'il existe des processeurs « universels ».

— Qu'est-ce que ça veut dire « universel » ?

— Cela veut dire qu'il est possible de définir un processeur capable d'exécuter *tous* les algorithmes qui existent, y compris ceux qui existeront un jour et qu'on ne connaît pas encore.

— C'est un joli résultat théorique mais j'imagine, compte tenu de tous les algorithmes qui existent, que le jour où on voudra construire ce fameux processeur universel, il

sera plus gros que l'Arc de Triomphe !

— Eh bien pas du tout, car le premier ordinateur venu fera l'affaire y compris le micro que vous avez offert à Sophie. Sauf certains ordinateurs dit « spécialisés » parce que destinés à des applications particulières, l'écrasante majorité des ordinateurs construits sont « universels » c'est-à-dire capables d'exécuter n'importe quel algorithme.

— A condition, naturellement, que ce « n'importe quel algorithme » soit écrit en n'utilisant que des consignes qui font partie de la compétence du processeur, c'est-à-dire, en d'autres termes, en n'utilisant que des instructions qui font partie du langage de programmation de l'ordinateur utilisé.

— Bravo, mon cher Watson vous avez gagné...

— Connaissant votre pingrerie, je préfère ne pas savoir ce que j'ai gagné, mais je voudrais vous posez une question indiscrète : Comment peut-on être sûr qu'un processeur est « universel » ? ou, en d'autres termes, comment peut-on être sûr qu'on ne trouvera pas, un jour, un algorithme que le processeur, jusque là considéré comme universel, sera incapable d'exécuter ?

— Re-bravo, Watson, vous venez de poser *la* question. En effet, pour montrer qu'un processeur est universel, il faut le *démontrer* (mathématiquement) ; or les ordinateurs ont un fonctionnement trop complexe et trop particulier pour faire l'objet d'une démonstration. C'est à A.-M. Turing que revient le mérite d'avoir, vers les années 30 (donc avant l'invention des ordinateurs), défini un processeur d'une simplicité extrême, puisqu'il ne comporte qu'une demi-douzaine de consignes en tout et pour tout, et d'avoir démontré que ce processeur était universel ; c'est ce qu'on appelle la « machine de Turing » bien qu'elle n'ait jamais été construite.

— Et pourquoi ne l'a-t-on jamais construite ? Si elle est tellement simple elle aurait été très bon marché.

— Sans doute, mais inutilisable en pratique car chacune de ses consignes effectue un traitement tellement élémentaire qu'il faut, par exemple, plus de dix consignes (un informaticien dirait plus de dix instructions) simplement pour faire une addition de 2 nombres entiers. J'ajoute que pour l'arithmétique, elle ne travaille ni en décimal ni en binaire mais avec des bâtons, c'est-à-dire que pour additionner 3 et 2 il faut lui fournir les données sous la forme III + II. Imaginez le travail du programmeur qui voudrait lui faire additionner 293625 et 531632 !

— Bon, d'accord, j'ai tort une fois de plus, mais pour en revenir à Turing et à son processeur universel, j'imagine que pour comprendre la démonstration, il faut au moins sortir de Polytechnique...

— Pas du tout, et c'est ce qui fait la beauté de la chose. Si la démonstration n'est pas très courte, elle est par contre extrêmement simple, il suffit de montrer (et c'est ce qu'à fait Turing) qu'il existe pour la machine de Turing un algorithme qui permet à cette machine de simuler le comportement de n'importe quelle autre machine de Turing exécutant n'importe quel algorithme. Vous voyez que par ce procédé...

— Doucement, Holmes, n'essayez pas de noyer le poisson. Pour que votre raisonnement soit correct, il faut encore prouver que pour n'importe quel algorithme, il existe une machine de Turing capable d'exécuter celui-ci, et cela ne me paraît pas évident.

— Rassurez-vous, cela va le devenir pour la simple raison que Turing se débarrasse de votre objection en posant comme axiome que pour tout algorithme, il existe une machine de Turing capable d'exécuter cet algorithme.

— Et ça ne vous choque pas ? Ce Turing invente un processeur à 6 schtroumpfs et pour se débarrasser de mes objections, déclare négligemment que par définition, son processeur peut exécuter tous les algorithmes possibles et imaginables, passés et à venir, et vous trouvez ça très bien. Moi, j'appelle ça de la mauvaise foi. Dites-moi, Holmes, sérieusement, juste entre vous et moi, ils sont nombreux à penser comme vous ?

— Eh bien, à vrai dire, quand Turing a publié ses travaux, il n'a pas vraiment fait l'unanimité, du moins au début. Mais, comme on n'a pas réussi à prouver qu'il avait tort et que, de plus, les conséquences de son axiome ne sont pas en contradiction avec d'autres travaux sur ces mêmes problèmes, il a bien fallu admettre qu'il avait raison.

— Parce que, en plus, ce fameux axiome a des conséquences ?

— Et non des moindres, car les thèses de Turing ont ouvert un nouveau chapitre de la logique dans lequel les algorithmes deviennent eux-mêmes des objets mathématiques sur lesquels on peut faire des démonstrations et où l'on trouve des choses aussi bizarres que des fonctions « non-calculables » ou des problèmes « indécidables » qui sont d'une importance fondamentale pour les informaticiens.

— Écoutez, Holmes, tout ceci est peut-être élémentaire, mais je propose que nous ne reprenions cette conversation que si nos lecteurs nous le demandent parce que vous avez fini par me donner mal à la tête. En fait, pour vous dire le fond de ma pensée, je me demande si l'étymologie du mot algorithme est bien celle que vous dites, et si ce mot ne dérive pas plutôt du grec ; de arithmos qui veut dire nombre et de algo qui veut dire douleur.

J. Hebenstreit

Do you speak basic ? Qu'importe, j'interprète...

Bruno Pétazzoni

Comment les ordinateurs peuvent-ils exécuter les instructions données par les programmes dans des langages évolués ? C'est grâce à un ensemble de programmes appelé généralement un « système », et principalement à « l'interpréteur ». Cet article tente d'expliquer, sans entrer dans les détails, les fonctions remplies par un interpréteur.

Les microprocesseurs qui équipent les micro-ordinateurs sont prévus pour manipuler des « octets », c'est-à-dire des groupes de huit bits. Que peut-on faire avec un octet ? Ça dépend... Notons tout d'abord qu'à l'aide d'un codage binaire, on peut représenter les nombres entiers de 0 à 255. En effet, il suffit de ranger les huit bits de gauche à droite, et de leur affecter des poids respectifs de 1, 2, 4... jusqu'à 128. Par exemple, l'octet : 00101001, vaudra, avec cette méthode :
0×128+0×64+1×32+0×16+1×8 +0×4+0×2+0×1,
soit au total, 41
On peut alors coder tout ce que l'on veut, avec ces numéros de 0 à 255 :
des couleurs, des noms de fromages ou des départements français (dans ce dernier cas, on n'utilisera pas tous les numéros possibles...). En pratique, il y a au moins deux codes importants : celui des CARACTÈRES et celui des INSTRUCTIONS.

— Le code des CARACTÈRES, sur micro-ordinateur, est pratiquement toujours le code « ASCII ». Par exemple, dans ce code, le numéro 66 est dévolu à la lettre B (majuscule), le numéro 59 au point-virgule, etc.

— Le code des INSTRUCTIONS lui, dépend du microprocesseur utilisé. En effet, pour le Z80 présent dans les micro-ordinateurs LX500 et MICRAL, le numéro 0 est dévolu à l'instruction « ne rien faire », et le numéro 118 à l'instruction « s'arrêter ». Intéressons-nous de plus près à ce code : chaque numéro correspond à une instruction particulière, que le Z80 peut exécuter. Le problème c'est que ces instructions sont très rudimentaires : ainsi le Z80 ne peut additionner que des nombres compris entre 0 et 255 (qui tiennent sur un octet quoi !). Or, nous voulons programmer en langage dit « évolué », tel que le LSE permettant d'additionner des nombres plus variés. C'est l'interpréteur qui va nous en fournir le moyen : il s'agit d'un programme écrit avec les instructions reconnues par le Z80, et capable de « comprendre » d'autres programmes qui eux, sont écrits en LSE.

LE CAS BASIC

Dans un premier temps, examinons le cas du BASIC (qu'il s'agisse du BASIC LOGABAX du LX500 ou du BASIC microsoft du MICRAL, c'est à peu près pareil). Nous remarquons tout d'abord qu'une instruction en BASIC se compose d'un verbe, éventuellement suivi de complément(s) :
PRINT « Ça marche au poil ! »
GOTO 345.

Une exception : l'affectation comme A = 34, bien que dans le BASIC original, il fallait écrire LET A = 34. Ce n'est que bien plus tard que la forme A = 34 fut autorisée. Aussi, laissons donc ce cas de côté.

Le travail de l'interpréteur va donc consister à reconnaître ce verbe. Ensuite, il confiera l'exécution du travail correspondant, à un programme particulier. Il y aura autant de tels programmes que de verbes différents. En même temps, l'interpréteur gère un « pointeur » qui indiquera à chaque instant où est la prochaine instruction à exécuter. C'est en général, celle qui est juste après la dernière instruction exécutée, mais il existe des cas particuliers comme le programme chargé de traiter GOTO, qui devra replacer ce pointeur au début de la ligne visée par le GOTO.

La reconnaissance des verbes (ou mots-clés) est très simple. En effet, le programme en BASIC n'est pas rangé en mémoire exactement comme il a été tapé : chaque verbe est remplacé par un seul octet au moyen d'un codage. La méthode utilisée est la suivante : le code ASCII des caractères n'utilise que les 128 premiers octets, les 128 suivants étant donc disponibles ; ainsi, en BASIC Microsoft, le verbe PRINT a pour code 145.

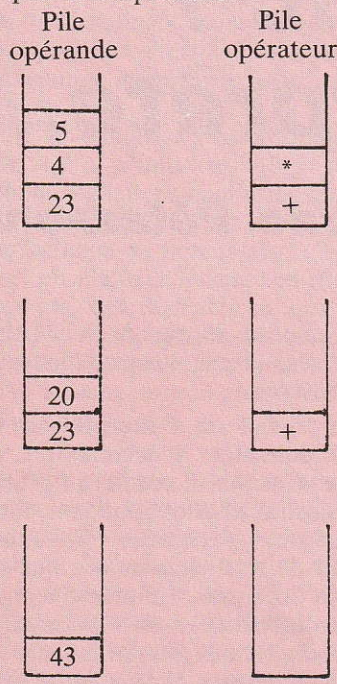
Il reste le problème de l'évaluation des compléments. Par exemple, dans PRINT 23 + 4 * 5 ; les différents verbes font appel à un programme « évaluateur » assez complexe, capable de réaliser les calculs. Cet évaluateur utilise deux « piles ». En effet, nous remarquons que dans l'expression PRINT 23 + 4 * 5, le signe + apparaît avant l'étoile ; cependant, il faut d'abord calculer le produit 4 * 5 puis l'ajouter à 23. La première pile reçoit les opérandes et la deuxième, les opérateurs (voir Tableau 1).

Tableau 1

23 est reconnu et empilé sur la pile des opérandes

+ est reconnu et empilé sur la pile des opérateurs

4 est reconnu et empilé sur la pile des opérandes



☆ est reconnu et, étant de priorité inférieure à + (sommet actuel de la pile des opérateurs) est également empilé sur la pile des opérateurs

5 est reconnu et empilé sur la pile des opérandes

L'évaluateur reconnaît la fin de l'expression. Il déclenche alors autant de calculs qu'il reste d'opérateurs non évalués; chaque calcul dépile les deux opérandes situés le plus haut et met le résultat sur la pile. Ici l'étoile va s'appliquer à 5 et 4; le résultat: 20, est mis sur la pile où il reste donc 20 et 23; ensuite, le + est appliqué à ces deux valeurs, ce qui donne bien le résultat escompté : 43.

Bien entendu, notre évaluateur fait avancer le pointeur évoqué plus haut à mesure qu'il évalue.

LE CAS LSE

Ici, les choses sont nettement différentes : chacun a pu constater que si l'on pouvait taper une ligne incorrecte en BASIC comme : PRINT GOTO PRINT sans provoquer de réaction du système. Par contre, en LSE, la frappe de la ligne : AFFICHER ALLER EN AFFICHER amène immédiatement un diagnostic d'erreur, dite « de compilation » sous la forme d'un message du genre : ERREUR C 34. Ceci vient de ce que LSE, aussi bien sur micro que sur mini, à ce jour est « précompilé ». C'est-à-dire que chaque ligne de programme tapée, est analysée par un élément du système appelé « compilateur ». Ce compilateur, lui aussi écrit en langage machine, regarde si la ligne est « bien » composée (si elle suit les règles de la grammaire LSE). Ainsi, cette grammaire affirme que le verbe AFFICHER doit être suivi d'une expression. Or, ALLER est un mot réservé qui ne eut apparaître dans une expression... D'où la vive colère du système LSE, à juste titre !

Mais le travail du compilateur ne s'arrête pas là : si la ligne est correcte, il en fabrique une version codée (comme dans BASIC) mais cette fois le codage est beaucoup plus poussé. Prenons l'exemple de la ligne : A←LGR(B) tapée en mode « calculateur de bureau ». La forme codée occupe 4 octets valant successivement 1, 161, 177 et 0 ;

— La forme codée est appelée « chaîne post-fixée » ou en abrégé, CHPF

— Dans la CHPF, il y a deux types d'octets : ceux de 0 à 127 servant à coder les identificateurs (noms de variables) et ceux de 128 à 255 utilisés pour coder les opérateurs

— Chaque opérateur est placé APRÈS les opérandes auxquels il s'applique sauf cas très particulier. Ici 161, est le code de LGR (longueur), 177 est le code de l'affectation.

