

# L'usage d'APL en HTML 4.0

par Bernard Mailhol

## 1 HTML 4.0

HTML (HyperText Markup Language) permet de coder les pages *Web*, largement utilisées en Internet [1] <sup>(p. 75)</sup>. Les fichiers à ce format peuvent aussi être consultés directement, en-dehors d'un réseau. Cette consultation étant indépendante de la machine utilisée, elle se répand largement, les minutes de congrès sont de plus en plus souvent composées de fichiers html.

L'usage de l'APL en HTML pose évidemment le problème du jeu de caractères. Les affichages étaient obtenus de deux façons :

1 - Par appel d'une image montrant le texte ou la fonction.

Cet usage est légal, mais limite les capacités de paramétrage du fureteur, car les images envoyées sont d'une taille fixe, quelle que soit la taille retenue par l'utilisateur pour les caractères normaux. Il pose aussi des problèmes de génération, de mise à jour.

2 - L'autre solution est d'utiliser l'attribut *face=* de la balise `<FONT ...`. Hélas, cet attribut **ne fait pas partie de la norme HTML** Il n'en n'a jamais fait partie.

Cette solution impose de plus une installation particulière sur son poste : charger et installer une police conjoncturelle.

### **La version 4.0 de HTML vient bouleverser ces méthodes**

Cette version est *recommandée* par le W3 Consortium, depuis le 18 décembre 1997 [2] <sup>(p. 75)</sup>. Elle comprend de nombreuses extensions, par rapport à la version précédente (Version 3.2, Janvier 1997) et permet une mise en page personnalisée, via des "feuilles de style".

L'une des évolutions - majeures à mon goût - est l'extension du jeu de caractères retenus : elle permet ainsi d'afficher - en toute compatibilité avec la norme - un texte en toute langue. On peut ainsi écrire le nom de chaque personne sans se tromper dans SON nom (quitte à compléter par une transcription phonétique).

Par exemple, pourquoi écrire seulement *Andreï Bouzine* qui n'est qu'une transcription phonétique, alors que le *vrai* nom est autre. Il s'agit de *Aylhtq <epby*, mais sa transcription - qu'il ne faut pas oublier de préciser - est librement choisie par le rédacteur.

Pouvez-vous vous permettre, aujourd'hui, d'écrire le nom de vos clients, émaillés de fautes dues à la limitation du jeu de caractères que vous utilisez

Ces écritures spéciales (ici en cyrillique) demandent des travaux d'installation très délicats (choix d'une police, recherche des caractères ...)

L'ensemble de ces problèmes est traité par un *produit miracle* : UNICODE.

L'usage d'Unicode ne signifie pas qu'aujourd'hui, tous les problèmes sont résolus. Ceci signifie seulement que nous disposons d'un outil unique pour traiter ces problèmes.

Des fureteurs affichant l'ensemble de l'Unicode sont annoncés comme opérationnels; d'autres verront bientôt le jour.

## 2 LA PRESENCE D'UNICODE

Le fait de disposer d'un alphabet de plus de 256 caractères, dont chacun d'eux ne peut plus donc être stocké sur un seul octet pose de très nombreux problèmes.

Nous pouvons examiner :

1 - La norme ISO 10646<sup>(p. 65)</sup>

2 - APL2 et Unicode<sup>(p. 65)</sup>

3 - HTML et Unicode<sup>(p. 67)</sup>

### 2.1 Unicode et ISO-10646

Unicode est un alphabet de près de 40000 caractères, contenant l'ensemble des caractères nécessaires à l'écriture de tout texte actuel, sur l'ensemble de la planète. Il est défini depuis 1991, indépendamment de l'ISO, organisme de normalisation.

Unicode est un code dont chaque caractère est décrit par 16 bits.

A côté de ce travail l'ISO a normalisé un alphabet *planétaire*, dont chaque caractère occupe 4 octets. Cette norme **ISO 10646** propose de répartir l'ensemble de ces caractères en *plans* de 65536 caractères, soit 256 *rangées* de 256 caractères, par plan.

ISO-10646 définit ainsi 65536 plans, dans un Universal Character Set UCS-4 (4 octets par caractère).

*Le plan 0 est identique à UNICODE*, et constitue un sous-ensemble de ISO-10646, sur deux octets seulement. Ce sous-ensemble (le seul qui soit défini actuellement) s'appelle UCS-2.

Cette norme n'aborde ni les problèmes de stockage, ni les problèmes de transmission, ni les problèmes de clavier... Elle se contente d'énumérer 40000 caractères [3]<sup>(p. 75)</sup>, et d'en donner quelques propriétés.

La bonne nouvelle - en ce qui nous concerne - est que (depuis sa première version), Unicode contient les caractères APL.

## 2.2 Unicode et APL2.

Depuis plusieurs années APL2 permet de stocker les caractères Unicode [4] (p. 75).

De même que les nombres peuvent prendre diverses représentations, selon leur valeur (de 1 bit par élément booléen à 128 bits par élément complexe), les caractères peuvent aussi avoir diverses représentations.

Deux représentations sont possibles, dépendant du caractère à stocker. Ainsi les mêmes caractères (en tant que dessins à reproduire) peuvent être utilisés dans tous les APL, sur différentes machines, sur différents systèmes. Il se peut seulement que certains stockages soient plus volumineux que d'autres.

