PLAIDOYER POUR UNE INITIATION DOUCE MAIS FONCTIONNELLE A LA PROGRAMMATION EN BASIC COMME APPORT A LA FORMATION DES ENSEIGNANTS DE FRANÇAIS.

Jean-Louis MALANDAIN

L'ORDINATEUR, UN OUTIL POUR LE FRANCAIS...

Parmi les approches qui peuvent convaincre un professeur de français de l'utilité de l'informatique, certaines sont fonctionnelles, c'est le cas de l'initiation au traitement de texte. Le professeur de français étant, par définition, concerné par l'écriture,il peut être séduit par un outil comparable à une "super" machine à écrire ; mais il ne sera "converti" que dans la mesure où l'usage du clavier devient banal et quotidien, au point de remplacer la plume : cela suppose, tout simplement, que le professeur et les élèves aient une machine performante et un traitement de qualité professionnelle à leur disposition... La démonstration est, ensuite, très facile à faire !

Une autre approche consiste à donner au professeur les connaissances et la pratique minimales pour utiliser des programmes conçus par des spécialistes ou d'autres collègues ; il lui restera alors à déterminer le moment opportun pour intégrer l'apport de ces programmes aux contenus de son enseignement. Si les didacticiels disponibles tournent bien , répondent à des attentes ou suscitent de nouvelles orientations de travail, le professeur deviendra vite un fervent consommateur de cette nouvelle technologie éducative... et on ne voit vraiment pas pourquoi il s'en priverait. Sauf, peut-être, à considérer qu'un beau matin il aura épuisé le stock disponible sans pouvoir faire gérer par la machine des séquences didactiques dont il a l'idée et qui lui rendraient de grands services dans la classe.

Reste une approche a priori plus ardue qui consiste à initier le professeur de français à la programmation en BASIC (le seul langage réellement à la disposition de tous). C'est cette voie qui a été choisie...

non pas exclusive des deux autres mais, à coup sûr, préalable car leur extrême facilité compromet trop souvent le capital de confiance, de bonne volonté et d'émerveillement nécessaire pour compenser les efforts demandés dans la phase d'initiation. L'usage du téléphérique n'est sans doute pas le meilleur moyen d'initier aux joies de l'alpinisme...

Cela ne signifie pas qu'il faut souffrir inutilement et l'approche ardue ne peut réussir que si elle apparaît d'emblée comme fonctionnelle, si elle aide le professeur à régler d'abord des problèmes dont il verra clairement l'utilité dans la classe de français. D'autant qu'ainsi, on situe les tâches dans le domaine didactique et que peuvent s'engager dès le début une réflexion et des échanges qui garantissent une bonne intégration de l'informatique comme outil de formation.

Il convient donc d'ignorer les particularités dont on n'a pas besoin immédiatement, même si elles font partie du bagage initial communément admis. Il en va d'un langage informatique comme de l'enseignement des langues : pour des adultes,en particulier, les approches fonctionnelles sur des objectifs spécifiques sont préférables à la description grammaticale et à l'inventaire lexical. Il n'y a pas, non plus, à tenir compte d'une gradation du simple au complexe, au détriment de ce qui est fonctionnellement prioritaire. L'imparfait est plus facile que le passé composé, dit-on parfois... on sait bien où conduit ce raisonnement. D'ailleurs, il faut bien se dire qu'au début tout est difficile car un peu mystérieux et surtout parce qu'il faut mobiliser en même temps un minimum d'une dizaine d'instructions pour faire le plus petit programme, et qu'on oublie forcément plusieurs fois, et qu'il faut un minimum de temps pour que notre tête organise de nouveaux espaces. Dans ces conditions, autant commencer par ce qui est vraiment utile dans le domaine visé : l'enseignement du français, sans imposer une démarche abstraite très démobilisatrice pour des enseignants de français qui se souviennent d'expériences malheureuses en mathématiques (car la confusion persiste entre cette discipline et l'informatique) ni laisser les stagiaires démunis par la double interrogation : COMMENT faire QUOI avec un ordinateur ?

Deux raisons supplémentaires plaident pour une telle approche : d'abord et paradoxalement, la brièveté de la formation (une dizaine de séances, disons un maximum de vingt heures) et la rareté de telles occasions dans la carrière d'un enseignant. Est-ce exagéré de dire que c'est l'occasion d'une vie ? Raison majeure pour ne pas la manquer ; le plus important est sans doute de convaincre (même si des zones restent

encore dans l'ombre) qu'un professeur de français peut avoir la maîtrise d'un outil informatique : "Ce n'est plus un mystère et, si je voulais, je pourrais en tirer des applications à ma portée !". Voilà le genre de réflexion qui semble une évaluation satisfaisante à la fin de l'initiation. On pourrait d'ailleurs nuancer en ajoutant : "Le jour où j'aurai cet outil à ma disposition, je pourrai...". Dans ce but, l'accent est mis sur les deux phases MONTRER CE QU'ON PEUT OBTENIR et MONTRER COMMENT ON PEUT L'OBTENIR, au détriment des manipulations individuelles si le nombre de machines et le temps disponible ne permettent pas le libre-service.