C'est parce que l'on met les opérateurs APRÈS les opérandes que l'on a donné le nom de « post-fixée » à cette notation. Ceux qui ont manipulé une calculatrice HULETT PACKARD la connaissent bien.

Le travail de l'interpréteur est simplifié puisque ce qui lui est proposé est syntaxiquement correct. De plus, il n'y a pas de pile d'opérateurs car le compilateur a fait le travail nécessaire avant ; ainsi, A + B * C est noté, en fait : A B C * + ; (tout étant bien évidemment codé). Chaque opérateur s'applique aux deux opérandes sommets de la pile. Donc l'étoile s'applique à B et C ; et + s'applique à A et au résultat B*C. On gagne ainsi un temps appréciable à l'exécution.

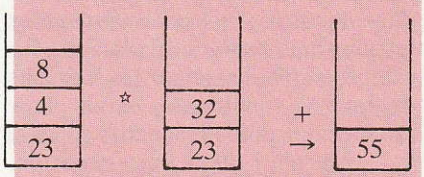
Voyons maintenant en détail le travail de l'interpréteur : quand il rencontre un octet compris entre 0 et 127, il s'agit d'un nom de variable qu'il considère comme un opérande et empile donc. Quand il rencontre un octet compris entre 128 et 255, c'est un opérateur ; il sous-traite le travail à un programme spécifique de cet opérateur. Celui-ci prendra ses opérandes sur la pile (si nécessaire, pour PAUSE, c'est inutile !).

Une fois son travail terminé, l'opérateur place éventuellement un résultat sur la pile, puis l'interpréteur continue à balayer la CHPF (voir tableau 2).

EXEMPLE D'INTERPRÉTATION EN LSE — Tableau 2

CHPF

23 + 4 ☆ 8 → 23 4 8 ☆ +



La contrepartie de cette méthode, c'est que le « système » est beaucoup plus gros. Pour pouvoir LISTER, il faut retraduire la CHPF en une forme externe lisible par l'utilisateur. C'est le rôle du « décompilateur ».

Accessoirement, on comprend pourquoi certaines lignes de programme, en LSE ne sont pas listées exactement comme elles ont été tapées. Ainsi la ligne

253 A ← (2 + 3) est listée 253 A ← 2 + 3

les parenthèses étant inutiles dans ce cas. On note d'ailleurs que dans la CHPF, il n'y a pas de parenthèses. C'est l'un des avantages de la notation post-fixée.

POUR CONCLURE

Il y aurait beaucoup à raconter par exemple, sur ce qui se passe

quand on rentre dans une boucle, ou quand on rencontre un appel de procédure, mais ce serait plutôt pénible et très technique. Ce qu'il faut retenir, c'est que en LSE, on a une analyse préalable lors de la frappe du programme, et un codage beaucoup plus complexe dans un véritable langage, intermédiaire entre LSE et le langage machine. Il faut noter que la même méthode avait été proposée par Wirth pour son langage PASCAL ; celui-ci était traduit dans un premier temps, par un compilateur, en une forme appelée P-code (Pseudo-code). Ensuite, ce P-code pouvait être interprété ou même exécuté directement par un microprocesseur dont il constituait alors le langage machine.

Il existe effectivement sur le marché un ordinateur ainsi construit. C'est la « Pascaline », réalisée autour d'un microprocesseur 16 bits Western Digital. Il serait très intéressant de disposer d'une machine analogue pour le LSE, mais ceci semble incompatible avec sa diffusion encore faible. Actuellement, il n'est implanté que sur un peu plus de 1 000 micro-ordinateurs. L'avenir tranchera ! La précompilation du langage apporte un confort d'utilisation indiscutable : les erreurs « bêtes » sont détectées et corrigées instantanément. Si la CHPF est bien conçue et si l'interpréteur est bien écrit, on peut également gagner du temps à l'exécution : codage des nombres au format interne déjà réalisé, mise en forme des expressions... Bien entendu, on n'a rien sans rien et ce confort se paie en place mémoire : si un BASIC correct avec un système gestion de fichiers convenable laisse une quarantaine de K octets à l'utilisateurs, il n'en restera plus que 16 en LSE.

Cependant, il semble que la technique du langage intermédiaire doive se répandre, même en BASIC. En effet, selon un article paru dans la presse américaine, MOTOROLA ferait écrire, pour son microprocesseur 6809, un BASIC dit « à compilation incrémentale », qui ressemblerait comme un frère à LSE, aux verbes près...

Bruno Pétazzoni

VOS PROGRAMMES

La 7e version de Martine

Monsieur,

Je ne comprends plus rien. Dans vos programmes du N° 6, vous publiez 6 versions d'un programme qui pour moi est tout-à-fait inutile. Avez-vous lu « Les proverbes de programmation » de H.T. Ledgard (je ne fais pas de publicité !) ? Les deux premiers proverbes sont :

« Définissez le problème complètement »

« Réfléchissez d'abord, vous programmerez plus tard »

Que voulez-vous obtenir ? La somme des n premiers nombres entiers positifs $S = n + (n-1) + (n-2) + \ldots\ldots + 2 + 1$
Si je réfléchis, je vois que S peut s'écrire encore :
$S = n + (n-1) + (n-2) + \ldots\ldots + (n-(n-2)) + (n-(n-1))$

Et si maintenant j'ai l'idée de regrouper tous les termes précédés du signe + et tous les termes précédés du signe −, j'obtiens :
$S = n + n + \ldots + n - (1 + 2 + \ldots + (n-2) + (n-1)$

Alors dans ma grande parenthèse à droite, j'ai presque S aussi. Il suffit que je rajoute n. ainsi, je peux encore écrire :
$S = n^2 - (1 + 2 + \ldots (n-2) + (n-1) + n) + n$
$S = n^2 - S + n$

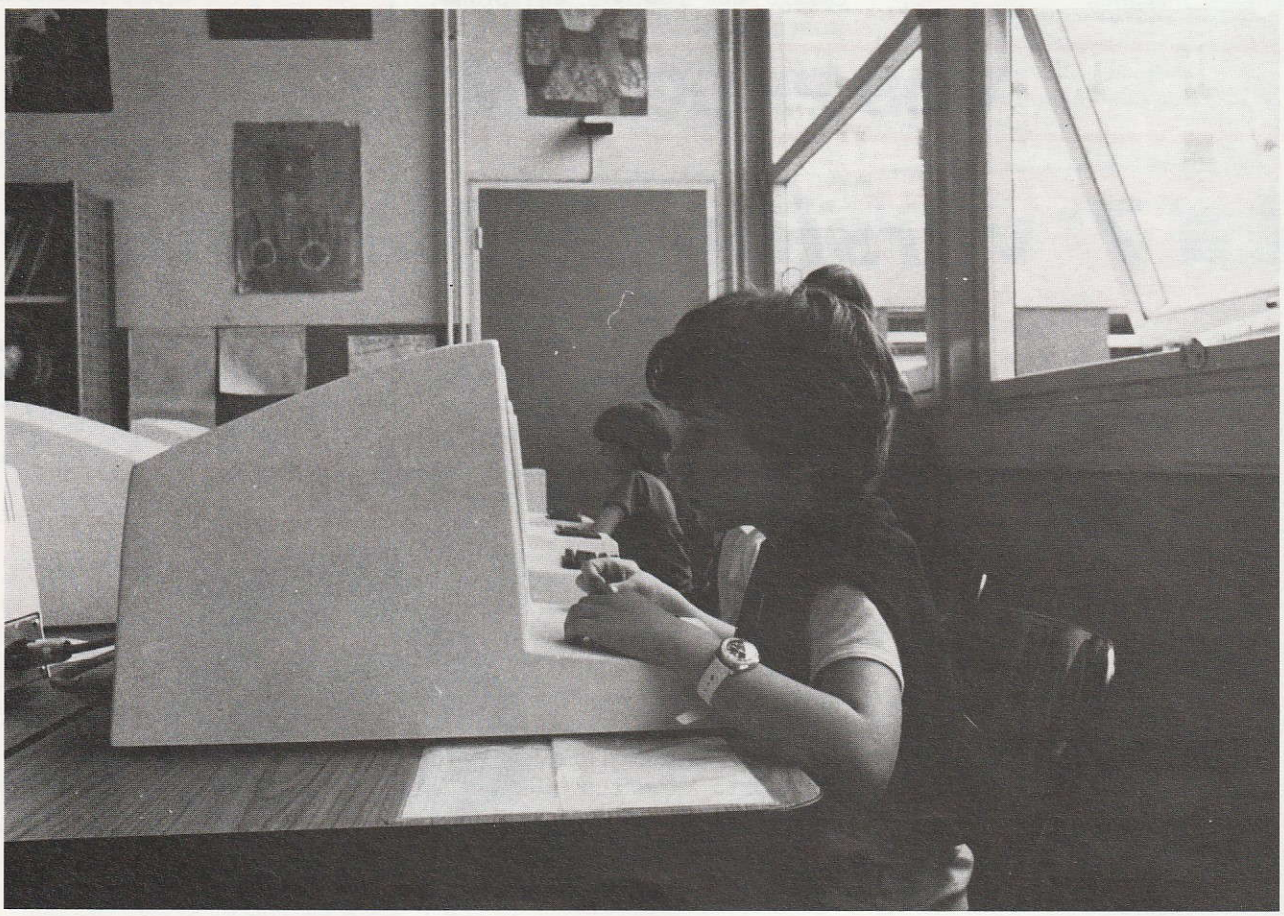
Et j'ai donc finalement $S = n(n+1)/2$ *que je peux calculer facilement à la main quelle que soit la valeur de n.*

D'accord, on a l'impression que c'est un truc et peut-être que ma prof de maths dira que ce n'est pas une belle démonstration. Mais moi, je vous ai montré que je n'ai pas besoin d'ordinateur pour votre problème. Alors, vous publiez ma version VII ! - Martine

Les classes élémentaires de l'école Bossuet

L'enseignement assisté par ordinateur dans le primaire

C. Le Roy



L'école Bossuet, établissement expérimental de plein exercice, fonctionne en « aires ouvertes ». La classe traditionnelle fait place à des ateliers depuis novembre 1980. Un atelier d'informatique où sont installés huit terminaux, est ouvert à titre expérimental aux 50 élèves des cours élémentaires.

Une préfiguration de l'expérience a été réalisée en juin dernier. Huit terminaux ont été installés à l'école. La C.G.I., Compagnie Générale d'Informatique, disposait en effet à cette date, d'un système d'Enseignement Assisté par Ordinateur — DIDAO — et souhaitait trouver un « terrain d'expérimentation » dans une école. Jean Michel Di Falco, en sa double qualité de directeur de l'I.S.P. (Institut Supérieur de Pédagogie) et des classes élémentaires de l'école Bossuet, a permis de réaliser cette expérience. 150 enfants ont pu travailler sur un cours de mathématiques (MA) à raison de deux séances consécutives de 7 minutes par jour.

Pourquoi une telle expérience ? Jean Michel Di Falco répond :

« Préparer l'avenir, c'est vivre au présent. L'informatique, la télématique, ne sont pas pour demain mais pour aujourd'hui. C'est pourquoi, tout doit être mis en œuvre pour que les enfants aient la maîtrise dès aujourd'hui de ces instruments. Le déphasage entre vie quotidienne et vie à l'école s'accentuerait si l'introduction des outils modernes d'apprentissage à l'école ne se faisait pas. Il importe que les enfants soient familiarisés à leurs usages, qu'ils apprennent à vivre, à grandir avec ces outils pour apprendre à les dominer. »

Historique DIDAO

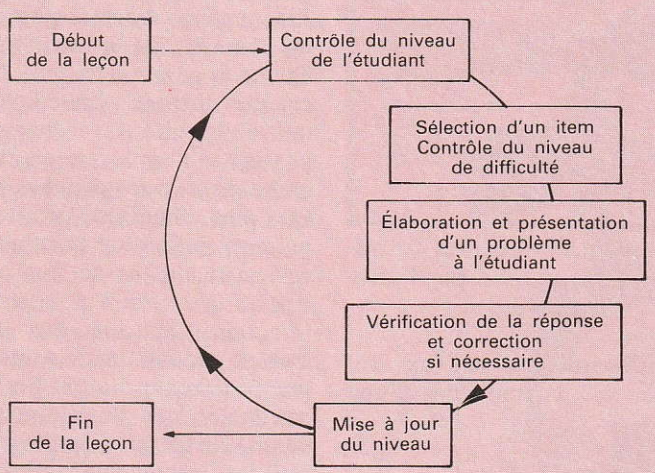
C'est d'une interaction entre l'université et l'industrie que sont nés les systèmes d'Enseignement Assisté par Ordinateur (E.A.O.), comme le système DIDAO de l'université de Standford en Californie. Mis au point par une équipe de chercheurs pédagogues de l'Institut de recherches Mathématiques en Sciences Sociales, que dirige le Pr. Suppes, ce système est maintenant utilisé par plus de 2 000 élèves des écoles primaires, secondaires, ou pour handicapés aux U.S.A. Le Pr. Suppes a créé en 1967 une société, Computer Curriculum Corporation (C.C.C.) qui avait installé 8 000 terminaux à la fin de 1980 dans plus de 3 000 écoles.

En France, c'est une société de service en informatique, la C.G.I., qui diffuse ses travaux, sous le nom de « DIDAO » et qui aura pour vocation de réaliser tous les didacticiels en langue française de la gamme C.C.C.