- 1 - Si le caractère fait partie d'un alphabet réduit de 256 caractères (appelé le `•AV`), il est stocké en mémoire sous un seul octet, dans l'alphabet de la machine.
- 2 - Si le caractère ne fait pas partie de cet alphabet limité, ce caractère est stocké sur 4 caractères, dans le codage ISO-10646. Cette valeur est indépendante du système support, est indépendante de la machine.

La primitive APL2 permettant d'obtenir la valeur unicode d'un caractère est le `•UCS`. On peut alors demander :

```

      •UCS 'šæ'
232 9076
      •AV î 'šæ'
138 230
    
```

Le caractère *e accent grave* a la valeur 232 dans la première rangée de l'UNICODE, mais se trouve (dans l'une des implémentations d'APL2) à la position 138 `•AV`.

Le caractère *rho* a la valeur 9076 en Unicode (et ne se trouve donc pas dans la première rangée). Il se trouve dans le `•AV`.

L'usage d'Unicode permet de décrire tout texte APL (une fonction, par exemple), en utilisant un alphabet indépendant de tout constructeur (à quelques problèmes d'interprétation près, comme par exemple l'étoile qui peut être assimilée à une étoile, ou à un opérateur mathématique)

APL2 a été le premier langage de programmation à prendre en compte l'alphabet Unicode (mais sans l'afficher). L'usage de cet alphabet se répand aujourd'hui dans les autres langages :

La définition de Java (1996) reconnaît Unicode comme police standard, mais il a fallu attendre la livraison de Java : JDK 1.1.4 de Novembre 1997, pour prendre effectivement en compte les affichages Unicode. La police Times Roman contenue dans cette version occupe 11.4 MB.

Le kit d'internationalisation du C (I18N) permet aussi de manipuler ces caractères.

### **2.3 Usage d'Unicode dans les fichiers html 4.0**

Ainsi, HTML 4.0 reconnaît les caractères APL comme des caractères Unicode, c'est à dire des caractères *ordinaires*. Les fureteurs compatibles avec HTML 4.0 sauront donc afficher et imprimer les caractères APL sans aucune déclaration, installation ou précaution complémentaire.

Un texte ordinaire peut ainsi contenir tout caractère APL.

1 - Les caractères *ASCII* sont directement placés dans le fichier html. Ces caractères sont :

\*Les minuscules a-z

\*Les majuscules A-Z

\*Les chiffres 0-9

\*La ponctuation habituelle

2 - Les caractères accentués faisant partie de l'alphabet ISO8859-1 (latin-1)

\*Ces caractères peuvent être transmis directement ex : "*très*". Ce mode n'est pas conseillé.

\*Il peut être codé par un symbole : *tr&egrave;s*;

Ce mode est recommandé, mais il ne concerne pas tous les caractères latin-1

\*Il peut être transmis par son numéro d'ordre dans l'alphabet latin-1. On peut alors écrire "*tr&#232;s*" pour envoyer ce même mot.

Cette dernière méthode est avant tout utilisable par les systèmes de conception, mais n'est guère lisible.

3 - Les autres caractères UNICODE ne peuvent être transmis que par leur numéro.

Voici quelques exemples

\*Le *rho* (æ) se code `&#9076;` ou `&#x2374;`;

\*Le *quad* (•) se code `&#9647;` ou `&#x25AF;`;

\*La *flèche d'affectation*(½) se code `&#8592;` ou `&#x2190;`;

Toutefois, vous pouvez définir votre propre liste de symboles, et la transmettre (ou la référencer) dans chaque page html 4.0 envoyée.

On peut ainsi définir `&rho;` ou `&quad;`

Voici un exemple de fichier html utilisant des caractères Unicode. Ils reprennent l'exemple donné plus haut

```
<pre>
    &#x25AF;UCS '&grave;&#x2374;'
232 9076
    &#x25AF;AV &#x2373; '&grave;&#x2374;'
138 230
</pre>
```

Cette séquence rend alors.

```
    •UCS 'Šæ'
232 9076
    •AV î 'Šæ'
138 230
```

Évidemment, les générateurs de page HTML devront prendre en compte cet alphabet riche.

### 3 L'ALPHABETUNICODE

Cet alphabet est accepté par l'ISO, mais proposé par l'UTC (Unicode Technical Committee). On en trouvera une description exhaustive sur son site :

*<http://www.unicode.org>*

En particulier, deux rangées concernent plus spécialement APL. On peut en trouver une description complète en :

1 - Symboles mathématiques

*<http://www.unicode.org/Unicode.charts/glyphless/U2200.html>*(présentation simplifiée)

*<http://www.unicode.org/Unicode.charts/normal/U2200.html>*(présentation individuelle de chaque code)

2 - Symboles divers, symboles APL

*<http://www.unicode.org/Unicode.charts/glyphless/U2300.html>*(présentation simplifiée)

*<http://www.unicode.org/Unicode.charts/normal/U2300.html>*(présentation individuelle de chaque code)

Parmi ces caractères

- Certains sont utilisés par APL2<sup>(p. 69)</sup>
- D'autres ne le sont pas<sup>(p. 71)</sup>.