La seconde raison tient aux particularités des configurations en usage : un matériel peu coûteux tel qu'on peut le trouver dans les systèmes scolaires des pays développés, c'est-à-dire autour de 5000 francs pour l'ensemble unité centrale, mémoire de masse et moniteur. C'est le type de micro-informatique dit "familial" auquel le grand public peut avoir accès et trouver dans les super-marchés. A moyenne échéance, on peut parler d'une option individuelle : tel professeur de langue, qui s'est doté d'une radio-cassette emploiera la même énergie à se procurer un écran et un ordinateur s'il est convaincu de l'utilité de ces nouveaux auxiliaires.

De ce point de vue, les machines Thomson, tant décriées par ailleurs, ont des avantages incontournables tels que le branchement sur une télévision (parfois déjà installée pour la vidéo), la couleur, l'affichage en 40 colonnes et en double taille (donc visible de loin), sans compter, inestimable en langue, la diffusion de séquences sonores associées à des messages écrits.

On remarquera, en passant, qu'il s'agit bien d'installer UN ordinateur et UN moniteur dans la classe et non d'aménager une salle blindée avec une batterie d'ordinateurs où on emmènerait les élèves de temps en temps selon un rite qui tient de la vénération plus que l'apprentissage. Bien sûr, tant mieux si l'institution scolaire a les moyens de multiplier les machines pour l'entraînement individuel, à condition que leur accès soit libre. Mais la machine est d'abord dans la classe, comme outil d'animation.

Lorsqu'auront été mises en évidence et en pratique quelques unes des grandes fonctionnalités de l'ordinateur en classe de français, il faudra amorcer la réflexion sur la conception d'un scénario et le contenu du "dialogue" entre l'utilisateur et l'écran-clavier selon des cheminements programmés à l'avance. C'est tout le domaine de l'E.A.O.

qui mérite une phase spécifique de formation. A défaut, il faut l'évoquer et renvoyer à des ouvrages mais ne pas en faire un préalable car l'expérience montre que ce discours n'est pas entendu si les stagiaires sont venus pour "être initiés à l'informatique" ; légitimement, ce qui compte pour eux est le contact avec la machine car c'est elle qui est sacralisée par les médias -et non la pédagogie comme on aura pu le remarquer !

Ces considérations préalables sont les enseignements tirés de plusieurs stages ou journées d'information dans des conditions de fonctionnement très diverses ; elles ont donc fait l'objet d'un rodage avec, bien sûr, quelques déboires ou déceptions. les résultats obtenus permettent de proposer un contenu et des procédures satisfaisantes et reproductibles dans le cadre fixé au départ : l'initiation de professeurs de français (ou de langue).

Une dernière précaution concernant la disposition et la mise en place des matériels : le formateur n'explique rien qui ne soit immédiatement réalisé sous les yeux des stagiaires. L'ordinateur est aussi un outil pour le formateur, intégré à la formation et installé pour être vu de tous. C'est, en même temps, l'illustration de l'emploi préconisé dans la salle de classe.

PREMIERE FONCTION : LE STOCKAGE ET L'AFFICHAGE DES MESSAGES.

Le stockage des énoncés et leur affichage au moment où on le souhaite sont deux fonctionnalités immédiatement perçues comme intéressantes par le professeur de français. Comme les mécanismes de mise en oeuvre sont relativement simples, leur présentation peut constituer une première séance "choc". Il ne faut pas penser pour autant que les techniques d'affichage sont rudimentaires ; elles recoupent en effet l'art de la mise en page (spatiale) et de la mise en scène (temporelle) d'où naît la dynamique d'un programme captivant.

Fort
Belle
Elle
Dort,

Sort

Pour les besoins de la cause, on imaginera que le professeur a prévu d'afficher à un moment précis de son cours le sonnet monosyllabique de Jules de Rességuier (XIX° siècle), suffisament original pour éveiller l'atten-

Frêle !
Quelle
Mort !

Rose
Close,
La

Brise
L'a
Prise.

tion des stagiaires et illustrer les problèmes de l'affichage.

Dans le cadre de cet article, il n'est pas possible d'entrer dans le détail de toutes les explications données au cours de la séance ; elles sont nombreuses et ponctuellement indispensables,même si la plupart ne restent pas présentes à la mémoire, heureusement !