Les élèves dans un premier temps, ont adopté le système sans aucune réticence. Ils étaient aidés en cela par la grande simplicité des procédures de manipulation (donner son numéro, son prénom pour confirmation, puis le nom du cours choisi). Sensibles à l'aspect ludique de la machine, les élèves employaient pour parler d'elle un vocabulaire se référant très souvent au jeu. *« Est-ce que je peux faire encore une partie ?...* La relation individuelle et personnelle que l'ordinateur entretien avec chacun d'entre eux les a beaucoup impressionnés ; l'ordinateur doit avoir un « cœur » puisqu'il les salue en leur disant bonjour et au revoir et qu'il se rappelle d'eux. Certains élèves ont déclaré : *« Au moins, il est gentil parce qu'avec lui, on peut se tromper, il ne nous gronde jamais... »*

Le système DIDAO d'enseignement assisté par ordinateur :

CYCLE D'UNE SÉANCE



Ne pouvant aboutir à une évaluation sérieuse après une expérience trop brève, et très intéressée par l'EAO, l'I.S.P. a déposé un dossier de demande de subvention auprès de l'agence pour le développement de l'informatique pour la prise en charge de huit terminaux pendant un an, ceci afin de poursuivre la recherche engagée au cours du mois de juin.

L'A.D.I., en favorisant un tel projet, répond aux objectifs qu'elle s'est fixés : soutien et sensibilisation vis-à-vis des nouveaux utilisateurs de l'informatique.

Ainsi donc, une équipe d'évaluation, sous la tutelle de l'I.S.P. suit le déroulement de l'expérience et fera connaître ses conclusions sous la forme d'un rapport en novembre prochain.

A l'heure actuelle, cet atelier d'EAO trouve sa place dans la structure en « aires ouvertes » de l'école : c'est un atelier supplémentaire proposé aux enfants, qui organisent eux-mêmes leur travail chaque semaine, et se répartissent librement le matin dans les différents lieux de travail — Mathématique, Français, Peinture, Musique, Bibliothèque, Éveil de la foi (facultatif).

Après un temps d'initiation systématique pour tous les enfants concernés, l'accès aux terminaux, durant la première phase de l'expérience, a été facultatif. Depuis janvier 1981, les enfants des cours élémentaires travaillent obligatoirement chaque jour sur les terminaux, à raison de deux séquences toutes les dix minutes, ceci pour la rigueur de l'expérimentation.

Ils testent des programmes de maths (connaissance des nombres, assimilation des mécanismes opératoires) d'une part, l'application de l'arithmétique à la résolution de problèmes d'autre part, et des programmes de français (lecture et compréhension de textes, orthographe, grammaire, vocabulaire).

Les réactions des enfants sont diverses. Certains sont aidés par ce type de travail. La rigueur des programmes et la limite de temps imposées pour chaque réponse, développent la rapidité de leur raisonnement opératoire (calcul mental), surtout pour ceux ayant des difficultés d'écriture.

Certains apprécient l'aspect linéaire de la progression : répétition d'exercices, progression très lente, vérification incessante des connaissances.

D'autres enfants au contraire, sont gênés par un certain « piétinement » : les exercices ne sont pas assez variés, l'évolution en est trop lente. Habitués à progresser à leur rythme, à prendre en charge leur enseignement et à voir dans l'enseignant un guide, une aide, ils souffrent de la rigidité d'un tel système. Ils ont vite su démystifier la machine et ne pas la prendre au sérieux. On observe des enfants qui répondent volontairement de manière inexacte pour voir comment l'ordinateur va réagir, ou bien des enfants attendre que l'ordinateur donne lui-même les réponses.

Les enseignantes concernées par l'expérimentation ont volontairement adopté une attitude assez neutre. Bien renseignées sur l'EAO et ses implications, elles n'ont pas subi la « crainte mythique » de se voir supplantées

Le système DIDAO propose des séances de travaux dirigés sur des notions qui ont été préalablement présentées par l'enseignant. L'idée fondamentale de la méthode est de découper chaque cours en un certain nombre de « strands » bien définis, les items, chaque item correspondant à l'étude d'une notion très précise (par exemple, la multiplication verticale, la multiplication horizontale, etc., en cours de mathématiques).

Chacun de ces items est alors découpé en différents niveaux de difficulté croissante et à chaque niveau correspond une structure caractéristique d'exercices. C'est ce découpage rigoureux par items qui rend possible l'évaluation du niveau de progression de l'élève, par l'analyse des performances réalisées.

A l'heure actuelle, le cours de mathématiques commence au niveau du cours préparatoire (C.P.) et va jusqu'à la classe de 4e incluse. Il est divisé en quatorze « items ». De plus, chaque classe est divisée en dix niveaux.

L'ordinateur enregistre les travaux des élèves séparément pour chaque « item ».

Le système assure alors les fonctions suivantes :

— En principe, il choisit aléatoirement entre les différents items, en respectant les fréquentes données qui varient en fonction du niveau général atteint par l'élève. Puis pour l'item choisi, il propose un exercice (les valeurs numériques sont elles aussi, tirées au hasard).

— Il permet également, si le besoin s'en fait sentir, de consacrer une séance à l'étude d'un item donné (utilisation des « fixed strands »).

— Selon les résultats de l'élève, l'ordinateur ajuste le niveau des exercices proposés.

— Devant l'intensité de la concentration demandée à l'élève, chaque séance ne dure pas plus de dix minutes.

— A la fin de chaque séance, l'ordinateur fournit à l'élève le score global réalisé pour la séance.

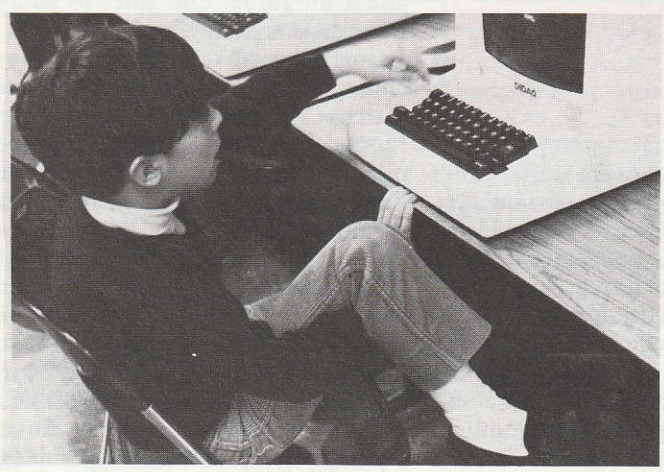
— Enfin, l'ordinateur peut fournir à tout instant à l'enseignant, un rapport détaillé soit par élève, soit par classe, lui précisant :

- *Le niveau actuel de l'élève dans chacun des items du module considéré et le niveau général qui en résulte,*
- *les progrès accomplis,*
- *les points faibles de l'élève.*

L'enseignant dispose donc d'un véritable « historique » de l'élève.

par l'ordinateur et ont accepté d'examiner les modifications que peut apporter ce système au niveau de leur enseignement. Elles analysent le contenu des cours et sont amenées à le critiquer. Elles utilisent également l'évaluation du travail des enfants fournie par l'ordinateur.

Quoiqu'il soit encore trop tôt pour tirer les conclusions d'une telle expérience, il est déjà possible de formuler un certain nombre d'hypothèses sur les implications pédagogiques du système employé.



Un système intégré à l'école

Un des aspects fondamentaux de la pédagogie mise en œuvre à l'école Bossuet est le travail personnalisé et l'acquisition des connaissances de base qu'il permet, en particulier en français et en mathématiques, par chaque élève à son rythme propre. Concrètement, ces acquisitions se font au moyen d'un ensemble de fiches de travail individuelles et progressives permettant la découverte et l'application des notions étudiées.
Le système DIDAO peut donc s'intégrer à cette pédagogie comme un substitut particulièrement performant au système des fiches. Il présente aussi un intérêt propre en renforçant certains aspects de cette pédagogie et en donnant de nouvelles possibilités.

Une progression adaptée à chaque élève

Le système guide pas à pas l'élève, lui signale ses erreurs et lui permet de les corriger sur le champ en lui donnant une deuxième chance, enfin lui fournit la réponse exacte en cas d'erreur persistante et lui demande de la taper lui-même. Et ceci, avec chaque élève, à chaque instant, bénéficiant d'un don d'ubiquité que l'enseignant ne possède qu'imparfaitement dans le cadre du travail sur fiches habituel. De plus, l'enfant profite de la patience infinie de l'ordinateur qui reviendra autant de fois qu'il faudra sur une notion mal assimilée, la sélection aléatoire des exercices évitant par ailleurs de sombrer dans une fastidieuse monotonie. Dans le cas d'exercices plus élaborés nécessitant une certaine forme de raisonnement (par exemple, l'usage à bon escient de la règle de trois) le système peut aider l'élève en décomposant en partie le raisonnement.

Éviter les situations d'échec

En outre, le système évite à l'enfant de subir des situations d'échec :

- Par rapport à lui-même en ne le laissant pas piétiner sur une difficulté, mais en le corrigeant immédiatement et en revenant éventuel-

lement à un niveau d'exercices plus simples ou à l'étude d'une autre notion.

- Par rapport aux autres puisque, comme dans le travail sur fiches, chacun évolue à son rythme propre de manière totalement indépendante

Concentration et comportement affectif

Enfin, le système DIDAO permet à l'enfant de travailler par séances relativement brèves (dix à vingt minutes), d'où une faculté de concentration optimum particulièrement bénéfique pour une acquisition rapide et durable des connaissances. De ce point de vue, l'utilisation d'un clavier pour dialoguer avec l'ordinateur impose un comportement actif à l'enfant qui renforce encore sa concentration.

Signalons enfin que l'utilisation de ce clavier fait disparaître le handicap que l'écriture peut représenter dans l'apprentissage d'une notion nouvelle en ajoutant une difficulté supplémentaire à la compréhension de cette notion.

Les enseignants libérés des tâches répétitives

En assurant lui-même le choix des exercices, leur correction et l'évaluation permanente des élèves, l'enseignant se trouve libéré des tâches répétitives et devient plus disponible, d'où la possibilité :

— D'accentuer une recherche didactique pour la présentation des notions à acquérir (diversification des moyens de présentation, approches plus personnalisées en petits groupes) ;

— De déceler rapidement les difficultés particulières de certains élèves et de développer en conséquence une pédagogie de soutien, l'enseignant pouvant revenir avec certains, sur la présentation d'une notion mal assimilée, puis éventuellement, grâce à l'usage des séances sur un thème donné (les « fixed strands »), pour compléter la pratique de cette notion en priorité par rapport au reste du programme.

Insistons à ce propos sur le fait que le système DIDAO ne doit pas être envisagé comme une pédagogie de soutien, mais facilite la mise en place de celle-ci par l'enseignant.

Une familiarisation avec l'informatique

Enfin, pour les enfants comme pour les enseignants, l'expérimentation de ce système permet une familiarisation avec l'environnement informatisé qui semble devoir se développer au cours des prochaines années. Et la possibilité offerte d'une initiation au langage BASIC devrait favoriser une participation de plus en plus active des enseignants à la conception et à l'amélioration des programmes.

La présentation très brève de cette expérience montre assez bien les avantages et les limites d'un système d'enseignement assisté par ordinateur du type enseignement programmé dans un domaine où beaucoup de choses restent à découvrir et plus particulièrement, tout ce qui touche au développement cognitif de l'enfant.

Pour conclure, il semble intéressant de citer le Pr. J.-C. Simon : *« Toutefois, il ne faut pas considérer l'introduction des moyens modernes comme un remède-miracle. Ces moyens ne seront efficaces que dans la mesure où le professeur sera convaincu de leur intérêt. Ce n'est que dans cette mesure qu'ils pourront renouveler la pédagogie en l'améliorant. »*

Corinne Le Roy

Une procédure automatique pour jouer aux échecs (suite et fin)

J.-L. LAURIÈRE

L'évaluation des positions

Il s'agit maintenant de faire correspondre à toute situations une valeur numérique f(s) d'autant plus grande et positive que le camp ami a l'avantage, de valeur de l'ordre de 0 si il y a à peu près égalité sur la position, d'autant plus grande et négative que l'adversaire domine. Bien sûr, il n'existe aucune théorie aux échecs donnant cette fonction f exactement. Tout au plus connaît-on, pour certains débuts répertoriés, le signe du résultat. Pour tous les autres cas tout est à inventer. Dès lors, l'évaluation de certaines positions pourra malencontreusement être jugée négative alors qu'on a l'avantage. C'est là une raison fondamentale pour laquelle un programme de ce type ne peut jouer parfaitement. Avec cette approche, à chaque fois que le programme joue mal il faut trouver la raison dans la fonction f et changer l'expression de celle-ci de façon à corriger l'erreur.

La partie la plus importante dans l'évaluation est toujours le matériel et le premier terme de f est proportionnel à la somme algébrique des valeurs des pièces de part et d'autre, appelée *balance* B. Les valeurs retenues des pièces aux échecs sont habituellement, comme pour les joueurs humains : Pion 1, Cavalier et Fou 3, Tour 5 et Dame 10. Puis, d'autres termes, concernant chacun une caractéristique différente, viennent s'ajouter à f pour donner une forme polynômiale du type :

$$f(s) = a.B + b.R + c.M + d.C + e.P + f.A$$

Si le camp ami a un fou d'avance pour un pion de retard le premier terme vaut ainsi 2.a. La valeur du coefficient a, comme celles de b, c, d, e et f est laissée à « l'astuce » de chaque programmeur... les termes suivants sont grosso modo d'importances décroissantes :

La valeur R tient compte des sûretés relatives des deux *rois*. R est d'autant plus grand que le roi ami est bien protégé derrière son roque. Le coefficient b doit varier en cours de partie pour aller jusqu'à s'annuler lorsque les pièces majeures ont disparues de l'échiquier, les rois devant alors soutenir leurs pions ou encore aider l'attaque en protégeant leurs propres pièces.

Le terme M mesure la *mobilité* des pièces : il récapitule le nombre de cases où chacune des pièces peut se déplacer, et il pondère ce nombre par un facteur dépendant de la nature de la piece.

Le terme suivant C favorise le *contrôle du centre,* c'est-à-dire des cases d4 et d5, e4 et e5. Le coefficient d, qui lui est associé, n'a une valeur élevée qu'en début et en milieu de partie.