#### 3.1 Symboles UNICODE utilisés en APL2

L'alphabet d'APL2 comprend **en ce qui nous concerne** deux familles de caractères :

1 - Des caractères issus de l'alphabet latin-1 (première rangée de l'Unicode)

2 - Des caractères ne se trouvant que dans l'alphabet Unicode. Voici la liste de ces caractères, triés selon leur valeur Unicode.

8359	ž	U+20A7	PESETA SIGN
8592	½	U+2190	LEFTWARDS ARROW
8593	Æ	U+2191	UPWARDS ARROW
8594	,	U+2192	RIGHTWARDS ARROW
8595	Ç	U+2193	DOWNWARDS ARROW
8710	¶	U+2206	INCREMENT
8711	·	U+2207	NABLA
8714	î	U+220A	SMALL ELEMENT OF
8728	ø	U+2218	RING OPERATOR
8744	ë	U+2228	LOGICAL OR
8745	ï	U+2229	INTERSECTION
8746	¬	U+222A	UNION
8757	Ò	U+2235	BECAUSE
8800	ô	U+2260	NOT EQUAL TO
8801	İ	U+2261	IDENTICAL TO
8804	ó	U+2264	LESS-THAN OR EQUAL TO
8805	ò	U+2265	GREATER-THAN OR EQUAL TO
8834	â	U+2282	SUBSET OF
8835	ã	U+2283	SUPERSET OF
8854	é	U+2296	CIRCLED MINUS
8866	Ö	U+22A2	RIGHT TACK
8867	×	U+22A3	LEFT TACK
8868	~	U+22A4	DOWN TACK
8869	•	U+22A5	UP TACK
8968	©	U+2308	LEFT CEILING
8970	¾	U+230A	LEFT FLOOR
9014	Ÿ	U+2336	APL FUNCTIONAL SYMBOL I-BEAM
9015	Ó	U+2337	APL FUNCTIONAL SYMBOL SQUISH QUAD
9017	'	U+2339	APL FUNCTIONAL SYMBOL QUAD DIVIDE
9019	Õ	U+233B	APL FUNCTIONAL SYMBOL QUAD JOT
9021	è	U+233D	APL FUNCTIONAL SYMBOL CIRCLE STILE
9023	ö	U+233F	APL FUNCTIONAL SYMBOL SLASH BAR
9024	ñ	U+2340	APL FUNCTIONAL SYMBOL BACKSLASH BAR
9026	Ô	U+2342	APL FUNCTIONAL SYMBOL QUAD BACKSLASH
9033	í	U+2349	APL FUNCTIONAL SYMBOL CIRCLE BACKSLASH
9035	û	U+234B	APL FUNCTIONAL SYMBOL DELTA STILE
9038	ˉ	U+234E	APL FUNCTIONAL SYMBOL DOWN TACK JOT
9042	ÿ	U+2352	APL FUNCTIONAL SYMBOL DEL STILE
9045	®	U+2355	APL FUNCTIONAL SYMBOL UP TACK JOT
9049	÷	U+2359	APL FUNCTIONAL SYMBOL DELTA UNDERBAR
9053	ä	U+235D	APL FUNCTIONAL SYMBOL UP SHOE JOT
9054	`	U+235E	APL FUNCTIONAL SYMBOL QUOTE QUAD
9055	µ	U+235F	APL FUNCTIONAL SYMBOL CIRCLE STAR
9067	ú	U+236B	APL FUNCTIONAL SYMBOL DEL TILDE
9073	ç	U+2371	APL FUNCTIONAL SYMBOL DOWN CARET TILDE
9074	å	U+2372	APL FUNCTIONAL SYMBOL UP CARET TILDE

