Les caractères Unicode

1. Introduction

Alors que chaque caractère ASCII correspond à un octet, le codage des caractères Unicode peut s'étendre sur plusieurs octets. Considérons par exemple la chaîne suivante :

$$\Delta E = \Delta m \cdot c^2$$

Nous allons découper cette chaîne en ses octets constitutifs, au moyen du programme suivant :

```
dim a$, c$, i%

a = "ΔE = Δm · c²"
for i = 1 to len(a)
  c = mid(a, i, 1)
  print i, c, asc(c)
next i
```

***Note** : Pour afficher correctement le listing dans l'éditeur FBCroco, utilisez une police de caractères compatible avec l'Unicode, par exemple Consolas. La police Amstrad n'est pas compatible !*

Pour chaque octet, le programme écrit le numéro de l'octet, suivi du caractère ASCII correspondant et de sa valeur numérique. Le résultat (sur la console) est le suivant :

```
 1            ╬            206
 2            ö            148
 3            E            69
 4                         32
 5            =            61
 6                         32
 7            ╬            206
 8            ö            148
 9            m            109
 10                        32
 11           Ô            226
 12           ê            136
 13           Ö            153
 14                        32
 15           c            99
 16           ┬            194
 17           ▒            178
```

On voit facilement que les caractères « Δ » et « ² » prennent 2 octets chacun, tandis que le caractère « · » en prend 3 !

2. La bibliothèque utf8

La bibliothèque **utf8** de F. Hoogerbeets fournit des alternatives aux fonctions classiques de traitement des chaînes de caractères, utilisables pour les caractères Unicode codés selon la norme UTF-8. Avec ces nouvelles fonctions le programme précédent devient :

```
#include "utf8.bi"

dim a$, c$, i%

a = "ΔE = Δm · c²"

for i = 1 to ulen(a)
  c = umid(a, i, 1)
  print i, c, uasc(c)
next i
```

La première ligne inclut le fichier d'en-têtes de la bibliothèque. Les fonctions spécifiques de l'unicode sont préfixées par **u** (**ulen**, **umid**, **uasc** …). Les résultats sont les suivants :

```
 1            ╬ö           916
 2            E           69
 3                        32
 4            =           61
 5                        32
 6            ╬ö           916
 7            m           109
 8                        32
 9            ÔêÖ          8729
 10                       32
 11           c           99
 12           ┬▒          178
```

On voit que chaque caractère est traité individuellement, quel que soit le nombre d'octets.

Les fonctions apportées par cette bibliothèque sont les suivantes :

uasc uchr uleft uright umid ulen uinstr : comme dans FBCroco

ulcase(a$) : mise en minuscule

uucase(a$) : mise en majuscule

uinsert(a$, b$, i%) insère la chaîne **b$** dans la chaîne **a$** à la position **i%**

ureverse(a$) écrit la chaîne à l'envers

En plus, les fonctions suivantes retournent TRUE ou FALSE :

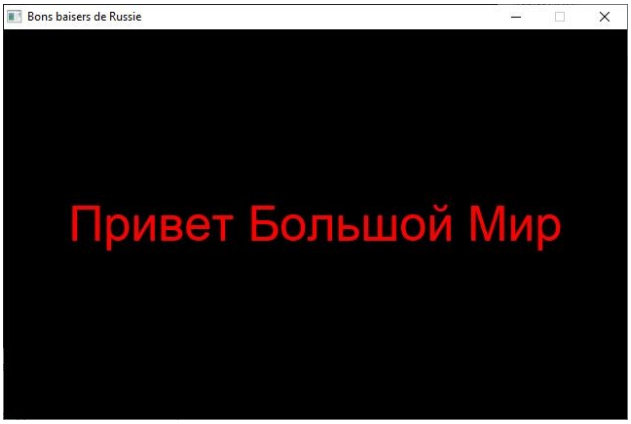
uisascii(a$) teste si la chaîne **a$** est une chaîne ASCII

uivstr(a$) teste si la chaîne **a$** est une chaîne UTF-8 valide

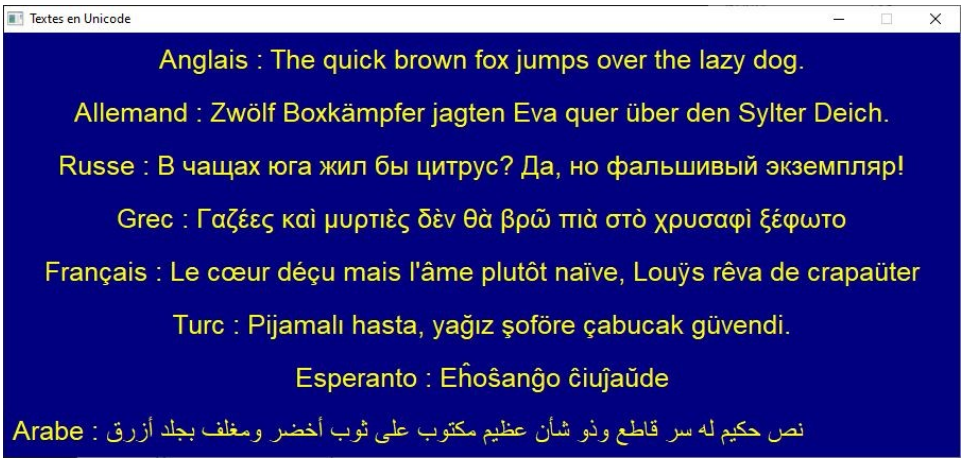
3. Chaînes Unicode en mode graphique

Les fonctions graphiques de FLTK permettent d'afficher des chaînes unicode dans le mode graphique de FBCroco. Les 5 programmes exemples fournis avec cet *addon* montrent les possibilités :

Le programme `test1.bas` affiche ce qui semble être l'équivalent russe de « Bonjour tout le monde » :



Le programme `test2.bas` affiche une série de phrases en différents langages :



Les 6 premières phrases sont tirées d'un exemple fourni avec FLTK. Les phrases en Espéranto et en Arabe ont été proposées sur le forum PANORAMIC, respectivement par « *Ouf ça passe* » et par « *papydall* », que je remercie.

La phrase arabe n'est pas centrée ; cela pourrait être dû à la présence de codes non imprimables, notamment ceux indiquant que la lecture se fait de droite à gauche.

Le programme `test3.bas` reprend la phrase en Espéranto et l'affiche sous différentes configurations (couleur, taille, inclinaison) choisies aléatoirement :

Le programme `test4.bas` reprend l'exemple de la section 1 en affichant les codes hexadécimaux fournis par les fonctions **`uasc`** et **`asc`**. Conformément à la norme UTF-8, les deux codes sont identiques pour la première série de caractères ASCII (code inférieur à 127 = &h7F).

Le programme `test5.bas` affiche des caractères semi-graphiques de type *Emoji*.

A cette fin nous avons inclus dans le fichier **`emoji.bi`** une liste de 1117 caractères compatibles avec la police `FL_HELVETICA` de FLTK. Le programme **`emoji.exe`** (dont le code source est en FreeBASIC) permet de choisir les caractères :