La caractéristique évaluée par P concerne la *structure des pions :* les pions amis protégés avancés ou passés interviennent positivement pour le camp ami, tout comme les pions ennemis doublés, isolés ou non protégés.

Enfin (mais la liste donnée n'est pas limitative et quelques programmes prennent en compte d'autres caractéristiques) le dernier terme, en A, fait le bilan des possibilités d'*attaques* en tenant compte des pièces avancées au-delà de la quatrième ligne et protégées, des tours sur des colonnes ouvertes, des tours doublées, des possibilités d'échange contre des pièces adverses de grande mobilité, des attaques sur les cases voisines du roi ennemi, des possibilités d'échecs et enfin des clouages.

La mise au point des coefficients de pondération de a à f est une affaire de patience et rien n'interdit de faire varier leurs valeurs en cours de partie en fonction du nombre de coups joués depuis le début et/ou en fonction des pièces restant sur l'échiquier. Il est alors, on le voit, *dans la nature même de la fonction d'évaluation de permettre des sacrifices positionnels* c'est-à-dire d'amener des positions où la caractéristique B a une valeur négative (une pièce en moins que l'adversaire) mais dans laquelle la somme pondérée des autres termes compense avec bénéfice cette différence défavorable de matériel.

Si la fonction f était parfaite, il suffirait d'engendrer tous les coups amis légaux d'évaluer les situations auxquelles ils conduisent et de jouer celui qui correspond à la meilleure d'entre elles. En ratique, on est loin de savoir construire d'aussi bonnes fonctions. Comme par ailleurs on ne peut développer l'arbre assez loin pour atteindre des positions où l'avantage de l'un ou de l'autre est indiscutable, il faut *trouver un compromis* entre ces deux extrêmes, *compromis entre la profondeur (moyenne) de la recherche et la qualité de la fonction d'évaluation*. En effet augmenter chacune de ces grandeurs contribue pareillement à la croissance du temps de calcul !

En tout état de cause, un même problème doit encore être traité qui consiste à savoir déduire des positions terminales atteintes à profondeur maximale, le coup à jouer effectivement au premier niveau. La méthode qui fait l'unanimité depuis que les machines ont été programmées pour jouer, et quelque soit le jeu, est connue sous le nom de *minmax*.

Le minmax et le choix du coup à jouer :

Remarquons tout d'abord que, précisément parce que notre fonction d'évaluation ne saurait être parfaite dès la position initiale, il ne servirait à rien d'évaluer les positions intermédiaires à mesure qu'elles sont rencontrées en cours de descente de l'arborescence de recherche, tant que la profondeur maximale choisie, soit n, n'est pas encore atteinte : ces positions risquent fort, notamment, d'être instables et d'évaluation incommode. Une fois parvenu à la profondeur n il convient de prendre encore en compte toutes les captures, dans l'ordre convenable de valeur des pièces, et les échecs. C'est alors seulement, en parcourant en arrière l'arborescence des coups, que nous attribuons des valeurs aux sommets qui précèdent chronologiquement les situations associées aux feuilles. Mais justement comment procéder ? Oscar Morgernstern et John von Neumann avaient, en 1945, proposé une procédure valable pour tout jeu de stratégie. L'hypothèse fondamentale supplémentaire ici sera que l'adversaire s'en remet à la même fonction d'évaluation que nous mêmes. Dès lors, *maximiser* la valeur de la fonction d'évaluation de la situation courante, en revanche, il convient pour notre adversaire de *minimiser* celle-ci.

Ainsi, à tout niveau où c'est à lui de jouer, dans le pire des cas l'adversaire retiendra le coup qui mène à la situation d'évaluation la plus faible. Tout se passe donc comme si l'évaluation à ce niveau, avant que l'adversaire ait joué, était le *minimum* des évaluations qui lui succèdent immédiatement. Quand le niveau, au contraire, correspond à un trait ami, c'est le *maximum* des évaluations un niveau plus bas, qui doit être retenu, d'où le nom de minmax donné à la procédure de von Neumann et Morgenstern. De proche en proche tous les nœuds intermédiaires reçoivent ainsi une

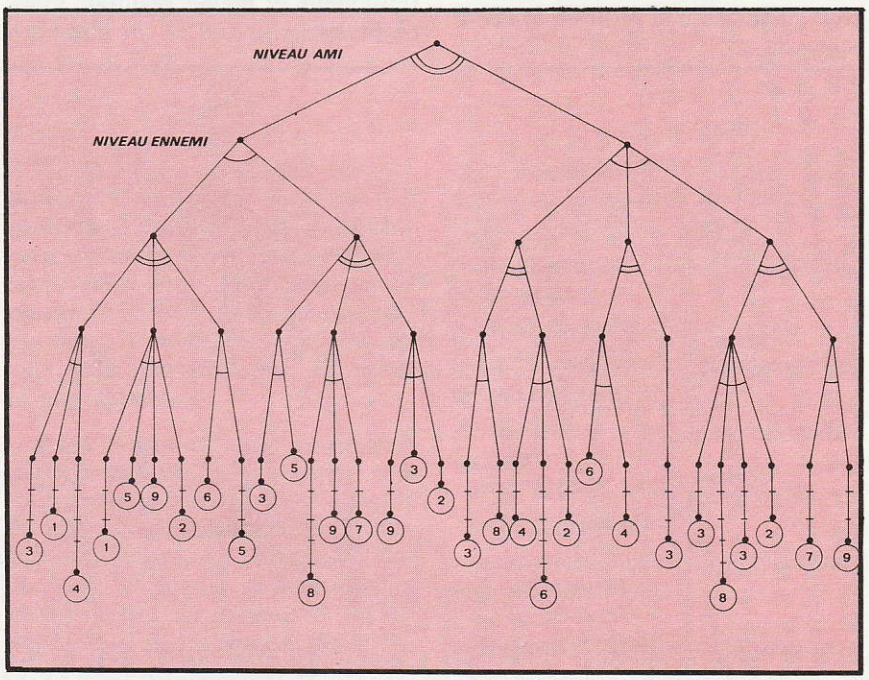


Fig. 4 : Une arboresscence des coups légaux avec évaluations à profondeur variable des positions stables.

évaluation. Quand tous les successeurs d'un nœud sont connus, la valeur maximum ou minimum d'entre eux, suivant la parité du niveau, lui est finalement attribué. Remontant ainsi les évaluations jusqu'à la racine de l'arborescence c'est-à-dire jusqu'à la position initiale, on en déduit, au bout du compte, le meilleur coup ami, qui correspond à la position au premier niveau dont l'évaluation *remontée* est la plus forte. Il est important de noter que cette évaluation n'a a priori absolument rien à voir avec celle obtenue directement en appliquant f à la position sans tenir compte des coups ultérieurs : on a bien une *évaluation corrigée* par l'étude en profondeur et remontée par minmax.

La figure 4 donne un exemple d'étude et de résultat fourni par le minmax. Deux coups légaux seulement, soient c1 et c2, se présentent sur la position initiale où le trait est au camp ami. A partir de la position p1, atteinte après le coup c1, on engendre tous les coups légaux possibles soient c3 et c4. Etudiant d'abord la variante associée à c3, on engendre à nouveau tous les coups à partir de la situation correspondante. Trois coups c5, c6 et c7 sont ainsi produits. Le coup c6 mène à une position où trois autres coups c8, c9 et c10 sont à leur tour engendrés.

Supposons que quatre soit, pour notre programme, la profondeur maximale. Chaque situation après c8, puis c9, puis c10, est donc étudiée, jusqu'à l'élimination des prises et des principaux échecs au roi et est alors évaluée. Les évaluations données par la fonction f sont portées dans la figure sur la dernière ligne ; elles sont entourées d'un cercle. Si le coup ami a le trait sur la position initiale c'est le camp ennemi qui avait le choix du coup au dernier niveau développé : des trois évaluations associées à savoir 3, 1 et 4, il retiendra donc, du moins s'il joue avec la même évaluation que nous, la plus faible soit 1.

La valeur 1 devient donc celle de la situation obtenue si nous prenons la décision de jouer le coup c5 au troisième niveau. De la même façon, nous ne pouvons compter après le coup c6 sur un résultat de

valeur 9 puisque l'adversaire a, encore une fois, une riposte qui lui assure une position où il peut limiter nos ambitions à un score d'une unité. Le coup c7, enfin, est à ce niveau légèrement supérieur, sous les hypothèses correspondant au jeu des coups des niveaux un et deux, puisque, au mieux, notre adversaire atteint une position de valeur 2. Les trois positions issues de la position amenée par c1 puis c3 ont maintenant reçu une valeur. Or, dans cette position, nous avons le trait : nous ferons donc en sorte, si cette position se présente dans la suite de la partie réelle, de jouer le coup qui maximise la valeur atteinte : entre 1, 1 et 2 nous retiendrons 2, qui correspond au coup c7. Cette valeur devient donc l'évaluation, remontée par minmax, associée à la position après c1 suivi de c3.

et retenir en mémoire les coups qui, à chaque niveau, ont déjà été examinés. Au total, si le temps d'exécution peut être important, car il croît très vite, nous l'avons vu, avec la profondeur de recherche, la *place nécessaire en mémoire* reste toujours quant à elle relativement *faible* : toute la partie de l'arborescence et tous les échiquiers qui ne correspondent pas à la seule suite des coups présentement examinés, n'ont pas à être mémorisés ; seule la meilleure valeur numérique à chaque niveau est utile. C'est pourquoi même des microordinateurs bon marché mènent cette tâche à bien.

Il existe une procédure, équivalant théoriquement au minmax, puisqu'elle fournit toujours rigoureusement le même résultat, mais nettement *plus rapide* car elle permet d'éliminer définitivement, sans avoir à les examiner, des parties entières de l'arborescence. Cette procédure utilisait deux variables nommées alpha bêta dans la publication où elle fut pour la première fois décrite en 1961 d'après une idée de John Mac Carthy ; elles sont à l'origine de son nom curieux.

Mais, même ainsi améliorée, la procédure du minmax reste entachée d'un défaut essentiel : une partie est jouée coup à coup *sans aucun plan d'ensemble.*

La force de l'homme réside précisément dans sa capacité de concevoir, de développer, d'adapter des stratégies.

Un autre type de procédure, plus proche du raisonnement humain, est ainsi programmable. Les chercheurs en « intelligence artificielle » s'efforcent aujourd'hui d'écrire des programmes de cette autre famille : dans le cas du jeu d'échecs c'est toute la connaissance des joueurs accumulée depuis des siècles, qui doit être donnée ; elle s'exprime en termes de clouage, d'attaque double, de colonne ouverte, de faiblesse du roque, etc. Tous ces termes doivent être compris et manipulés par le programme.

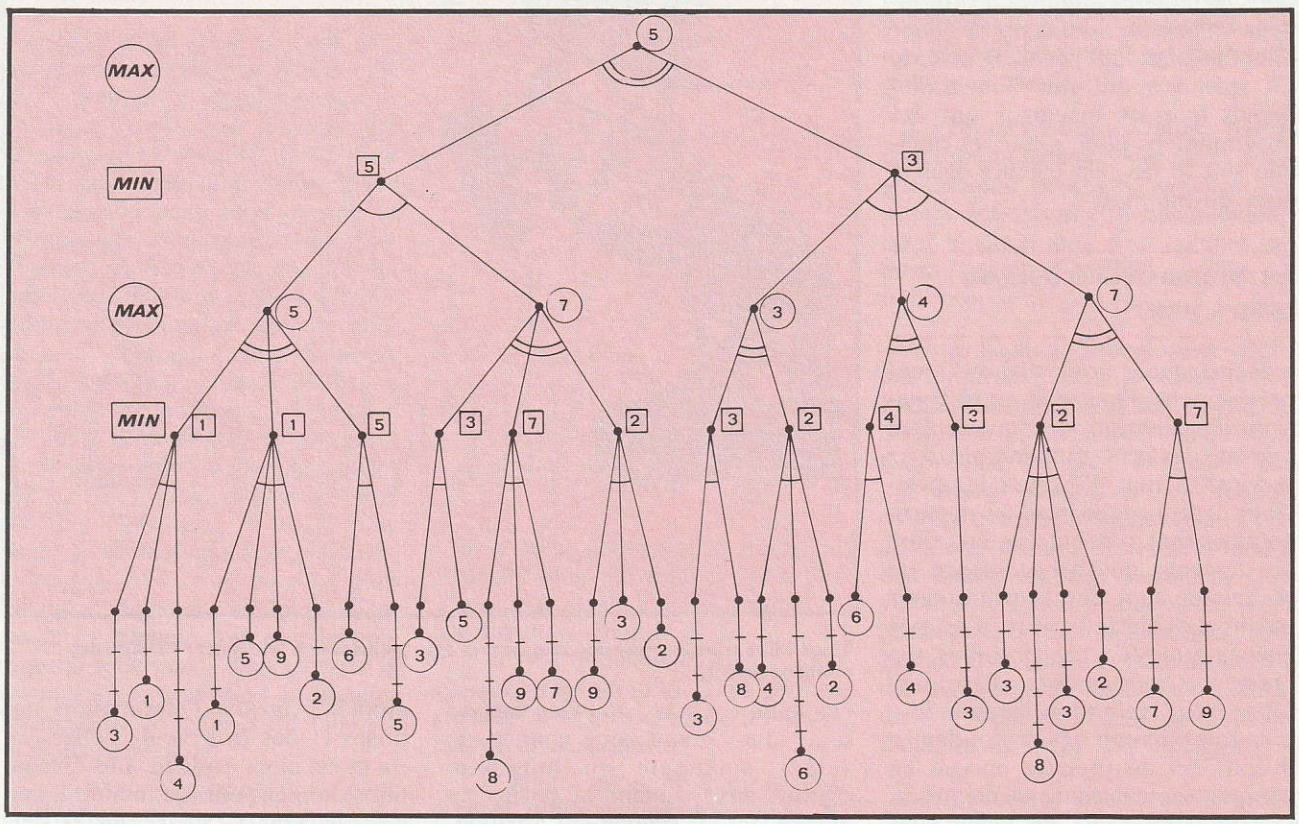


Fig. 5 : Remontée des évaluations par minimax pour l'arborescence précédente.