La première opération consiste à déposer les textes utiles (ou données), ici les vers du sonnet, dans la mémoire de l'ordinateur sous la forme d'une liste tapée "au kilomètre", en séparant chaque élément par une virgule :

```
1000 DATA FORT,BELLE,ELLE,"DORT,",SORT, ....
```

quand un segment à isoler comporte déjà une virgule, il est inscrit entre guillements pour éviter la confusion entre la virgule-séparateur du BASIC et la virgule ponctuation du texte.

On continue ainsi jusqu'à la fin du sonnet. Si la liste à inscrire dépasse 255 caractères (environ 6 lignes d'écran), il est nécessaire de prendre un nouveau numéro de ligne pour ouvrir une autre zone dans la mémoire de l'ordinateur puisque la première est pleine.

Deuxième opération, demander à la machine d'aller chercher ces segments pour les inscrire à l'écran. Ce qu'il faut faire comprendre est la nécessité d'utiliser un "récipient" pour aller puiser les segments (ce qu'on appelle l'affectation d'une variable) : c'est l'utilisateur qui crée ce récipient par le seul fait de le nommer, du nom de son choix avec la marque **$** pour indiquer que le contenu sera textuel et non numérique.

`10 READ VER$` l'instruction `READ` va chercher le premier segment stocké (FORT) et le "transporte" dans `VER$`

`30 PRINT VER$` l'instruction `PRINT` affiche le contenu de `VER$`

`40 GOTO 10` la machine revient à la ligne 10 et exécute à nouveau l'instruction `READ VER$`. Cette fois, le premier segment rencontré est BELLE puisque FORT est déjà pris.

La machine passe alors à la ligne 30 puis 40 et ainsi de suite jusqu'à ce que la série des segments soit épuisée.

Cette petite machinerie imaginaire peut être facilement simulée à la main, mimée ou représentée au tableau noir, matérialisée avec un gobelet et des cartons sur lesquels on a écrit les segments, tout ce qui contribue à donner une représentation tangible (mais inexacte, bien sûr) du fonctionnement complexe de l'ordinateur. Pour finir, l'écriture est exécutée à la machine par un RUN. On constate alors que la liste s'inscrit à gauche de l'écran et que le programme s'arrête sur un message d'erreur (Error 4 in 10) car, à la fin, READ ne trouve plus rien à mettre dans VER$...

La troisième opération consiste naturellement à introduire le contrôle du programme grâce à la condition IF... THEN.

Après le dernier segment de la série, on place un symbole quelconque qui va signaler la fin :

```
1000 ......L'A,PRISE.,&
```

On indiquera à la machine que si VER$ contient le signe &,il faudra s'arrêter au lieu d'aller plus loin.

Reste à trouver le bon endroit dans la série ordonnée des actions pour "informer" le programme : c'est juste après le remplissage de VER$ qu'il faut observer son contenu pour réagir à temps. Cette constatation amène à commenter le déroulement "algorithmique" des opérations : la machine suit l'ordre prescrit par le programmeur !

```
  10 READ VER$        (READ retrouve DATA, où qu'il soit !)
  20 IF VER$="&" THEN END  (vérification à faire avant toute chose) (le
     signe = se lit "contient")
  30 PRINT VER$
  40 GOTO 10
1000 DATA FORT,BELLE,.........,L'A,PRISE.,&
```

Cette fois, le programme tourne normalement. Les stagiaires ont déjà à leur disposition plusieurs outils essentiels mais feront remarquer que l'affichage n'est guère satisfaisant : il faudrait, au moins séparer les strophes. C'est une bonne occasion de montrer la commodité de l'éditeur plein écran pour compléter ou modifier un programme sans réécrire les lignes. On va donc ajouter des symboles dans la série des segments pour signaler les passages à la ligne, par exemple l'astérisque :

```
1000 DATA FORT,BELLE,ELLE,"DORT,",*,SORT,FRELE !, QUELLE,MORT !
     ,*,ROSE,"CLOSE,",LA,*,BRISE ,L'A, PRISE.,&
```

Il faut évidemment repérer si cette nouvelle information est dans VER$ mais ne pas l'afficher, d'où l'ajout de cette ligne

```
21 IF VER$="*" THEN PRINT :GOTO 10
```

Si VER$ contient le signe *, passer à la ligne (sans rien écrire) et aller chercher le segment suivant.

On voit ainsi comment il est possible d'habiller un texte avec les indications nécessaires à son édition, à la façon d'un code destiné à un imprimeur. Il faudra s'en souvenir pour réaliser de vraies mises en page.

A l'exécution, nouveau motif d'insatisfaction car on voit le programme et, à la suite, les strophes qui font remonter les premières lignes. Le remède est simple : il suffit de placer en tête du programme une instruction qui effacera l'écran :

```
5 CLS
```

C'est l'heure de la pause. Chacun peut vérifier que le programme tourne, qu'on peut le revoir grâce à la commande LIST et le réécrire sur une autre machine.