9075	î	U+2373	APL FUNCTIONAL SYMBOL IOTA
9076	æ	U+2374	APL FUNCTIONAL SYMBOL RHO
9077	ù	U+2375	APL FUNCTIONAL SYMBOL OMEGA
9079	Ñ	U+2377	APL FUNCTIONAL SYMBOL EPSILON UNDERBAR
9080	Đ	U+2378	APL FUNCTIONAL SYMBOL IOTA UNDERBAR
9082	à	U+237A	APL FUNCTIONAL SYMBOL ALPHA
9472	Ä	U+2500	BOX DRAWINGS LIGHT HORIZONTAL
9474	³	U+2502	BOX DRAWINGS LIGHT VERTICAL
9484	Ũ	U+250C	BOX DRAWINGS LIGHT DOWN AND RIGHT
9488	¿	U+2510	BOX DRAWINGS LIGHT DOWN AND LEFT
9492	À	U+2514	BOX DRAWINGS LIGHT UP AND RIGHT
9496	Û	U+2518	BOX DRAWINGS LIGHT UP AND LEFT
9500	Ã	U+251C	BOX DRAWINGS LIGHT VERTICAL AND RIGHT
9508	´	U+2524	BOX DRAWINGS LIGHT VERTICAL AND LEFT
9516	Â	U+252C	BOX DRAWINGS LIGHT DOWN AND HORIZONTAL
9524	Á	U+2534	BOX DRAWINGS LIGHT UP AND HORIZONTAL
9532	Å	U+253C	BOX DRAWINGS LIGHT VERTICAL AND HORIZONTAL
9552	Í	U+2550	BOX DRAWINGS DOUBLE HORIZONTAL
9553	º	U+2551	BOX DRAWINGS DOUBLE VERTICAL
9556	É	U+2554	BOX DRAWINGS DOUBLE DOWN AND RIGHT
9559	»	U+2557	BOX DRAWINGS DOUBLE DOWN AND LEFT
9562	È	U+255A	BOX DRAWINGS DOUBLE UP AND RIGHT
9565	¼	U+255D	BOX DRAWINGS DOUBLE UP AND LEFT
9568	Î	U+2560	BOX DRAWINGS DOUBLE VERTICAL AND RIGHT
9571	¹	U+2563	BOX DRAWINGS DOUBLE VERTICAL AND LEFT
9574	Ë	U+2566	BOX DRAWINGS DOUBLE DOWN AND HORIZONTAL
9577	Ê	U+2569	BOX DRAWINGS DOUBLE UP AND HORIZONTAL
9580	Ï	U+256C	BOX DRAWINGS DOUBLE VERTICAL AND HORIZONTAL
9600	ß	U+2580	UPPER HALF BLOCK
9604	Ü	U+2584	LOWER HALF BLOCK
9608	Û	U+2588	FULL BLOCK
9617	°	U+2591	LIGHT SHADE
9618	±	U+2592	MEDIUM SHADE
9619	²	U+2593	DARK SHADE
9647	•	U+25AF	WHITE VERTICAL RECTANGLE (quad)
9674	∅	U+25CA	LOZENGE
9675	ê	U+25CB	WHITE CIRCLE

### 3.2 Symboles appelés APL non présents en APL2

La rangée U+U2300 comprend divers symboles, dont plusieurs portent le nom *APL FUNCTIONAL SYMBOL \*\*\*\**. APL2 en utilise certains. Il en ignore d'autres, dans son alphabet habituel. Certains de ces caractères sont effectivement utilisés dans d'autres APL; d'autres sont en réserve.

U+2338	APL FUNCTIONAL SYMBOL QUAD EQUAL
U+233A	APL FUNCTIONAL SYMBOL QUAD DIAMOND
U+233C	APL FUNCTIONAL SYMBOL QUAD CIRCLE
U+233E	APL FUNCTIONAL SYMBOL CIRCLE JOT
U+2341	APL FUNCTIONAL SYMBOL QUAD SLASH
U+2343	APL FUNCTIONAL SYMBOL QUAD LESS-THAN
U+2344	APL FUNCTIONAL SYMBOL QUAD GREATER-THAN
U+2345	APL FUNCTIONAL SYMBOL LEFTWARDS VANE
U+2346	APL FUNCTIONAL SYMBOL RIGHTWARDS VANE
U+2347	APL FUNCTIONAL SYMBOL QUAD LEFTWARDS ARROW
U+2348	APL FUNCTIONAL SYMBOL QUAD RIGHTWARDS ARROW
U+234A	APL FUNCTIONAL SYMBOL DOWN TACK UNDERBAR
U+234C	APL FUNCTIONAL SYMBOL QUAD DOWN CARET
U+234D	APL FUNCTIONAL SYMBOL QUAD DELTA
U+234F	APL FUNCTIONAL SYMBOL UPWARDS VANE
U+2350	APL FUNCTIONAL SYMBOL QUAD UPWARDS ARROW
U+2351	APL FUNCTIONAL SYMBOL UP TACK OVERBAR
U+2353	APL FUNCTIONAL SYMBOL QUAD UP CARET
U+2354	APL FUNCTIONAL SYMBOL QUAD DEL
U+2356	APL FUNCTIONAL SYMBOL DOWNWARDS VANE
U+2357	APL FUNCTIONAL SYMBOL QUAD DOWNWARDS ARROW
U+2358	APL FUNCTIONAL SYMBOL QUOTE UNDERBAR
U+235A	APL FUNCTIONAL SYMBOL DIAMOND UNDERBAR
U+235B	APL FUNCTIONAL SYMBOL JOT UNDERBAR
U+235C	APL FUNCTIONAL SYMBOL CIRCLE UNDERBAR
U+2360	APL FUNCTIONAL SYMBOL QUAD COLON
U+2361	APL FUNCTIONAL SYMBOL UP TACK DIAERESIS
U+2362	APL FUNCTIONAL SYMBOL DEL DIAERESIS
U+2363	APL FUNCTIONAL SYMBOL STAR DIAERESIS
U+2364	APL FUNCTIONAL SYMBOL JOT DIAERESIS
U+2365	APL FUNCTIONAL SYMBOL CIRCLE DIAERESIS
U+2366	APL FUNCTIONAL SYMBOL DOWN SHOE STILE
U+2367	APL FUNCTIONAL SYMBOL LEFT SHOE STILE
U+2368	APL FUNCTIONAL SYMBOL TILDE DIAERESIS
U+2369	APL FUNCTIONAL SYMBOL GREATER-THAN DIAERESIS
U+236A	APL FUNCTIONAL SYMBOL COMMA BAR
U+236C	APL FUNCTIONAL SYMBOL ZILDE
U+236D	APL FUNCTIONAL SYMBOL STILE TILDE
U+236E	APL FUNCTIONAL SYMBOL SEMICOLON UNDERBAR
U+236F	APL FUNCTIONAL SYMBOL QUAD NOT EQUAL