La suite et la fin de la procédure s'effectuent pareillement et la figure 5 donne les résultats. Sur cette figure, on trouve, à côté de chaque position intermédiaire, l'évaluation calculée par le minmax. En particulier l'évaluation de la position initiale vaut 1.

Remarquons que, dans cette procédure, il convient, lors de la descente dans l'arborescence de jouer les coups, et les « déjouer » lors de la remontée. Mais il suffit pour cela de *tenir à jour l'échiquier*

De tels programmes basés sur la connaissance et non plus sur la force brute commencent à exister dans plusieurs autres domaines : chimie, médecine, mathématiques.

J.-L. Laurière

les nouvelles technologies dans l'éducation

Un nouveau rapport américain

par J. HEBENSTREIT

Moins de 8 ans après la parution du célèbre rapport de la Commission Carnegie : The Fourth Revolution -Technology in education (La quatrième révolution - La Technologie dans l'éducation), publié chez McGraw Hill en 1972, la National Science Fondation publie un nouveau rapport sur ce même thème.

La comparaison des deux rapports est d'autant plus instructive qu'ils diffèrent assez peu quant aux principes qu'ils énoncent puisque les finalités de la technologie dans l'éducation et les modes d'utilisation recommandés de cette technologie sont assez voisins.

Le rapport Carnegie était extrêmement pondéré dans le fond comme dans la forme et la plupart des prévisions portaient sur la période 75-85.

Le nouveau rapport « Technology in Science Education : The next 10 years - Perspectives and Recommendations » date de juillet 1979 et il est nettement plus percutant.

Il est plus percutant d'abord parce qu'il se présente sous la forme de 4 textes écrits par des spécialistes en technologie de l'éducation. De ce fait, il permet l'expression d'opinions individuelles souvent vigoureuses et évite le phénomène « d'opinion moyenne » qui caractérise les compromis propres à tous les travaux de groupe.

Il est plus percutant ensuite parce qu'il insiste sur l'urgence des décisions à prendre, non seulement pour la technologie dans l'éducation, mais aussi pour aider à la naissance d'une industrie correspondante (matériels, logiciels, didacticiels), et enfin pour informer le grand public.

Il est percutant enfin parce qu'il n'hésite pas à dire clairement que toute absence de décision dans ce domaine aura nécessairement des répercussions sur l'égalité des chances, le niveau de l'emploi, la compétitivité de l'industrie, le niveau de la recherche, la place des USA dans le monde et la défense du territoire.

Précisons enfin, avant d'entrer dans le vif du sujet, que ce rapport résulte d'un ensemble de réunions organisées par la NSF en 1977-78 avec l'objectif de lui fournir un ensemble de recommandations destinées à orienter son activité dans le domaine de la technologie de l'éducation pour les 10 prochaines années.

Le premier et le plus important des 4 textes est celui de J.C.R. Licklider du MIT, intitulé : « L'impact de la technologie de l'information sur l'enseignement des sciences et de la technologie ». Après avoir rappelé les progrès rapides de la technologie qui ont permis d'aboutir à des ordinateurs rapides, petits et d'un prix accessible, l'auteur indique les domaines dans lesquels l'informatique peut améliorer la qualité de l'enseignement, à savoir :

— la pratique des exercices sur ordinateur

— l'enseignement tutoriel

— l'enseignement individuel géré par ordinateur

— l'aide à la résolution des problèmes

— l'aide à la modélisation

— la simulation

— l'encouragement à l'autonomie des étudiants.

Enfin, il suggère l'établissement de « banques de connaissances » gérées par ordinateur et incluant les techniques récentes de réseaux sémantiques, de réseaux relationnels, de scénarios et de langages spécialisés dans la représentation des connaissances.

Pour l'auteur, le danger majeur de l'introduction de l'informatique dans l'enseignement est lié au fait que « ce qui est le plus facile à faire et aussi le plus générateur de profits n'est, en général, pas du tout ce qui devrait être fait dans l'intérêt de la société ».

Il propose, en conséquence, de développer en priorité les applications dont les avantages sont évidents et les dangers faibles, telles que la résolution de problèmes, la modélisation, la simulation, l'encouragement à l'autonomie des élèves, et de freiner les applications

de type tutoriel dans lesquelles on se contente de recopier les manuels existants sur des disquettes.

L'article de John Seely Brown (Xerox, Palo Alto Research Center) traite de « Recherches fondamentales sur la technologie dans l'enseignement des sciences ». Pour l'auteur, l'accroissement considérable de la puissance des ordinateurs mis à la disposition des enseignants doit entraîner un bond qualitatif pour l'utilisation des ordinateurs dans l'enseignement. Il propose, en conséquence, un programme de recherches en 3 points visant « à la fusion de l'informatique et des sciences cognitives » :

— Définition de langages puissants permettant la représentation des connaissances et destinés à remplacer les langages de programmation actuels beaucoup trop rudimentaires.

— Jusqu'à quel point peut-on simuler sur ordinateur l'expérience et les modes de raisonnement d'un expert en physique et/ou en mathématiques permettant ainsi à l'ordinateur de déceler et de corriger les erreurs méthodologiques faites par un étudiant lors des tentatives de solution d'un problème ?

— Peut-on programmer un ordinateur de façon à le rendre capable de porter un jugement sur le niveau et la qualité des connaissances d'un étudiant et, à partir de là, d'aider l'étudiant à progresser ?

L'auteur fait ici visiblement allusion aux travaux de recherche qu'il poursuit avec R.R. Buton et qui ont abouti au programme BUGGY (ce programme, après avoir proposé une série d'additions à un élève, serait capable par la seule analyse des résultats de diagnostiquer les erreurs de *méthode* commises par l'élève).

L'auteur note également qu'il est plus difficile de communiquer avec un ordinateur qu'avec un humain et cela parce que l'ordinateur n'a pas de « vision du monde » et qu'il ne peut, de ce fait, accepter les finesses et les métaphores qui permettent de préciser la pensée. Il est intéressant de noter que, contrairement à un certain nombre d'informaticiens qui font de nécessité vertu et prônent la valeur formatrice d'un dialogue avec la machine qui doit, chaque fois, partir de zéro. J.S. Brown s'insurge contre cet état de fait et propose, au contraire, de chercher ce qu'il faudrait fournir à la machine afin de lui faire partager notre « vision du monde » et donc de rendre la communication avec la machine aussi facile que la communication avec un être humain.

Dans ce contexte, l'auteur propose une généralisation des techniques pédagogiques de simulation qu'il appelle « l'exploration des micro-mondes ».

Un « micro-monde » est un ensemble d'objets plus ou moins complexes doués d'un grand nombre de propriétés cohérentes, et simulé sur ordinateur.

Le micro-monde peut être une partie de la réalité ou complètement imaginaire. L'étudiant doit découvrir la structure et les lois de ce micro-monde en faisant, des expériences, puis des hypothèses menant à de nouvelles expériences, etc. Ce type d'activité aurait pour effet, selon l'auteur, de développer des démarches cognitives fondamentales : stratégies de recherche d'information, optimisation de ces stratégies, construction d'hypothèse, recherche et découverte de contre-exemples, etc.

Les grandes lignes du rapport Carnegie

Le Rapport de la Commission Carnegie intitulé « La quatrième révolution » rédigé par un groupe de 19 experts, après 3 années de travail, a été publié en juin 1972 sous l'égide de la Fondation Carnegie pour l'avancement de l'éducation.

L'expression « Quatrième révolution » est due à Ross Ashhy qui distingue dans l'évolution :
- *— la première révolution : remplacement (partiel) de l'éducation par et dans la famille par l'école*
- *— la deuxième révolution : utilisation de l'écriture comme outil pour l'éducation en plus de l'engagement oral*
- *— la troisième révolution : utilisation du livre, après l'invention de l'imprimerie, comme outil nouveau*
- *— la quatrième révolution : utilisation de moyen électroniques comme outils supplémentaires, à savoir la radio, la télévision, le magnétophone, le magnétoscope et les ordinateurs.*

En une centaine de pages, le rapport fait une étude extrêmement profonde des possibilités et des limites de la technologie dans l'éducation tout en restant très nuancé dans ses conclusions.

« Nous pensons que la technologie doit être la servante et non la maîtresse de l'instruction. Elle ne doit pas être adoptée simplement parce qu'elle existe ou parce qu'une institution a peur d'être dépassée par le progrès si elle ne l'utilise pas. Nous pensons aussi que l'utilisation de la technologie ne doit pas être confondue avec la saturation. Dans certains cours, l'utilisation de la technologie peut n'être appropriée que pour quelques heures sur toute l'année ; pour d'autres cours, la technologie peut être utilisée de manière plus intensive ; enfin, dans certains cas, la technologie pourra être le support unique du cours ».

« Il faut cesser de braquer les projecteurs sur la nouvelle technologie et raisonner plutôt en termes de systèmes d'éducation intégrant la nouvelle technologie, la technologie classique, la théorie de l'apprentissage l'organisation physique de salle de classe et l'enseignement en un ensemble cohérent. »

« Si nous voulons que la technologie réponde aux besoins de l'éducation, la plus grande priorité doit être donnée pendant les deux prochaines décades au développement de didacticiels de bonne qualité, utilisables par le plus grand nombre possible d'institutions. »

« Il existe des possibilités d'usage abusif de dispositifs technologiques coûteux et complexes. Pour les éviter, il faut :
- *— que les matériels et équipements soient conçus, sélectionnés et utilisés pour remplir des tâches d'enseignement spécifiques tout en offrant les meilleures conditions possibles d'apprentissage aux élèves*
- *— qu'il y ait une correspondance étroite entre les possibilités de la technologie et les besoins de l'enseignement*
- *— que sans céder à la mode, on sélectionne, pour chaque tâche d'enseignement, l'équipement le moins complexe et le moins coûteux qui donne satisfaction. »*

Le texte est tellement dense et tellement actuel encore en 1981 que tout serait à citer y compris les 25 recommandations. Le chapitre final propose une série de « buts raisonnables » pour 1980 et 1990, permettant, en l'an 2000, d'envisager un réseau national de communications et de ressources pédagogiques permettant à tout étudiant d'avoir accès à toute heure et en tous lieux et sous une forme assimilable, à la totalité des connaissances humaines accumulées.

Cette technique des « micro-mondes » peut sembler ambitieuse mais, en plus de ses qualités pédagogiques, il faut souligner que cette technique est extrêmement motivante comme le prouve le succès sans précédent remporté aux USA par les jeux sur ordinateurs du type « Adventure land » (Ce jeu simule un micro-monde rempli de géants et de dragons contre lesquels il faut se battre, de trésors et de caches d'armes à découvrir, de princesses à délivrer, etc., en bref un conte de fées interactif où le joueur à son clavier n'est autre que le héros du conte).

L'article d'A. Luerhamann (Université de Californie à Berkeley) sur « La technologie dans l'enseignement des sciences » suggère que « le faible niveau d'utilisation de moyens technologiques dans l'éducation des USA et les systèmes technologiques destinés à l'éducation ; à quelques exceptions près, personne n'a trouvé le moyen d'intégrer cette technologie au système éducatif ».

L'auteur ajoute que « la promotion active de moyens technologiques dans l'éducation par les Agences Fédérales (des USA) est ressentie par le système d'éducation comme imposée de l'extérieur et comme une menace ». Il poursuit en affirmant que « les dispositifs technologiques les mieux connus et les plus efficaces ne pourront être utilisés dans le système d'éducation, sans modifications révolutionnaires de celui-ci et c'est pourquoi il faut s'attendre à ce que le système existant les rejette ».

Partant de la constatation que « *toutes* les applications de la technologie ne sont cependant pas en conflit avec le système existant », l'auteur consacre son article à la spécification de ces domaines et termine par cinq recommandations :

— Définition d'un curriculum permettant aux élèves de se familiariser avec l'usage des ordinateurs.

— Formation des enseignants

— Création de Centres de développement rassemblant des enseignants de talent afin de définir les modes d'utilisation des ordinateurs dans toutes les disciplines

— Utilisation des bibliothèques et des musées pour informer un vaste public sur les ordinateurs et leur utilisation.

— Développement de système d'enseignement individuels basés sur l'utilisation de « vidéo-disques intelligents ».

Le dernier article est de J.I. Lipson de la National Science Foundation et il est intitulé « Recommandations pour un programme en faveur de la technologie ». L'auteur utilise deux arguments majeurs en faveur de la technologie dans l'enseignement :

— les USA sont pour le moment la nation dominante en informatique et cette technique jouera un rôle décisif dans l'avenir.

Cette position dominante est actuellement menacée par le Japon et d'autres pays. Si le Gouvernement, l'industrie et les Universités n'unissent pas leurs efforts dans le domaine des ordinateurs et des activités informatiques, les USA perdront leur primauté dans ce domaine comme ils sont en train de la perdre en électronique ou dans l'industrie automobile. Un facteur clé pour le succès dans ce domaine sera la qualité de l'enseignement en général, l'utilisation des ordinateurs et des technologies annexes pour améliorer cet enseignement et le niveau moyen de compétence de la population dans l'utilisation des ordinateurs.