Le programme tourne mais n'est pas pour autant satisfaisant : il faudrait cadrer le texte, mettre un titre et l'auteur. Surtout on ne voit pas grand chose de loin... Tous ces perfectionnements seront l'objet d'une autre séance pour explorer les modalités de l'affichage (localisation dans l'écran, taille des caractères, minuscules accentuées, couleurs) et aboutir à un programme définitif. Il suffira, cette fois, de donner la liste détaillée des instructions et de confier le travail aux stagiaires.

L'important est d'avoir reconnu les principes généraux du stockage des textes "au kilomètre" (ou presque) par segments qui peuvent être plus longs, bien sûr, puisqu'on peut manipuler de cette façon des "paquets" de 255 caractères, isolés par des guillemets si c'est nécessaire ; on peut alors parler de phrases ou de paragraphes, rassemblés en textes, prêts à surgir à l'écran ou à disparaître instantanément... C'est déjà un exercice de lecture rapide !

LE PROGRAMME DEFINITIF ET SON RESULTAT A L'ECRAN

```
5 CLS
8 ATTRB1,1:LOCATE 10,1:PRINT "SONNET"
9 PRINT:ATTRB 1,0
10 READ VER$
20 IF VER$="#" THEN 50
21 IF VER$="*" THEN PRINT:GOTO 10
30 PRINT SPC(6) VER$
40 GOTO 10
50 ATTRB 0,0:LOCATE 15,24:PRINT "Jules d
e Resseguier";:LOCATE 0,0,0
1000 DATA FORT,BELLE,ELLE,"DORT,",*,SORT
,FRELE !,QUELLE,MORT !,*,ROSE,"CLOSE,",L
A,*,BRISE,L'A,PRISE,#
```

SONNET

FORT
BELLE
ELLE
DORT ,

SORT
FRELE !
QUELLE
MORT !

ROSE
CLOSE ,
LA

BRISE
L'A
PRISE

Jules de Rességuier

SECONDE FONCTION : La réponse de l'élève.

La réponse de l'élève, surtout lorsqu'il s'agit d'un énoncé, intéresse évidemment le professeur de français : son travail consiste, pour l'essentiel, à susciter les productions langagières et à vérifier leur "acceptabilité".

Pour recueillir ces énoncés dans un programme, le BASIC utilise une instruction "miracle" `LINE INPUT` qui permet de garder une réponse dans la mémoire de l'ordinateur. Le mécanisme est simple : quand le programme rencontre `LINE INPUT`, il s'arrête et attend une réponse (jusqu'à 255 caractères) qui est stockée dans une variable (un récipient que le programmeur définit en le nommant, par exemple `REP$`). Il suffit ensuite d'observer le contenu de `REP$` pour vérifier s'il est conforme à la réponse attendue.

Mais il y a un inconvénient de taille pour effectuer la vérification : on ne dispose que du signe `=` qui implique une égalité stricte au caractère près. Pour le BASIC les deux énoncés "Il fait beau." et "il fait beau ;" ne sont pas égaux pour trois raisons : I et i au début, espace avant et après le mot fait, point et point-virgule à la fin.

Au plan de la communication ce manque de souplesse est un handicap majeur qui réduit souvent les didacticiels à des Q.C.M. où l'élève choisit un numéro au lieu de rédiger la réponse : en langue, c'est souvent catastrophique.

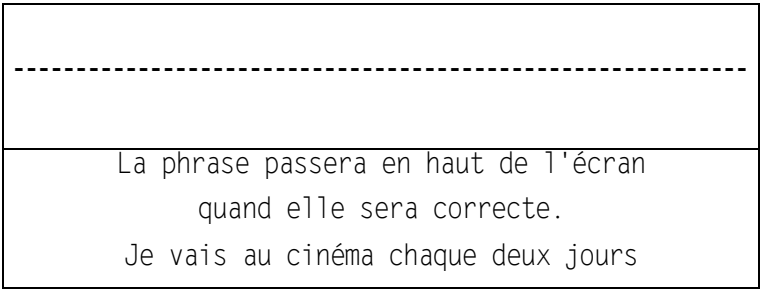
Avant de voir les moyens de dépasser la rigidité de l'égalité stricte, on peut donner aux stagiaires l'occasion de constater un cas exemplaire où la rigueur est payante.

UN CAS D'UTILITE DE L'EGALITE STRICTE.

Tous les professeurs ont présents à l'esprit des erreurs persistantes dont les élèves ont beaucoup de mal à se débarrasser, des fautes incurables en quelque sorte qui tiennent souvent aux habitudes linguistiques et peuvent porter préjudice à l'élève. C'est une situation hélas fréquente où on aimerait se faire aider par une machine dans la fonction de répétiteur.