U+2370	APL FUNCTIONAL SYMBOL QUAD QUESTION
U+2376	APL FUNCTIONAL SYMBOL ALPHA UNDERBAR
U+2379	APL FUNCTIONAL SYMBOL OMEGA UNDERBAR

### 3.3 Extensions attendues à l'Unicode.

Certaines extensions sont attendues pour l'Unicode. Parmi celles-ci, deux caractères nous concernent particulièrement :

- 1 - Le caractère quad (•) est actuellement le caractère *WHITE VERTICAL RECTANGLE* de l'Unicode. Une proposition (acceptée) crée un nouveau caractère appelé *APL FUNCTIONAL SYMBOL QUAD*, dont le code est U+237B (accepté en Mars 96 par UTC).
- 2 - Un autre caractère est aussi accepté. Il s'agit du *EURO SIGN*, dont le code est U+20AC (accepté en mai 1997 par UTC).

Nous en aurons bientôt besoin.

Ces deux propositions sont actuellement (Janvier 1998) acceptées par l'UTC, mais en sont à "l'étape 3" de l'ISO.

Il faut en effet faire la différence entre UTC (Unicode Technical Committee), et l'organisme de normalisation qui doit modifier la norme ISO-10646.

Voici les étapes ISO de la préparation de cette norme :

- 1. Initial proposal
- 2. Provisional acceptance by WG2
- 3. Final acceptance by WG2 - in Bucket <-- état actuel.
- 4. Hold for ballot in WG2
- 5. SC2 Ballot
- 6. JTC 1 Ballot
- 7. ITTF Publication

Ces propositions de modification sont explicitées en [\*http://www.unicode.org/unicode/alloc/pipeline.html\*](http://www.unicode.org/unicode/alloc/pipeline.html).

## 4 REFERENCES

•[1] Le format HTML, présentation générale

Les concepts généraux des I\*NET, B. Mailhol (Les Nouvelles numéro 25) ou en <http://www.mailhol.com/pub/afapl> (présentation du 6 novembre 1997, Paris)

•[2] HTML 4.0

Sa définition est trouvée en <http://www.w3.org/TR/REC-html40-971218>; Cette définition existe aussi bien sous des formes consultables (html) que sous des formes directement imprimables (postScript ou PDF).
































































































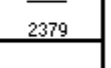







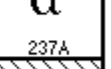
























•[3] Unicode

Pour avoir accès à une information concernant Unicode, interroger <http://www.unicode.org>

•[4] Unicode Support in APL2

J. Brown, GUIDE & SHARE meeting, Vienna, October 1994.

## 5 ANNEXE

	230	231	232	233	234	235	236	237
0	 2300	 2310	 2320	 2330	 2340	 2350	 2360	 2370
1		 2311	 2321	 2331	 2341	 2351	 2361	 2371
2	 2302	 2312	 2322	 2332	 2342	 2352	 2362	 2372
3	 2303	 2313	 2323	 2333	 2343	 2353	 2363	 2373
4	 2304	 2314	 2324	 2334	 2344	 2354	 2364	 2374
5	 2305	 2315	 2325	 2335	 2345	 2355	 2365	 2375
6	 2306	 2316	 2326	 2336	 2346	 2356	 2366	 2376
7	 2307	 2317	 2327	 2337	 2347	 2357	 2367	 2377
8	 2308	 2318	 2328	 2338	 2348	 2358	 2368	 2378
9	 2309	 2319	 2329	 2339	 2349	 2359	 2369	 2379
A	 230A	 231A	 232A	 233A	 234A	 235A	 236A	 237A
B	 230B	 231B	 232B	 233B	 234B	 235B	 236B	
C	 230C	 231C	 232C	 233C	 234C	 235C	 236C	
D	 230D	 231D	 232D	 233D	 234D	 235D	 236D	
E	 230E	 231E	 232E	 233E	 234E	 235E	 236E	
F	 230F	 231F	 232F	 233F	 234F	 235F	 236F	