— L'utilisation d'ordinateur dans l'enseignement peut améliorer considérablement la qualité de cet enseignement : « Ces dispositifs permettent aux gens de contrôler le temps et l'espace. En montrant aux étudiants de nouveaux univers, la technologie peut faire acquérir une expérience qui développe l'intuition et sur laquelle pourront être bâtis des concepts plus abstraits. L'ordinateur, par la modélisation et la simulation, permettra à l'étudiant d'accéder à un monde dans lequel il doit apprendre en vue de l'explorer et de le maîtriser... La puissance de la simulation par le contrôle qu'elle permet sur les variables et les possibilités de prédiction qu'elle offre donne à l'enseignement des possibilités qui n'ont jamais existé. Grâce à l'ordinateur, l'étudiant dispose d'outils intellectuels nouveaux et d'une puissance jamais atteinte. De nouvelles images peuvent être créées, de nouveaux modèles peuvent être construits et testés. Symétriquement, l'ordinateur et les dispositifs visuels connexes peuvent aider les étudiants peu favorisés, les handicappés, les étudiants âgés ou isolés, en mettant à leur disposition de nouveaux univers d'action, de couleur, d'art, d'efforts créateurs et d'apprentissage ».

Par la suite, l'auteur souligne que l'industrie privée seule ne pourra pas assurer le développement des nouvelles technologies destinées à l'éducation. Il invite, en conséquence, le Gouvernement Fédéral à intervenir dans ce domaine qu'il considère comme d'intérêt national puisqu'il a des retombées directes sur : l'égalité des chances, le chômage, la qualification de la main-d'œuvre, la défense nationale et le développement industriel et scientifique.

En conclusion, l'auteur propose une série de recommandations :

— Définir une politique des télécommunications et promouvoir l'utilisation des nouvelles technologies dans l'éducation.

— Développer des programmes de formation destinés à faire surgir les nouveaux talents qui sont nécessaires pour utiliser les nouvelles technologies dont les possibilités sont déjà en train de dépasser nos capacités d'imagination.

— Participer à la diffusion la plus large d'informations sur les nouvelles technologies en y incluant des programmes de formation des enseignants.

— Aider à l'information et à l'éducation du public sur les nouvelles technologies en utilisant les medias appropriés.

— Financer le développement de didacticiels prototypes de très haut niveau pour enseigner à la fois les connaissances et les compétences dans les environnements nouveaux que la technologie permet de réaliser.

— Financer des recherches sur les problèmes d'apprentissage et, plus particulièrement, sur les problèmes soulevés par les nouvelles technologies que ces mêmes technologies permettent de résoudre.

— Financer l'achat d'équipements informatiques en vue de promouvoir des démonstrations convaincantes, aider à la naissance d'une industrie de la technologie pour l'éducation et permettre l'accès aux ordinateurs aux parties de la population qui n'en ont pas les moyens.

J. Hebenstreit

Un premier bilan de l'expérience des 58 lycées

Menée entre 1970 et 1980 dans cinquante-huit lycées, l'expérience d'introduction de l'informatique dans les établissements de second degré, fait l'objet, depuis 1976, d'une opération d'évaluation. Celle-ci, confiée à l'équipe réunie autour de Christian Lafond et Pierre Muller à l'INRP, est terminée. Un rapport va être remis à M. Saurel, directeur des lycées qui fait le bilan d'une expérience originale.

Éducation et Informatique : Quel est, précisément, l'objet du rapport que vous avez été chargé de réaliser ?

Pierre Muller : Le rapport de 340 pages (sans compter les annexes) que nous venons de terminer, devait procéder à une évaluation de l'expérience des 58 lycées, afin de conclure — ou non — à la nécessité de généraliser celle-ci. Durant les deux premières années de notre travail, nous avons tenté de réaliser un inventaire de ce qui s'était fait dans les lycées. Puis, de 1978 à 1980, nous avons centré notre étude sur l'utilisation de l'informatique dans huit classes, de huit établissements différents, où s'était créée une « ambiance informatique ». Dans ces classes, l'informatique a permis une interdisciplinarité, et des conditions, pour ainsi dire, « expérimentales » ont été réunies.

Que conclure de tout cela ? Tout d'abord, constatons que 90 000 élèves, dans les cinquante-huit lycées, ont eu accès à la salle des ordinateurs en dix ans. Dans ces cinquante-huit lycées équipés, un élève sur deux, et un enseignant sur six, ont fréquenté la salle des ordinateurs, au moins une fois. Les centres ont fonctionné en moyenne 32 heures par semaine.

Si l'on s'en tient à cet aspect quantitatif, disons encore qu'une bibliothèque de quatre cent logiciels a été constituée à l'INRP, sans même parler des programmes faits localement, et qui ne sont pas remontés jusqu'à nous. Au début de l'opérations, deux objectifs étaient visés : démythifier l'informatique, et introduire celle-ci dans les différentes disciplines d'enseignement. Ces deux objectifs ont été atteints. Et, au-delà, ce qui est un motif de satisfaction, c'est que les enseignants ont géré eux-mêmes les centres informatiques, sans le secours de techniciens, dans un esprit d'inter-disciplinarité.

Maintenant, l'impact d'une expérience est toujours difficile à mesurer. Que deviendra tout cela, une fois que l'informatique sera imposée aux élèves, que les enseignants ne seront plus, comme ça a été le cas jusqu'à présent, volontaires ? Il est difficile de le dire. Mais telle quelle, l'expérience permet de préciser un certain nombre de choses.

E & I : On a beaucoup parlé des effets pédagogiques de l'introduction de l'informatique dans les lycées. Qu'en est-il ?

Pierre Muller : Nous avons étudié les motivations des élèves, et nous avons constaté qu'elles étaient fortes, qu'elles ne diminuaient pas en cours de route. Dans les questionnaires que nous avions envoyé aux établissements, on voit que les élèves sont attachés à l'informatique autant parce qu'ils y voient un outil efficace, un moyen d'acquérir une certaine liberté par rapport au contrôle de l'enseignant, et enfin un jeu. Le dialogue avec l'ordinateur diminue la culpabilité de l'élève en cas d'erreur, permet un apprentissage plus individualisé. Mais les élèves sont très conscients des limites de l'informatique. Ils sont soucieux de contacts humains et ne souhaitent pas un enseignement complètement automatisé.

E & I : Les enseignants, durant l'opération, ont réalisé eux-mêmes les logiciels. Qu'en dit le rapport ?

Pierre Muller : Comme je vous le disais, quatre cent logiciels ont été réalisés, dans toutes les disciplines. Logiciels tutoriels, programmes de simulation, de jeux... toute la gamme. Ce qui apparaît, c'est que les enseignants sont soucieux de la souplesse des logiciels, ils souhaitent pouvoir les adapter aux besoins de leur enseignement. De ce point de vue, il est frappant de voir que sur les 400 logiciels collectés par l'INRP, une vingtaine représente la moitié des utilisations : ce sont les plus fiables et les mieux adaptables. Certains logiciels, difficiles d'emploi, sont ignorés en dépit de leur qualité. C'est normal : il y a souvent une urgence, et un manque de temps, qui obligent à parer au plus pressé. Mais on ne doit pas conclure qu'il faille diffuser uniquement des logiciels d'utilisation facile. Il y a un problème de formation des maîtres qui doit être résolu, de telle sorte que les enseignants puissent utiliser des logiciels plus compliqués, mais plus novateurs

La formation devra aussi tenir compte des limites de l'expérience des 58 lycées. Car il faut constater que cinq professeurs sur six, durant le temps de celle-ci, n'ont pas fréquenté la salle informatique. Ils n'en connaissaient même pas l'existence, ils n'y sont jamais allé, même pour voir.

E & I : La généralisation est-elle possible ?

Pierre Muller : Sans doute, mais il faudra alléger les obligations administratives, organiser la coordination nationale — et cela suppose des moyens —, tenir compte des contraintes matérielles, des limites de fonctionnement des salles informatiques. Pour des conditions optimales d'utilisation, ce ne sont pas dix mille, mais quarante mille micro-ordinateurs qu'il faudrait mettre en service.

Propos recueillis par Marc Coutty

Un millier d'enseignants à la journée CEDIC/NATHAN

Maryvonne Corbel

Il a fait très chaud le mercredi 13 mai dans le salon Récamier de l'hôtel Lutécia à Paris... où se tenait la journée débat organisée par les Éditions CEDIC/NATHAN sur le thème : les outils d'éducation « Livre scolaire » et « micro-ordinateur ». Des centaines de participants, avides d'informations ont permis de généraliser un débat dont le professeur Jean-Claude Simon a bien voulu accepter d'être la cible...

L'exposition, centrée sur les productions scolaires mathématiques et informatiques, a vu son succès confirmé par la présentation au public des didacticiels de mathématiques devant être prochainement commercialisés. Ces derniers ont suscité un vif intérêt. En ce qui concerne les livres, la collection classique s'est vue enrichie de deux nouveaux ouvrages, présentés par A. Deledicq et M. Glayman :
— Faire des mathématiques 6e (Nouvelle collection) par A. Deledicq et Lassave
— Mathématiques 2nde, par M. Glayman et J. Malaval

Puis, vers 16 h, les participants se sont regroupés pour assister au débat dont nous ne citerons ici que les idées essentielles exprimées. Dans un exposé introductif, Jean-Claude Simon a rappelé les points forts de son rapport, ramenant l'ordinateur à des « dimensions humaines », c'est-à-dire, à un produit technologique d'utilisation quasiment courante de nos jours, cela afin de souligner la nécessité d'adapter l'éducation à l'évolution technologique.

Hôtel Lutétia - L'exposition CEDIC/NATHAN « Livre scolaire »

Il a ensuite insisté sur le double aspect de l'informatique, à la fois en tant que science et en tant que moyen, distinguant l'utilisation des ordinateurs à des fins pédagogiques pour toutes les disciplines, appelée informatique « transparente » ou « presse-bouton », et l'initiation à l'informatique proprement dite.

Sur le premier point, J.-C. Simon dégage ce qui lui semble l'essentiel des expériences d'EAO entreprises et de celle des « 58 lycées » :

— *« Les moyens modernes de télécommunications, audiovisuels et informatiques doivent tous être utilisés parce que tous peuvent apporter une sérieuse amélioration des moyens pédagogiques actuels, notamment en ce qui concerne : l'enseignement dispensé aux personnes handicapées physiques, le rattrapage des capacités de base, l'enseignement professionnel. »*

Notons cependant que tout acte d'enseignement ne passera pas forcément par une démonstration sur écran...

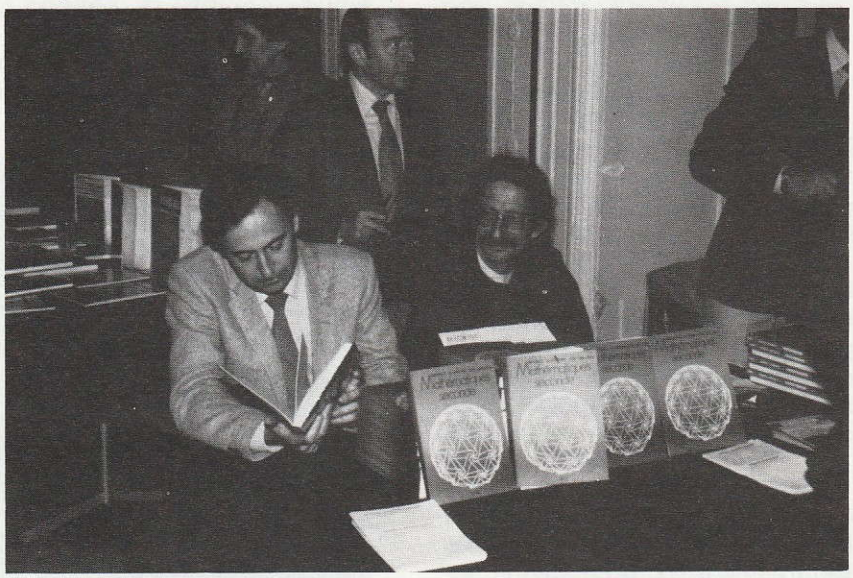
— Tous les enseignants ne pourrons pas faire eux-mêmes leurs didacticiels à moins d'avoir beaucoup de temps libre et d'être très passionnés... Une heure de cours représentant environ 100 heures de travail...

« Une production industrielle de didacticiels se révèle donc indispensable, qui demandera les mêmes précautions de réalisation qu'un livre scolaire, professeurs/auteurs, informaticiens et éditeurs devant travailler conjointement pour que le produit soit le plus fiable possible dans sa démarche pédagogique et sa facilité d'utilisation. »

Mais, pour le professeur Simon, l'utilisation judicieuse de l'informatique passe forcément par l'apprentissage de la programmation, base de la compréhension du « fonctionnement informatique ». Dans ce cas, comment dispenser cet apprentissage sinon à l'école ? Il ne faut pas se limiter à l'informatique aux fins d'enseigner mais faire maintenant de l'informatique à l'école, une discipline à part entière.

Le professeur Simon, attend avec grand intérêt les résultats des expériences qui seront entreprises en 1981/1982 dans quelques dizaines de classes, d'une option informatique en seconde. A cette fin, des professeurs, choisis parmi

Le Professeur Jean-Claude SIMON. A ses côtés, A. DELEDICQ



M. GLAYMAN et J. MALAVAL

ceux qui avaient suivi la formation approfondie à l'informatique entre 1970 et 1976, suivent un complément de formation dans les centres universitaires de Paris, Toulouse, Grenoble, Nancy et Rennes, tout en essayant d'envisager des modalités originales pour cette initiation.

Dans l'assistance, les interrogations, marques d'intérêt ou d'inquiétude furent nombreuses, nourrissant ainsi le débat.

Au premier chef, on constate deux demandes majeures :

— Celle de la formation des enseignants, pour laquelle Jean-Claude Simon indique les initiatives prises dans le cadre général de la politique d'informatisation de la société.

— Celle ensuite des didacticiels et du maintien de la responsabilité pédagogique de l'enseignant face à des produits finis.

Jean-Claude Simon fit justement remarquer sur ce dernier point, qu'une telle question ne se posait plus pour les autres moyens didactiques, et que l'enseignant pourrait toujours choisir l'outil le mieux adapté à ses propres méthodes.