L'illustration proposée est le montage d'un scénario fondé sur l'égalité stricte pour corriger un élève qui persisterait à dire "Je vais au cinéma chaque deux jours." et l'obliger à produire le bon énoncé.

Le programme va présenter cet énoncé en signalant qu'il n'est pas correct et demander à l'élève de le rectifier pour obtenir un résultat visible. Voici l'écran de départ :



Voyons maintenant le détail du programme qui commence par une affectation ordinaire : FAU$ contient ... BON$ contient...

```
 5 CLS
10 FAU$="Je vais au cinéma chaque deux jours."
20 BON$="je vais au cinéma tous les deux jours."
30 LOCATE 0,10 :PRINT "La phrase passera en haut de l'écran quand
   elle sera correcte."
40 LOCATE 0,15 :PRINT FAU$
50 LOCATE 0,15 :LINE INPUT REP$
60 IF REP$=BON$ THEN CLS :LOCATE 0,5 :PRINT BON$ :END
70 GOTO 50
```

Toute l'astuce consiste à demander la réponse de l'élève sur la ligne où est inscrit l'énoncé fautif ; ainsi, il n'a pas à réécrire la phrase mais seulement à déplacer le curseur et à faire les corrections ; en validant, l'élève fait entrer sa proposition dans REP$. Tant qu'il n'a pas fait toutes les rectifications nécessaires, il revient sur l'énoncé fautif (ligne 50). Dès que la phrase corrigée (REP$) est identique au contenu de BON$, l'écran s'efface et la bonne phrase réapparaît en haut de l'écran.

Comme précédemment, ce noyau de programme doit être "mimé" à la main puis exécuté à la machine avant d'être essayé et réécrit par les stagiaires. Au cours d'une séance suivante, le programme sera amélioré jusqu'à devenir une machine de guerre contre toutes les fautes

fréquentes et graves, les seules qui vaillent la peine de faire un effort de programmation et que le professeur est le seul à bien connaître. Cet exemple du bon usage de l'informatique est d'ailleurs emprunté à une équipe de professeurs qui a fondé sur ce principe un didacticiel très élaboré qui passionne les élèves (cf. Bulletin EPI, N° 37, 38, 39 article d'Agnès COUSIN, "Apprentissage et pédagogie", Collège de Chanteloup-les-Vignes).

Sans aller aussi loin, on peut au moins recenser toutes les fautes rassemblées par les stagiaires : il suffira de les écrire en DATA et d'aller les récupérer par un READ ; autre avantage, la possibilité d'en ajouter autant qu'on veut sans modifier le programme. On peut même partager le travail en attribuant à chaque stagiaire (ou pour chaque niveau, ou par thème etc.) des numéros de ligne de DATA à remplir.

```
  5 CLS
 10 READ FAU$
 12 IF FAU$="&" THEN END
 20 READ BON$
 30 LOCATE 0,10 :PRINT "La phrase passera en haut de l'écran quand
    elle sera correcte."
 40 LOCATE 0,15 :PRINT FAU$
 50 LOCATE 0,15 :LINE INPUT REP$
 60 IF REP$=BON$ THEN CLS :LOCATE 0,5 :PRINT BON$ :GOTO 100
 70 GOTO 50
100 IF INKEY$="" THEN 100
110 GOTO 5
200 DATA Je vais au cinéma chaque deux jours.,Je vais au cinéma
    tous les deux jours.,J'ai beaucoup malade.,Je suis très
    malade.,*
```

Cette fois, quand l'élève a corrigé la phrase, le programme va en ligne 100 et attend qu'une touche soit enfoncée avant d'effacer l'écran pour aller chercher la faute suivante. Si le profes

seur veut compléter la liste inscrite en 200, il suffit d'effacer ,* et d'ouvrir une autre ligne pour écrire la suite :

```
210 DATA Il sait pas chanté.,Il ne sait pas chanter.,*
```

et ainsi autant de fois qu'on le souhaite.

Peut-être pensera-t-on qu'il est cruel de laisser ainsi l'élève aux prises avec des fautes qu'il ne sera pas forcément en mesure de corriger... C'est l'occasion idéale d'introduire la notion d'ergonomie ou de confort des logiciels, car l'objectif d'un programme n'est pas de faire souffrir mais de captiver !