## Une devinette:

---

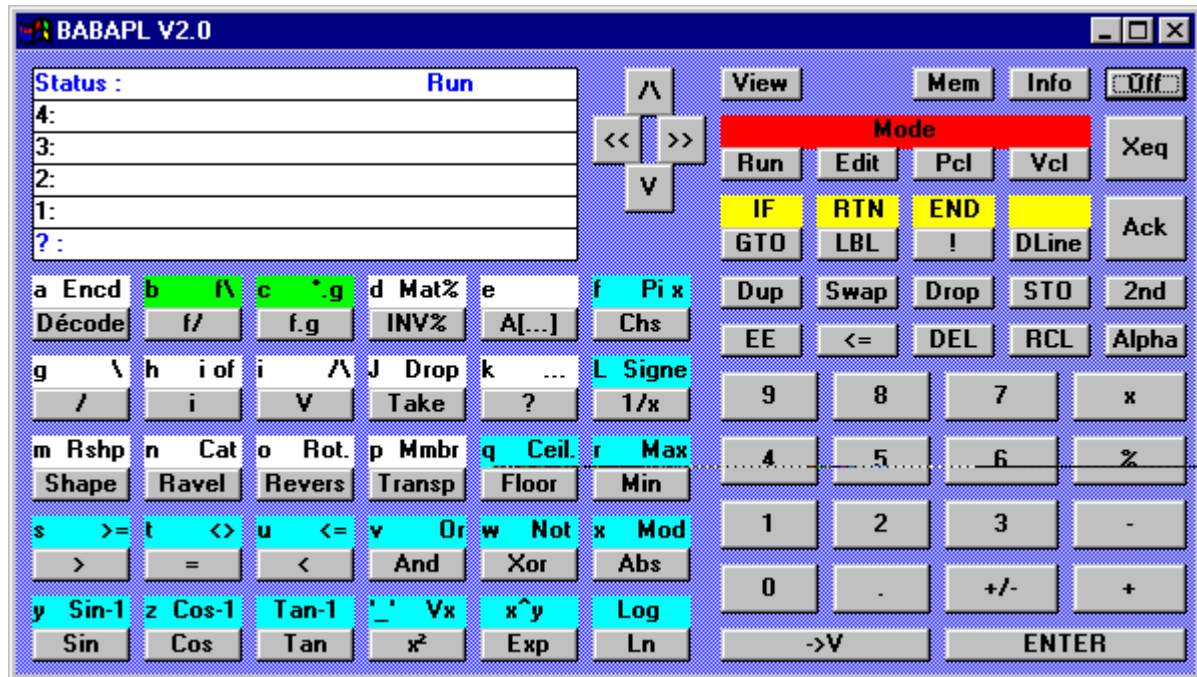
$- \cdot | \sim - * + \pm \text{---} \times \div ! ^{23} \check{Z} \frac{1}{2}, \mu \sim 0$

Quel est le résultat de cette ligne "pornographique"  
sans utiliser d'APL sur vos PC?  
Si vous avez des lignes APL intéressantes, n'hésitez  
pas à les partager avec nous tous!

---

# Une calculette programmable en APL

par Roger BUSI



## Motivations

Le langage APL est comme FORTH, un langage puissant qui mérite d'être mieux connu. Malgré son style sans égal, il reste assez confidentiel, et cela est dommage. Dans la vie courante, je constate souvent que pour résoudre de petits calculs, nos PC et leurs logiciels deviennent lourds à manipuler, et leurs documentations deviennent dissuasives par leur épaisseur. De plus, on ne les a pas toujours à portée de main, ce qui fait que la résolution du problème prend alors du temps !

Même si leur miniaturisation a fait en quelques années des progrès significatifs, ils n'ont toujours pas l'attrait et la facilité d'emploi des calculettes de poche. De ce constat, est née l'idée qu'une calculatrice APL pourrait permettre aux calculettes de continuer à exister, en offrant de la puissance de calcul sous un faible encombrement. En la rendant programmable, la calculette APL deviendrait alors un formidable outil d'investigation pour le technicien, l'étudiant, et le grand public.

## Constat sur l'évolution des calculettes

Au fur et à mesure des années, les calculatrices programmables ont évolué. Nous sommes ainsi passé d'une programmation en langage machine, à une programmation évoluée, et même pour certaines calculettes, au calcul symbolique. Toutefois, bien que les calculettes actuelles offrent des fonctions mathématiques propres à satisfaire

la majorité des utilisateurs, elles sont, et c'est un point de vue personnel, trop orientées «mathématiques», pèchent par le manque de généralité des opérations sur les matrices, et peuvent être complexes à maîtriser (cas des traitements symboliques). Pourtant, il existe un langage de programmation puissant, relativement simple, offrant la plus grande généralité, il s'agit bien sûr d'APL. Pour une utilisation sur le terrain, il serait idéal car puissant, donc concis, les programmes n'en seraient que plus courts à frapper au clavier, ce qui est fondamental pour une calculette de poche.

#### Une calculette programmable en APL aurait plusieurs avantages :

- On disposerait de la puissance d'APL dans le creux de la main, et rapidement (plus de phase de "boot" interminable comme sur nos PC),
- Elle serait puissante mais pas trop complexe à programmer grâce à la concision d'APL, et sa documentation ne devrait pas être trop épaisse,
- Le prix serait abordable,
- Et enfin, on peut espérer qu'elle serve indirectement de vecteur pour la promotion des langages orientés «tableaux» (tels qu'APL et J), auprès de l'industrie, de l'enseignement et du public.

#### **Maquette informatique de calculette programmable APL**

---

J'ai donc développé une maquette de faisabilité sur PC sous Windows afin de m'assurer de la pertinence de l'idée. Cette maquette a été écrite en Visual Basic, sans réel soucis d'optimisation du code. Par rapport à une programmation avec un APL sous Windows, cela m'a fait perdre un peu de temps en programmation, mais d'un autre côté, cela m'a permis de retrouver modestement, les algorithmes de base d'un interpréteur APL. Afin de faciliter l'écriture de cette maquette, et de réduire le temps de développement, des simplifications ont eu lieu, je l'espère provisoirement.