Le professeur conclut ce débat en soulignant l'importance primordiale de la formation, confirmant ainsi son adhésion aux préoccupations des enseignants et sa volonté de participer à résoudre cette situation :

« Le grand problème de notre société actuelle, est le problème de la formation et de la formation à tous les niveaux, aussi bien du niveau le plus élevé que du niveau de l'enseignement général. Je crois que l'on a négligé ce problème jusqu'à présent et cela me paraît assez dramatique dans une société qui s'intellectualise, qui se cérébralise, qui se veut de plus en plus intelligente, qui veut utiliser des machines de plus en plus compliquées ».

M. Corbel

DES LIVRES

Aux éditions du P.S.I., « *Visa pour l'informatique* » de J.-M. JEGO. Un livre « sympa » dont le principal défaut se révèle une qualité. Incomplet, ce livre ne prétend pas apprendre à programmer en 50 pages. Mieux, il suscite l'envie d'apprendre en permettant au lecteur de faire des programmes simples dès le départ (Imaginez-vous apprendre le piano sans commencer par le solfège et découvrir ce faisant, l'envie de l'apprendre...). Cette formule attrayante déclenche l'intérêt du lecteur et lui donne envie de poursuivre en passant aux ouvrages plus durs qu'il n'avait pas eu envie d'ouvrir auparavant...

« *L'apprentissage du Basic* » de C. ROSSI aux Éditions EYROLLES. Un livre utile pour celui qui apprend seul grâce à sa formule scolaire (chaque chapitre est ponctué d'exercices et de questions sur la bonne compréhension du thème étudié). Ce manuel, tout en faisant un tour d'horizon sur les différentes difficultés de la pratique du Basic, reste accessible et d'un bon niveau.

A l'occasion du CITEX 81, Éducation 2000 consacre un deuxième numéro à l'informatique et l'enseignement, appelé cette année « *Informatique au Présent* ». Il s'agit d'un bilan très complet présenté sous divers éclairages : témoignages d'enseignants, de chercheurs ou d'élèves, politiques et stratégies adoptées. En quelque sorte, une brochure en forme de kaléidoscope où chacun pourra revoir son image favorite, mais y trouver aussi matière à imagination.

E & I a également reçu :

« *Microprocesseurs et microordinateurs - matériel et logiciel* » de R.J. TOCCI et L.P. LASKOWSKY

« *Pico-Informatique et gestion d'entreprise - modèles et programmes pour calculatrices* » de G. BAUMGARTNER et J.-M. PETITGANG (Les Éditions d'Organisation).

DES JEUX QUI PARLENT...

Video Voice System est un ensemble de jeux (220 $) connectables à un téléviseur et utilisant la synthèse vocale grâce au circuit TMS-5200 de Texas Instruments.

Un petit système d'enseignement, le K-2-8 Talking Learning Computer, utilise également la synthèse vocale pour enseigner l'arithmétique, la lecture et l'orthographe. Extérieurement, il est analogue au Speak and Spell de Texas ; il utilise le circuit de synthèse vocale SC-01 de Votrax et coûte 99 $.

... DES MICROS AUSSI

VOTRAX fabrique, sous le nom de Type-n'-Talk, un périphérique connectable à un micro. Contrairement au module VOTRAX commercialisé précédemment par Radio-Shack, l'utilisateur n'a plus besoin d'écrire de programme phonétique. Il suffit de taper la phrase au clavier et le dispositif la prononcera chaque fois qu'on lui en fera la demande. Destiné aux amateurs, son prix est de 345 $.

VIDÉO

La compétition est serrée entre Matsushita, Sony et Hitachi. Tous trois se proposent de réaliser des caméras entièrement électroniques où l'image sera enregistrée directement sur ruban magnétique, grâce à un petit magnétoscope à cassette incorporé à la caméra. La cassette a pour dimensions 100 × 65 × 14 mm, utilise un ruban de 10 microns d'épaisseur et permet une durée d'enregistrement variant de 20 minutes. (Sony) à 2 heures (Matsushita). La caméra développée par Matsushita aurait pour dimensions 229 × 118 × 67 mm avec un poids total de 2 kg. La caméra servira de lecteur de cassettes, mais pour la « protection » du film sur un écran de télévision, il faudra intercaler un adaptateur entre la caméra et le téléviseur. Ces caméras ne seront pas disponibles avant quelques années et le prix prévu est de l'ordre de 1 000 $.

La Compagnie Genisco Computer développe un système de protection d'images à 3 dimensions dans l'espace. 5 images d'un objet à 5 profondeurs différentes sont affichées successivement sur un écran de télévision. Ces images sont reprises par un miroir en plastique qui se déforme de manière périodique en changeant de distance focale. On voit ainsi apparaître un objet dans l'espace sur une profondeur de 25 cm. Les premiers équipements seront livrés vers la fin de 1981 au prix de 100 000 $.

3 mois après avoir introduit sur le marché le lecteur de vidéo-disques de type Sélectavision, RCA va augmenter sa capacité de production de vidéo-disques à cause de la demande (3 millions de disques en 1981, 10 millions en 1982 et peut-être 30 millions en 1983). Par contre, seulement 28 000 lecteurs ont été vendus jusqu'à présent alors que les prévisions étaient de 200 000 unités pour 1981.

Comtal Corp. a exposé à la National Computer Conference (Chicago - mai 1981) un système de traitement d'image dont les actions sont commandées par une entrée vocale.

EN DIRECT DU JAPON

Nippon Telegraph and Telephone amorce le DIPS modèle 45. Il a un cycle de base de 15 nanosecondes et un cache à deux niveaux entre l'unité de traitement et la mémoire centrale de 128 Mega octets.

Casio commercialise le VL1 (30 cm de long, 500 grammes) permettant, avec un clavier à 29 touches, de produire un million de sons différents sur 4 octaves et demi. L'appareil peut stocker en mémoire des mélodies comportant jusqu'à 100 notes. Mis en vente en février de cette année, son prix est de 62 $ et sa production actuelle est de 70 000 unités par mois !

Fujîtsu annonce le microordinateur Bubcom bâti autour d'un microprocesseur Z80A. En plus d'une mémoire de 64 k octets et d'un disque souple il possède deux réceptacles pour des cartouches de mémoire à bulles magnétiques de 32 k octets (qui passeront à 128 k octets dans un an. Le Bubcom coûte, au Japon 1 200 $ et chaque cartouche coûte 130 $).

Le nouvel ordinateur japonais HITAC M-280H est plus rapide que l'ordinateur le plus puissant d'IBM, le 3081. Capable d'exécuter 30 millions d'opérations en virgule flottante par seconde, sa version minimale se loue 200 000 $ par mois.

ET D'AILLEURS...

L'institut Herz à Berlin-Ouest a développé un système de transmis-

sion à fibres optiques capable de transmettre 2,24 milliards de bits par seconde. Ceci représente la transmission simultanée de 30 700 communications téléphoniques ou une soixantaine d'émissions de télévision simultanément.

Le micro-ordinateur anglais SINCLAIR ZX80 coûte 100 livres sterling (sans console). Sorti il y a 9 mois, il a été vendu à plus de 50 000 exemplaires. Sinclair va mettre en production un écran de télévision plat 15 cm × 10 cm (épaisseur 2,5 cm) qui devrait sortir fin 1982, au prix de 60 livres sterling.

Un constructeur canadien qui fabrique le mini-ordinateur PERQ lance une nouvelle mode. Dans ses brochures publicitaires il énumère les questions habituelles des clients : quel microprocesseur utilisez-vous ? Quelle taille mémoire ? Quel système d'exploitation ? et y répond par une seule question « Qu'est-ce que ça peut vous faire puisqu'on vous dit que ça marche ».

DES MICROS, DES MICROS, TOUJOURS DES MICROS...

A l'International Solid State Circuits Conference, on a annoncé :

— Le microprocesseur 32 bits MAC-32 développé aux Bell-Laboratories et destiné à l'usage exclusif des équipements fabriqués par Western Electric.

— Un microprocesseur 32 bits de Hewlett-Packard sur une seule puce avec 450 000 circuits actifs. On s'attend à une vitesse de fonctionnement de l'ordre de 10 à 20 millions d'instructions par seconde. Le microprocesseur est entièrement microprogrammé et il se teste automatiquement chaque fois qu'il est mis sous tension.

— Un microprocesseur 16/32 bits de National Semi-Conductor, le NS 16 000 sur une puce de 64 mm² comportant 60 000 transistors dans un boîtier standard à 48 broches.

— Une mémoire 64 k bits de Nippon Electric sur une puce de 28 mm², avec un temps d'accès de 150 nanosecondes.

— Un micro-ordinateur 16 bits chez Texas Instruments, le TMS 99000 entièrement compatible avec la famille actuelle des microprocesseurs 8 bits, TMS 99000.

— Zilog annonce le microprocesseur 16 bits Z800 entièrement compatible avec le Z80. Ce microprocesseur 3 à 5 fois plus rapide que le Z80 devrait être disponible fin 1982 au prix de 10 $ et accepterait tout le logiciel du Z80.

Contrairement aux bruits qui avaient courus et selon lesquels IBM commercialiserait un micro fabriqué par Matsushita, il semble qu'IBM sortirait son propre micro baptisé CHESS construit autour d'un 8088 d'INTEL avec le même jeu d'instructions que celui du microprocesseur 16 bits 8086 d'INTEL. Pour 4 000 $ il y aurait 256 k octets, deux disques souples (double face, double densité) et une console noir et blanc (écran couleur en option).

Trois « grands » entrent en force dans le marché du micro-ordinateur :

— XERO X offre le 820 basé sur un Z80 avec 64 k octets de mémoire, 2 disques souples (184 k octets au total), clavier, écran et imprimante 40 caractères/seconde.

— Digital Equipment offre le modèle 278 basé sur un microprocesseur 12 bits dérivé du célèbre PDP-8 avec 32 k mots de 12 bits, des disques souples (1 Moctets au total) avec clavier, écran et une imprimante à 30 caractère/s.

— Data-General annonce l'Entreprise 1000 basé sur un microprocesseur 16 bits micro-Nova avec 64 octets, 716 k octets sur disques souples, écran, clavier et imprimante à 150 caractères/s.

Les 3 machines ont un prix voisin de 7 000 $; le logiciel est vendu séparément entre 800 et 1 600 $ par module (paye, comptabilité, facturation, etc).

La fabrication de classe professionnelle, la qualité du logiciel, le volume de logiciel disponible et les garanties de maintenance dues au réseau commercial international de ces compagnies risquent de porter un coup sérieux aux petits fabricants de micro-ordinateurs qui essayent depuis un an ou deux d'attaquer le marché des PME-PMI.

MAIS AUSSI...

Hewlett-Packard annonce la mise au point d'un dispositif à balayage électronique permettant de dessiner des circuits intégrés avec une largeur de trait inférieure à 0,5 microns. Ceci devrait permettre d'intégrer plus d'un million de transistors sur une seule « puce ».

Hewlett-Packard annonce son nouveau terminal HP125 qui comporte deux microprocesseurs Z80. L'un permet au HP125 de fonctionner en micro-ordinateur autonome sous le système d'exploitation CP/M tandis que l'autre sert de contrôleur pour un raccordement à un ordinateur HP-3000.

Les prochains postes téléphoniques permettront de stocker en mémoire jusqu'à 10 numéros que l'on pourra appeler avec seulement 2 boutons-poussoirs ; ils afficheront la date et l'heure, le numéro appelé, la durée de la communication en minutes et secondes ainsi que le nombre d'unités écoulées.

La société américaine Interlude édite un logiciel pour micro-ordinateur accompagné d'un manuel... pour adultes. Le programme pose des questions... indiscrètes à l'utilisateur sur lui-même et sa ou ses partenaires à la suite de quoi il présente sur l'écran la concrétisation de ses fantasmes ou le renvoie au manuel pour les plus amples informations. On ne précise pas si le programme est à sortie vocale...

La quasi totalité de l'industrie des circuits intégrés utilise, aujourd'hui, le silicium comme matière première. Des experts prédisent un remplacement progressif du silicium par l'arséniure de gallium (AsGa). L'AsGa permet de réaliser des circuits intégrés fonctionnant plus vite, tout en dissipant moins de chaleur, ce qui permet une plus grande densité de circuits par mm².

Pour éviter les extensions « sauvages » de PASCAL destinées à remédier à certaines lacunes de ce langage, N. Wirth (le père de PASCAL) vient de présenter une version étendue de PASCAL appelée MODULA-2.

programmation

par A. Deledicq

LEÇON 2
L'ACTION ET LA PASSION

Cette leçon est plus spécifiquement consacrée à la découverte « concrète » d'un micro-ordinateur aujourd'hui commercialisé. Vous y apprendrez les deux ou trois petites choses qu'il vaut mieux comprendre et ne pas oublier.

2.1 LE POUVOIR-FAIRE D'UN MICRO-ORDINATEUR

Le pouvoir-faire d'un micro-ordinateur est aujourd'hui le suivant :

• D'une part, sans laisser de traces visibles (car il s'agit de transformations électroniques très rapides et physiquement de très petite importance), le micro-ordinateur peut :

— Recevoir des informations en provenance :
- D'un clavier de machine à écrire (lettres ou chiffres)
- D'un lecteur de disquettes ou de cassettes magnétiques
- D'une ligne téléphonique
- D'un curseur ou autre « stylet » se déplaçant.

— Stocker ces informations dans ses mémoires :
(Un octet présentant 2^8 états différents, soit 256, permet de stocker au moins 1 caractère ; une mémoire de 4 K octets permet donc de stocker 4 000 caractères, soit environ 100 lignes de 40 caractères. (2 bonnes pages de livres, chiffres valables à un facteur 2 ou 3 près).