Il convient donc soit d'apporter une aide, soit de donner la bonne réponse après un nombre d'essais fixé par le programmeur, soit les deux. Appelons une variable ESSAI (un récipient qui reçoit des nombres puisque son nom n'a pas la marque $) et utilisons la capacité d'une variable à recevoir son propre contenu grâce à l'écriture ESSAI=ESSAI + 1 qui se lit "ESSAI contient ce qu'il y avait dans ESSAI plus un". Il reste à installer ce qu'on appelle un compteur au bon endroit dans le programme, par exemple en lignes

```
65 ESSAI=ESSAI+1
67 IF ESSAI>3 THEN LOCATE 0,18 :PRINT "La phrase correcte était
     " ; BON$ :ESSAI=0 :GOTO 100
```

A chaque fois que le programme passe par 65, le compteur augmente de un ; s'il y a plus de trois essais (la quatrième fois), le programme donne la bonne réponse puis remet le compteur à zéro (pour les phrases suivantes) avant d'aller en 100. On aurait pu aussi prévoir une aide après le deuxième essai, par exemple un petit commentaire écrit en troisième place dans les DATA et lu en début de programme dans une variable COM$. Il faudrait alors compléter le programme ainsi :

```
 25 READ COM$
 66 IF ESSAI=2 THEN LOCATE 0,17 :PRINT COM$
200 DATA Je vais au cinéma chaque deux jours.,Je vais au cinéma
     tous les deux jours.,La phrase correcte a huit mots !,.....
```

Tout l'art consiste à rédiger une aide judicieuse... et ce n'est pas un problème d'informatique !

INTERMEDE POUR TOURNER LA PAGE ET FAIRE LA PAUSE !

Rappel des fonctionnalités de la puissante instruction PRINT ou ?

à essayer en direct à l'écran :

1) le passage à la ligne.
2) le calcul et l'affichage du résultat d'une opération. `? 2 + 2`
3) l'affichage du résultat d'une fonction. `? LEN("Bonjour")`
4) l'affichage de la valeur de vérité d'un prédicat. `? 3 < 5`
5) l'affichage du contenu d'une variable. `? REP$  ? ESSAI`
6) l'affichage tel quel d'une chaîne de caractères. `?"Tel quel"`
7) la réalisation ou l'affichage du code ASCII. `? CHR$(65)`

Comment sortir de l'égalité stricte pour controler un énoncé ?

Jusqu'ici, les énoncés rencontrés étaient figés, mais le rêve d'un professeur de français est de faire parler ses élèves ou plutôt, tant qu'on n'a que le clavier, de les faire écrire puis de pouvoir apprécier les productions langagières qu'il a tout fait pour susciter.

Cela signifie qu'il faut dépasser la comparaison stricte avec un modèle unique car il existe de multiples façons d'exprimer la même information. Considérons les réponses possibles à la simple question "Quel temps fait-il ?". Pour faire la preuve de leur diversité, il suffit de demander aux stagiaires d'inscrire une seule réponse sur un papier ; on fait ensuite l'inventaire (par un jour sans soleil, par exemple) : "Pas très chaud !", "Le ciel est gris !", "Triste temps !", "Pas vraiment beau !" etc. Bref, c'est l'impasse...

Mais le BASIC dispose de deux ressources qui vont apporter une solution à ce qu'il est convenu d'appeler "l'analyse de réponse". A condition de faire un effort d'attention...

Tout d'abord il existe une fonction qui permet de rechercher la place d'un segment quelconque dans une suite de caractères : on peut, par exemple, savoir la place du segment BEAU dans la suite de caractères IL FAIT VRAIMENT BEAU AUJOURD'HUI. En termes informatiques, la fonction `INSTR` donne le rang de la sous-chaîne BEAU dans la chaîne IL FAIT VRAIMENT BEAU AUJOURD'HUI. L'écriture en est relativement simple ; on peut l'essayer en direct en tapant :

```
PRINT INSTR("IL FAIT VRAIMENT BEAU AUJOURD'HUI","BEAU")
```

On peut aussi passer par des variables, dans un petit programme qui permet d'essayer "in string" en changeant les contenus

```
10 B$="BEAU" :REP$="IL FAIT VRAIMENT BEAU AUJOURD'HUI."
15 B=INSTR(REP$,B$)
20 PRINT B
```

Dans les deux cas, la machine répond 18, c'est-à-dire que BEAU commence à la 18ème place de la chaîne REP$. Quel intérêt ? direz-vous. Aucun dans l'immédiat, 18, 19, 20 peu importe... mais puisque ce nombre est plus grand que zéro, c'est que le segment BEAU est dans la réponse de l'élève. Voilà qui est intéressant ! On peut donc savoir, non pas ce qu'à dit l'élève mais s'il a utilisé tel mot ou segment. C'est en multipliant ces observations qu'on parvient à évaluer un énoncé. S'il y a PAS et CHAUD ou BEAU etc. Désormais, la réponse à la question posée plus haut devient interprétable.

Pour aller plus loin dans l'analyse, il nous faut encore un outil qui évitera les longs détours et les écritures fastidieuses. Il s'agit d'une particularité de l'affectation qu'il est facile d'essayer en direct à l'écran : une sorte de diversion pittoresque qui a déjà été abordée pendant la pause à propos de la puissance de PRINT.