#### **ERGONOMIE GENERALE DE LA CALCULETTE**

Parmi les hypothèses de départ qui ont conditionné le développement de la maquette, il a fallu décider en premier, de son ergonomie générale et de son "scénario" d'utilisation.

En premier, il était tentant de reprendre une configuration comprenant un écran LCD, et un clavier ASCII classique, éventuellement agrémenté de touches dédiées à des fonctions APL. Cela aurait conduit à réinventer un ordinateur de poche programmable en APL, au lieu du classique BASIC. C'est peut être la bonne configuration à terme, mais en attendant, je lui en ai préféré une autre.

La configuration choisie est celle d'une calculette dont la forme et le clavier sont dans la plus pure tradition des calculettes programmables (telles les HP 11C et HP 41C qui me servent de référence). Ainsi, il est clair pour l'utilisateur que la fonction

primordiale de cette calculatrice est de faire du calcul, et pas du traitement de texte, ou autre chose.

## NOTATION POSTFIXEE

Une fois défini ce "look and feel", il a fallu se préoccuper de la façon dont on effectue les calculs. J'ai choisi d'utiliser la notation polonaise inverse.

Les avantages de cette notation sont bien connus :

- La visualisation des résultats intermédiaires se fait avec une pile de données de type LIFO,
- Il n'y a plus de parenthèses, donc jamais plus de doutes quant au nombre de parenthèses restant à fermer.

Seul inconvénient :

Il faut s'habituer à réfléchir en notation post-fixée. Au début c'est assez surprenant, mais après, on fini par trouver cela très naturel !

La conséquence de ce choix est que nous venons donc de post-fixer l'écriture d'APL, et plus précisément, de mettre APL à la sauce FORTH !

Si nous ajoutons qu'afin de faciliter le développement de la maquette informatique, les caractères APL ont laissé place à des mots clés alphanumériques (Shape, Ravel, Iota, ...), cela méritait alors de renommer le langage. BABAPL, tel est son nom ! Après coup, j'ai pensé qu'on aurait pu l'appeler AFAPL : Application du Forth à APL !

Quelques exemples d'utilisation de la calculatrice BABAPL :

- Création d'un tableau de dimensions 3 x 4, contenant la suite ordonnée des nombres allant de 1 à 12. Il faut effectuer la séquence suivante (appuis sur les touches) :

1	2	Iot a	3	Ente r	4	Reshape
---	---	----------	---	-----------	---	---------

- Multiplication de ce tableau par le scalaire 10. Il faut effectuer la séquence suivante :

1	0	x
---	---	---

- Réduction par l'addition de ce tableau, le long de l'axe des colonnes :

f/	+
----	---

**FONCTIONNALITES DE BASE DU LANGAGE DE PROGRAMMATION BABAPL****• Le langage et sa pile de données :**

BABAPL est avant tout un mini-APL en notation post-fixée. Une conséquence est qu'il n'y a plus d'opérateur tantôt monadique, tantôt dyadique selon le nombre d'opérandes autour de l'opérateur considéré. Ainsi Shape et Reshape sont deux fonctions primitives distinctes, l'une monadique, et l'autre dyadique.

La pile de données est symbolique dans le sens où les objets qu'elle contient peuvent être des scalaires, des vecteurs, ou des tableaux. Les fonctions primitives n'agissent que sur la pile de données (le sommet et éventuellement le sous-sommet, selon que la fonction primitive considérée est monadique ou dyadique).

Enfin, il est à noter que par défaut, la calculatrice BABAPL attend la saisie d'un vecteur. La création sur la pile du vecteur (10 2 33 -4) s'effectue alors comme suit :

1	0	Ente r	2	Ente r	3	3	Ente r	+/-	4	=>V
---	---	-----------	---	-----------	---	---	-----------	-----	---	-----

**• Les types de données :**

BABAPL ne traite que des valeurs numériques réelles uniquement. Les types de données manipulés sont les scalaires, les vecteurs et les tableaux à deux dimensions. Conséquence, il n'y a plus de fonction Execute, et pour l'instant, le vecteur vide n'est pas géré mais reconnu. BABAPL convertit d'office un vecteur d'un élément, en scalaire, et une matrice de dimension 1 x n (ou n x 1), en vecteur. C'est un choix provisoire, qui pourra être revu plus tard.

**• Les noms de variables, de fonctions, et leurs déclarations :**

Les identificateurs de variables et de fonctions sont alphanumériques. La déclaration des variables est implicite comme en APL. Le système effectue automatiquement un compactage de la mémoire des données, lors des stockages en mémoire. C'est pénalisant en temps de calcul, mais très utile lorsque l'on a peu de mémoire pour stocker les données. Enfin, il n'y a pas de variables "système" pour l'instant (tolérance de comparaison, ...).

**• Les fonctions :**

Il n'y a pas de notion de programme principal, puisque toute fonction peut être appelée par l'utilisateur (touche de fonction XEQ), ou par un programme appelant (instruction XEQ).

Il n'y a plus d'arguments car ces derniers sont déposés sur la pile avant l'appel de la fonction. Il peut donc y en avoir un nombre quelconque.