— Transformer les informations contenues dans ses mémoires selon des procédures spécifiées :
- Transport d'une mémoire à une autre
- Opérations diverses sur les informations numériques codées (des additions au calcul de cosinus, comparaisons...)
- Opérations diverses sur les informations alphabétiques codées (des mises bout à bout aux insertions ou extractions de texte).

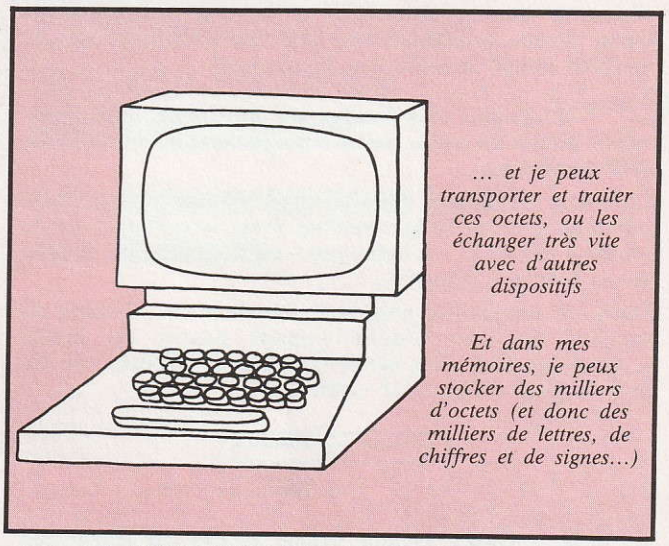
• D'autre part, en réponse à des ordres spécifiques, le micro-ordinateur peut :

— Écrire des signes, lettres ou chiffres sur un écran ou une imprimante.

— Émettre des sons de fréquence et de durée spécifiées.

— Dessiner sur un écran en marquant des points de couleurs à des endroits spécifiés d'un quadrillage ou sur une « page » en déplaçant une « plume » sur un trajet continu.

— Transmettre des informations contenues dans ses mémoires sur des disques, cassettes, lignes téléphoniques ou autres « sorties ».



... et je peux transporter et traiter ces octets, ou les échanger très vite avec d'autres dispositifs

Et dans mes mémoires, je peux stocker des milliers d'octets (et donc des milliers de lettres, de chiffres et de signes...)

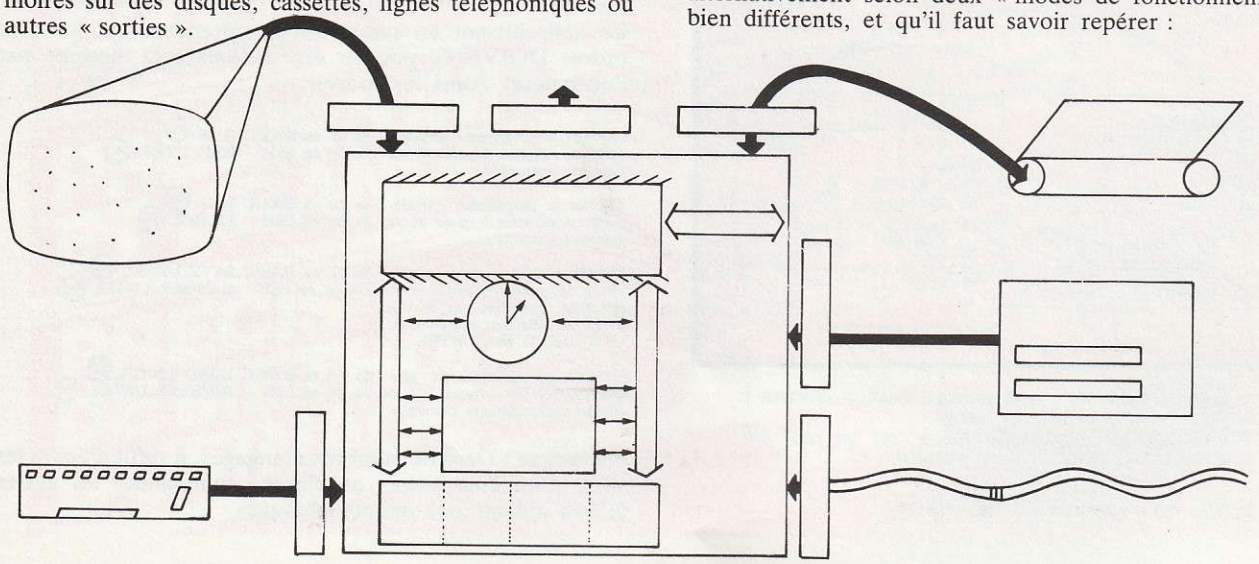
2.2 LA CONDUITE PAR PROGRAMME

La caractéristique essentielle d'un ordinateur est d'être programmable ; sa conduite est donc tout à fait différente de celle d'une voiture ou d'une entreprise. En effet, un programme est COMPLÈTEMENT « écrit » AVANT son exécution par la machine exécutante. Toutes les éventualités doivent donc être prévues à l'avance et une fois donné l'ordre d'exécution d'un programme, la machine se met à faire ce qu'elle doit faire, sauf intervention intempestive extérieure.

Il y a donc au moins deux phases successives dans l'utilisation d'une machine programmable :

— La phase passive : l'enregistrement d'un programme
— La phase active : l'exécution d'un programme

Plus précisément, une machine programmable travaille alternativement selon deux « modes de fonctionnement » bien différents, et qu'il faut savoir repérer :



PASSION : Le mode « enregistrement d'un programme »

Tout ce que l'on frappe au clavier est enregistré (recopié) sur une sorte de « cahier de texte ». Les instructions frappées sont rangées (l'une après l'autre, avec leurs numéros s'il y en a, à raison d'un numéro par ligne) dans une mémoire spéciale, la mémoire-programme.

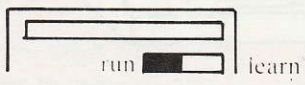
ACTION : Le mode « exécution »

Tout ce que l'on frappe au clavier est considéré comme un ordre à exécuter immédiatement. L'exécution de l'ordre déclenche une certaine action et un certain changement d'état interne de la machine. Une fois l'ordre exécuté, la machine attend la suite des événements.

1 Lorsque vous frappez sur un clavier, interrogez-vous : Suis-je en mode passif (enregistrement) ou en mode actif (exécution).

Dans une machine à laver le linge, « l'enregistrement du programme se fait en appuyant sur toute autre touche que la touche « départ ». « L'exécution » est déclenché par l'appui sur la touche « départ ».

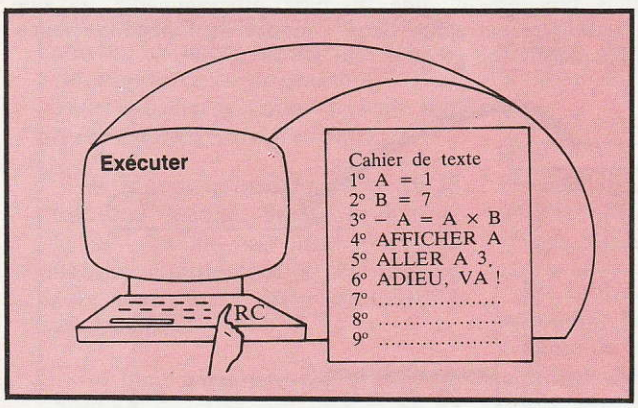
Dans une calculatrice programmable de poche, il existe un curseur ou une touche faisant passer du mode « enregistrement » au mode « exécution (repérés par les mots anglais « learn » et « run ») :



En mode « learn », chaque touche frappée est enregistrée sous forme codée dans une mémoire-programme.
En mode « run », chaque touche frappée déclenche une action de la machine : écriture d'un chiffre, transfert en mémoire, exécution d'une opération...

Dans un micro-ordinateur à clavier, le mode de fonctionnement est souvent sous-entendu par la convention suivante :

- Si la ligne frappée commence par un chiffre, alors cette ligne sera enregistrée comme une instruction de programme ; le nombre en début de ligne sera le numéro de cette instruction dans la suite de toutes les instructions.
- Si la ligne frappée ne commence pas par un chiffre, alors cette ligne sera considérée comme un ordre à exécuter immédiatement.



L'ordre « **EXÉCUTER** RC » va déclencher l'action suivante :
- Aller voir le cahier de textes
- Exécuter les instructions qui y ont un jour été écrites, dans l'ordre de leur numéro.

2 « Terminé, à vous »

Lorsqu'on donne un ordre, il est presque toujours nécessaire d'indiquer la fin de son énoncé.

Par exemple, si vous écrivez « effectuer l'instruction numéro 13 », il est normal que l'instruction numéro 13 ne soit pas exécutée ! En effet, il se peut très bien que vous vouliez écrire « effectuer l'instruction numéro 132 » et qu'il faille donc attendre le chiffre 2 final (mais vous êtes le seul à savoir qu'un chiffre est final ou pas !).

En conséquence, tout énoncé d'un ordre doit être suivi de l'indication « fin d'ordre ».

Sur les micro-ordinateurs à clavier, un ordre est écrit sur une ligne, l'indication « fin d'ordre » est donc donnée par la frappe de la touche « fin de ligne », souvent marquée ←/ ou RC (retour-charriot) : l'ordre est alors interprété et exécuté par la machine.

Remarque : lorsqu'une ligne de programme numérotée est entrée en mémoire, elle se place conformément à son numéro ; on peut donc frapper la ligne 15 après la ligne 20, elles seront ensuite exécutées dans l'ordre de leurs numéros. Le principe d'enregistrement est celui du magnétophone. Toute ligne 20 nouvellement entrée, efface la ligne 20 précédemment introduite. En particulier, la ligne « vide » : 20 RC efface le contenu de la ligne 20.

3 Ce que vous voyez sur l'écran n'est pas forcément dans la machine

Précisément, ce n'est que lorsque la touche « fin de ligne » est frappée,
— « Que l'instruction écrite sera enregistrée en mode « enregistrement ».
— Que l'ordre énoncé sera exécuté en mode « exécution ».

Remarque : Lorsqu'on frappe à la machine, on peut vouloir modifier ce que l'on a frappé ; il existe des touches facilitant ces modifications : déplacement du curseur d'écriture, effacement...

Repérer bien ces **touches d'aide à l'écriture** sur le clavier ; elles ne font pas à proprement parler, partie du langage de la machine, car elles n'agissent que sur les choses écrites sur l'écran et non sur les informations contenues dans les mémoires.

2.3 LES ORDRES DE GESTION DES PROGRAMMES

En réfléchissant un peu, vous comprendrez que certains ordres DOIVENT pouvoir être donnés à la machine par l'utilisateur. Ainsi les ordres :

Exécuter le programme actuellement en mémoire à partir de sa première instruction.	Se dit en BASIC RUN (RC) Se dit en LSE EXÉCUTER (RC)
Afficher le programme actuellement en mémoire à partir de sa première instruction.	Se dit en BASIC LIST (RC) Se dit en LSE LISTER (RC)
Enregistrer le programme actuellement en mémoire à partir de sa première instruction sur la disquette actuellement branchée en lui donnant le nom UNTEL	Se dit en BASIC SAVE UNTEL (RC) Qe dit en LSE RANGER UNTEL (RC)
Charger en mémoire le programme UNTEL stocké sur la disquette actuellement branchée.	Se dit en BASIC LOAD UNTEL (RC) Se dit en LSE APPELER UNTEL (RC)

Remarque : Dans de nombreux langages, il suffit d'écrire les deux premières lettres de chaque commande, les autres lettres n'étant que mémotechniques.

EDUCATION & INFORMATIQUE

Pour recevoir
EDUCATION ET INFORMATIQUE
à domicile pendant un an
pour 95 F au lieu de 110 F,
complétez et postez
la carte ci-contre
sans affranchir.

Vous paierez plus tard,
sur facture.

NE PAS
AFFRANCHIR

CORRESPONDANCE
RÉPONSE

Validité permanente

A utiliser seulement
en France Métropolitaine
et dans les départements
d'Outre-Mer
pour les envois
ne dépassant pas 20 g.

Librairie Fernand Nathan
Abonnements E et I
Autorisation N° 2 Paris-Sud
75681 PARIS CEDEX 14

J.C. ABRIL, G. C...

MECANIQUE thèmes d'étude

C. REYNAUD, J.L. THOMASSON

...nent à des co...
pour faire comprendre les ...
ses qui mettent en ordre les connais...
ment acquises.

TECHNOLOGIE ET DESSIN DE CONSTRUCTION

C. MERLAUD, J. OZBOLT

Les connaissances de base en technologie de construction en liaison avec l'initiation aux outils graphiques de communication.

PRECIS : UNITES ET GRANDEURS

En collaboration avec l'AFNOR, un instrument de travail accessible à tous.
Sous forme de dictionnaire, les généralités et principes d'écriture, les termes à employer, le système international d'unité, le classement des grandeurs, le classement alphabétique des unités.

PRECIS DE METHODE D'USINAGE

Toujours en collaboration avec l'AFNOR et après les 2 tomes du PRÉCIS DE CONSTRUCTION MÉCANIQUE, un nouveau PRÉCIS qui apportera une aide efficace tant au débutant qu'au préparateur confirmé, lors de l'établissement des processus d'usinage.

FERNAND NATHAN
Correspondance : 9, rue Méchain
75676 PARIS CEDEX 14
Magasin et vente aux enseignants :
18, rue Monsieur-Le-Prince PARIS 6e

81 C 6195

SICOB BOUTIQUE INFORMATIQUE

EXPOSITION D'INFORMATIQUE INDIVIDUELLE
A PROXIMITÉ IMMÉDIATE DU CNIT - ENTRÉE LIBRE.

SICOB 81

CNIT PARIS LA DEFENSE DE 9H30 A 18H. FERME DIMANCHE 27.

DU 23 SEPT. AU 2 OCT. 1981

PUBLICIS U 8283