A = 4 "A contient 4" ne pose plus de problèmes... Oublions-le ! Réessayons maintenant PRINT 4 = 2 + 2 qui signifie ECRIRE S'IL EST VRAI QUE 4 = 2 + 2 autrement dit donner la "valeur de vérité" de l'affirmation proposée. S'agissant de nombres, l'ordinateur est expert et répond instantanément, mais sous une forme conventionnelle, -1 quand c'est vrai et 0 quand c'est faux.

PRINT 4 = 2 + 2 va afficher -1 (c'est vrai !)

PRINT 3 = 2 - 1 va afficher 0 (c'est faux !)

PRINT 5 > 6 va afficher 0 (5 n'est pas plus grand que 6)

PRINT A = 3 va afficher 0 car la machine a une bonne mémoire et sait que A ne contient pas 3 mais 4... (voir plus haut !)

Autre sujet d'étonnement, cette écriture :

V = 5 > 6 qui se lit "V contient la valeur de vérité de l'affirmation selon laquelle 5 est plus grand que 6" qui permet de dire, sous forme abrégée :

`IF V THEN PRINT "VRAI !" ELSE PRINT "FAUX !"` qui se lit "si `V` contient -1 (autrement dit, si c'est vrai) alors afficher VRAI ! sinon afficher FAUX !

Détendez-vous, en général c'est là qu'on perd pied !

C'est cette possibilité de contrôler des affirmations qui sera utilisée pour accélérer l'analyse des réponses, après la pause, quand tout le monde aura vérifié ces étonnantes capacités du BASIC.

Ainsi, après la réponse à la question "Quel temps fait-il ?" on installera des machines à repérer la présence de certains mots ou segments significatifs :

```
10 PRINT "Quel temps fait-il ?"
15 LINE INPUT REP$
20 P = INSTR(REP$,"pas") > 0
21 B = INSTR(REP$,"beau") > 0
22 C = INSTR(REP$,"chaud") > 0
23 M = INSTR(REP$,"mauvais") > 0 et cetera...
```

Nous avons maintenant à notre disposition les "boîtes" `P, B, C,` et `M` -appelées ainsi pour rappeler les mots recherchés, mais le programmeur peu sûr de sa mémoire aurait pu les appeler carrément PAS, BEAU, CHAUD et MAUVAIS, tel autre, plus abstrait, aurait pu les appeler `A, B, C, D`... c'est affaire de goût !

Avec ces variables, nous allons pouvoir faire une phrase pour fixer nos exigences ; ce sera une sorte de filtre réalisé grâce à des articulations logiques assez proches de la langue naturelle (en informatique on parle d'opérateurs logiques `AND OR NOT` etc). Si `P` est vrai et que `B` ou `C` sont vrais, autrement dit s'il y a "PAS" dans la réponse etc. Les formulations sont diverses ; à chacun d'adopter celle qui rend le mieux compte des opérations d'analyse. Bref, si la réponse contient `P` et en même temps soit `B` soit `M` , ou alors si la réponse contient `M` mais pas `P`, alors la réponse est acceptable (Rappelez-vous, c'était par un jour sans soleil !)

Heureusement, pour ne pas perdre le fil du raisonnement, le BASIC autorise les parenthèses ou les cascades de filtres :

```
30 IF P AND (B OR M) THEN PRINT "C'EST EXACT !" :END
31 IF M AND NOT P THEN PRINT "C'EST JUSTE !" :END
40 PRINT "REGARDEZ MIEUX LE CIEL !" :GOTO 15
```

Bien sûr, cette analyse est trop rapide pour rendre compte de toutes les réponses. Ce serait d'ailleurs pratiquement impossible ou trop long sans l'expérience du professeur qui est le mieux placé pour prévoir ce que risque de dire un élève en fonction du niveau atteint et ce qu'il faut vérifier à ce moment-là. Une fois assimilée la technique informatique, les vrais problèmes de l'analyse des réponses restent pédagogiques et personne d'autre que l'enseignant ne peut les régler.

Il convient également de signaler que le programmeur n'est jamais à l'abri des risques ; si un élève facétieux répond "Quel beau repas !", le programme va trouver les segments "beau" et "pas" ... et répondre C'EST EXACT ! alors que la question est restée sans réponse. Tel est pris qui croyait prendre ! Il faudra affiner l'analyse (par exemple rechercher " pas" et non "pas") mais surtout compter sur la probabilité et rester très prudent dans les commentaires. Le péremptoire est exclu et l'informatique n'est qu'un outil pour activer la démarche d'apprentissage, pas une machine à sonder les esprits. L'élève facétieux a raison de déjouer le système : il fait la preuve qu'il connaît les éléments significatifs attendus par le professeur. L'objectif est atteint.