Il n'y a pas de variables locales, puisqu'elles sont aussi sur la pile. Pour faciliter leur accès, des fonctions de manipulation de la pile existent (Dup, Swap, Drop, ...). Indirectement cela permet de définir des fonctions sans variables explicites, un peu comme en J.

Exemple de fonction sans variables locales, avec le classique calcul de la moyenne, dont le vecteur des valeurs est déposé sur la pile avant l'appel de la fonction :

Dup    f /        +        Swap    Shape    %        Rtn

- **Les structures de contrôle :**

BABAPL est un peu plus explicite qu'APL au niveau des branchements, on y trouve des instructions dont la signification se devine aisément : GOTO, IF, XEQ (appel de fonction), RTN (sortie de la fonction), et END. Pour les sauts, il faut employer des étiquettes qui ne sont visibles que dans la fonction où elles sont déclarées. Dans l'avenir, BABAPL sera enrichi d'instructions de contrôle propres à faciliter la programmation de calculs récurrents, toujours avec l'objectif de minimiser le codage du programme, et donc la frappe au clavier.

- **Les entrées/sorties :**

L'affichage via l'écran LCD visualise les 4 premiers éléments de la pile de données, et le mode VIEW permet de visualiser tous les éléments d'un objet (VIEW appelle un mini-tableur).

- **Les caractéristiques supplémentaires :**

L'espace de travail courant (pile + données + fonctions utilisateurs) est sauvegardé en "mémoire constante" lors de l'arrêt de la calculatrice, et restauré automatiquement au démarrage. Pour l'instant, il n'y a pas de notion d'espaces de travail.

- **Les modes de fonctionnement :**

La calculatrice fonctionne selon différents modes en fonction de ce que l'on veut réaliser :

- RUN : mode par défaut, on peut y effectuer des calculs, exécuter des fonctions, et aussi changer de mode.
- EDIT : mode d'édition des fonctions (un peu comme le classique DEL Editor).
- PCL : mode pour l'effacement des programmes.
- VCL : mode pour l'effacement des variables.
- VIEW : mode de visualisation des objets (scalaires, vecteurs, tableaux).
- MEM : mode d'affichage de l'occupation mémoire, des noms de variables et de fonctions.
- INFO : mode d'affichage des coordonnées de l'auteur, et de l'AFAPL.

## **Conclusion**

Loin d'une présentation détaillée de la calculette, cet aperçu avait pour but de présenter les concepts de base de cette maquette de calculette APL. Mon objectif est maintenant de fiabiliser le fonctionnement de cette maquette afin de la tester pour juger de sa facilité d'emploi, et de réaliser en parallèle une documentation simple mais complète.

A court terme, j'envisage d'implémenter notamment la fonction Domino, l'opérateur d'axe, et de faire passer le rang maximal des tableaux à 3. La visualisation de courbes 2D et 3D sera un objectif à moyen terme, toujours dans le but de proposer un moyen de visualisation simple et ne nécessitant pas de programmation.

D'orès et déjà, les remarques de tout le monde sont les bienvenues, et seront forts utiles pour améliorer cette modeste maquette.

# APL98 à Rome

par Paolo Di Chio

*NdlR : Le congrès APL98 aura lieu à Rome du 27 au 31 juillet 1998.*

*Il est organisé par SIGAPL italien, voilà l'information qu'ils ont envoyé sur mon e-mail :*

APL98 Rome

July, 27th-31st, 1998

University of Rome "Tor Vergata" - Faculty of Economics

The Array Processing Languages Conference

**"New gems from old roots"**

# Les joyeusetés du correcteur orthographique

par la Rédaction

## Suggestions du correcteur:

Remplacer "logistic" par "logis tic" (il est souvent toc, ce correcteur...)

Remplacer "map" par "cap" ou "gap" ou "rap" ou "mal" ou " mat" ou "mât" (quelle culture!)

Remplacer " May" par "Mamy", "Gay", "Mac" (plusieurs interprétations possibles...)

Remplacer "Gallois" par "Gaulois" ou "Gallons" (Gaulois il l'était certainement)

Remplacer "trigo" par "trio" ou "frigo"

Remplacer "Euro" par "Eure" (et les autres alors?)

Remplacer "Euros" par "Éros" (c'est beau mais incorrect...)

Remplacer "PC" par "CCP" (encore un C et cela me rappellerait des souvenirs)

Remplacer "ISO" par "ISSN" ou "IPSO" (au choix de notre Président)

Remplacer "Printer" par "Pointer"

Remplacer "dpi" par "épi"

Remplacer "inch" par "inca"

Remplacer "Euler" par "Éculer" ou "Émuler" (juste pour le plaisir de mettre les accents sur lesÉ)

Remplacer "Guinness" par "Guindés", "Guinées", "Ruines", "Gaines", "Guides", "Guises" (qui a dit que l'on peut se passer du correcteur?)

Remplacer "of" par "bof" (il donne la réponse...)

Remplacer "Kanada" par "Canada"

Remplacer "loop" par "loup" ou "flop"

Remplacer "Plouffe" par "Pouffe"

Remplacer "feel" par "fiel" ou "fuel" ( et si on les mélangeait?)

Remplacer "Reshape" par "Rescapé" (de la calculette APL?)

Remplacer "fiabiliser" par "viabiliser".