Avant de mettre en oeuvre ces machines à filtrer les réponses et pour laisser un peu décanter cet amas d'acquisitions, une séance, au moins, sera consacrée à la recherche des éléments pertinents ou exclus dans des réponses à des questions ouvertes. On peut aussi se poser les mêmes questions pour la description d'un dessin ou les épisodes d'un récit en images, sur le résumé d'une anecdote ou d'un dialogue connus des élèves etc. Quels sont les risques d'erreurs pour les élèves ? Quel est le minimum pour considérer que l'énoncé proposé est valable ? Y a-t-il des mots à éviter absolument ? L'intérêt de cette démarche est double : elle prépare une analyse et elle installe une réflexion et des échanges salutaires parmi les stagiaires sur les contenus de l'enseignement. Souvent même, c'est l'occasion d'expliciter une attitude pédagogique, de justifier ses positions à propos de rigueur ou de laxisme, de clarifier le jugement porté sur les élèves etc.

Mais on approche de la dernière séance et le formateur va devoir s'effacer. S'il a su convaincre, les stagiaires sauront trouver dans des manuels de référence les instructions qui leur manquent encore et dans leur pratique quotidienne les idées de programme à mettre au point.

En guise de révision et pour illustrer une particularité des machines Thomson, voici un petit programme à faire tourner

mentalement. Il met en évidence trois stades bien distincts dans le développement d'un mini-scénario :

1) la mise en place d'une sollicitation qui incite l'élève à produire un énoncé. Ici, c'est une musique diffusée par le magnétophone à cassette grâce à l'instruction MOTOR ON.

2) le choix des éléments à vérifier et l'agencement logique des différents filtres mis en place. L'analyse proposée ne vaut que si vous avez une cassette avec une sonate pour piano de Mozart. Sinon, modifiez l'analyse en fonction de votre musique préférée.

3) les commentaires et les actions s'appliquant aux différents cas de réponses.

Au départ, le lecteur de cassette est mis en place avec la touche > (play) enfoncée, la cassette se mettra en route après RUN et s'arrêtera lorsque la réponse sera satisfaisante.

```
 10 CLS :COLOR 0
  20 MOTOR ON
  30 LOCATE 0,10 :ATTRB 1,1 :PRINT "QU'EST-CE QUE C'EST ?"
  40 LOCATE 0,15 :LINE INPUT REP$
  50 LOCATE 0,18 :PRINT SPC(35) " " REM effacement éventuel 102,103
  51 LOCATE 0,20 :PRINT SPC(35) " " REM effacement éventuel 110
  60 A = INSTR(REP$,"MOZART") > 0
  61 B = INSTR(REP$,"PIANO") > 0
  62 C = INSTR(REP$,"SONATE") > 0
  63 D = INSTR(REP$,"MUSIQUE") > 0
  70 LOCATE 0,18 'localisation des commentaires
  80 IF A AND B AND C THEN 100
  81 IF A AND B THEN 101
  82 IF A THEN 102
  83 IF B OR C OR D THEN 103
  90 GOTO 110 'fond de panier pour les autres cas
 100 PRINT "C'EST PARFAIT !" :MOTOR OFF :END
 101 PRINT "C'EST EXACT !" :MOTOR OFF :END
 102 PRINT "OUI, ESSAYEZ DE PRECISER !" :GOTO 40
 103 PRINT "MAIS ENCORE ?" :GOTO 40
 110 LOCATE 0,20 :PRINT "ECOUTEZ PLUS ATTENTIVEMENT !" :GOTO 40
```

Une remarque encore : tel qu'il est, ce programme comprend les réponses tapées en lettres majuscules mais on aurait pu prévoir les minuscules en complétant, par exemple, les contenus des variables A, B, C et D :

```
60 A = INSTR(REP$,"MOZART")>0 OR INSTR(REP$,"Mozart")>0
61 B = INSTR(REP$,"PIANO")>0 OR INSTR(REP$,"piano")>0
```

et ainsi de suite ; de même, on aurait pu vérifier la présence du seul segment MUSI au lieu du mot MUSIQUE pour accepter une réponse comportant MUSICAL, MUSICIEN ou MUSIQUE. L'analyse de réponse devient vite un jeu passionnant où il faut concilier exigence et économie (1).

Jean-Louis MALANDAIN
Chargé d'Etudes au BELC
Bureau National de l'EPI
29/03/87

Cet article reprend des propositions publiées dans les Actes des 3èmes Journées Pédagogiques organisées par l'Association des professeurs de français de Madrid (oct. 1987).

(1) Pour le détail de l'analyse de réponse, en particulier l'utilisation des DATA lorsqu'on doit prendre en compte des séries de synonymes ou d'équivalents, consulter la brochure :
"UN MICRO-ORDINATEUR POUR DES PROFESSEURS DE FRANÇAIS PASSIONNES DE... PEDAGOGIE DU FRANÇAIS", BELC, 1986. 8, Rue Malebranche 75005 PARIS.