Stringbolge

Un langage semi-ésotérique pour l'extraction de données depuis une string

Explicature

Introduction

Le besoin initial

Le besoin qui a motivé le développement du langage **Stringbolge** est tout simplement le suivant. Soit une string de la forme suivante :

```
{abcd} {efgh {ijkl}}
```

Je veux pouvoir extraire la sous-string `efgh`, en sachant bien évidemment que :

- Les strings `abcd`, `efgh` et `ijkl` sont de longueur quelconque.
- Elles peuvent contenir des espaces, y compris en fin
- On partira aussi de l'hypothèse simplificatrice qu'elles ne peuvent pas contenir d'accolades `{` ou `}`

En pratique, partant de la string

```
{Charles} {de Gaulle  {Général}}
```

Je désire obtenir le résultat "`de Gaulle  `" (car il y a bien *deux* espaces entre `Gaulle` et `{`).

Il est très certainement possible d'utiliser une expression régulière pour extraire la string en question. Je préfère cependant une version plus procédurale :

1. Commencer au début de la string.
2. Passer le premier caractère (qui est une accolade)
3. Avancer jusqu'à une accolade.
4. Avancer d'un caractère.
5. Retenir la position courante.
6. Avancer jusqu'à la séquence " {"
7. Conserver tous les caractères entre la position retenue en (5) et la position juste précédant la position courante.

Principes de base de Stringbolge

Stringbolge agit sur une string de départ appelée *input* et produit une string résultat dénommée *output*.

On dispose d'un nombre quelconque de *marqueurs*. Un marqueur dénote une position dans la string d'input. Cette position peut être soit la position d'un caractère particulier, soit *<eof>* dénotant une position juste après le dernier caractère.

Stringbolge possède des instructions capables de déplacer les marqueurs dans la string d'*input*, ainsi que d'instructions capables de copier vers l'*output* une partie de l'input entre deux marqueurs. Il n'est pas possible de modifier l'input.

Exemple

Reprenons notre exemple en montrant le déplacement des marqueurs. Les deux marqueurs sont représentés par ^ et v

- Commencer au début de la string.

```
v
{Charles} {de Gaulle  {Général}}
^
```

- Passer le premier caractère (qui est une accolade)

```
v
{Charles} {de Gaulle  {Général}}
 ^
```

- Avancer jusqu'à une accolade.

```
v
{Charles} {de Gaulle  {Général}}
          ^
```

- Avancer d'un caractère.

```
v
{Charles} {de Gaulle  {Général}}
           ^
```

- Retenir la position courante.

```
           v
{Charles} {de Gaulle  {Général}}
           ^
```

- Avancer jusqu'à la séquence " {"

```
v
{Charles} {de Gaulle  {Général}}
                     ^
```

- Conserver tous les caractères entre la position retenue en (5) et la position juste précédant la position courante.

```
           v
{Charles} {de Gaulle  {Général}}
                     ^
```

Programmer en Stringbolge

Un programme Stringbolge est une suite linéaire d'instructions individuelles. Chaque instruction utilise un argument textuel. Par exemple, l'instruction Find utilise un argument textuel qui donne le pattern à chercher dans l'input.

Syntaxe Stringbolge

La syntaxe Stringbolge pure utilise une unique lettre pour dénoter une instruction. Une lettre-instruction peut être suivie de rien, d'un nombre ou d'une string.

Instruction	Nom	Argument	Description
A		Nombre	Advance – Avance le marqueur courant de N caractères.
B		Nombre	Back – Recule le marqueur courant de N caractères
C		Nombre	Copy – Copie depuis un marqueur jusqu'au marqueur courant dans l'output
D		Nombre	Delete – Supprime les N derniers caractères de l'output
E		Aucun	End – Avance le marqueur courant sur l'EOF
F		String	Find – Avance le marqueur courant jusqu'au premier caractère du pattern suivant la position courante.
G		Nombre	Goto – Met le marqueur courant à la même position que le marqueur nommé.
H		String	Hollerith – Ajoute le pattern à l'output
I		Aucun	Ignore – Passe en mode ignore
J		String	Jump – Avance le marqueur courant jusqu'au premier caractère qui est dans le pattern
K		Aucun	Keep – Passe en mode keep
L		Aucun	Loop - Démarre une boucle
M		Nombre	Mark – Copie le marqueur courant dans le marqueur nommé
N		Aucun	Next – Fin de la boucle
O		String	Over – Avance le marqueur courant jusqu'au premier caractère qui n'est pas dans le pattern.
P			
Q		Aucun	Quit – Quitte la boucle
R		String	Reverse – Recule le marqueur courant jusqu'au premier caractère du pattern qui précède la position courante
S		Aucun	Start – Place le marqueur courant sur le début de la string
T		Aucun	Try - Début de bloc conditionnel
U			
V			
W			
X			
Y			
Z		Aucun	Zero – Vide l'output

Argument

L'argument suit immédiatement la lettre de l'instruction.

Lorsque l'argument est un nombre, il est constitué de tous les chiffres qui suivent l'instruction, jusqu'à un caractère qui n'est pas un chiffre. Si aucun chiffre n'est précisé, la valeur de l'argument est 1.

Lorsque l'argument est une string, il est soit :

Unquoted ; il est alors constitué du caractère qui suit immédiatement l'instruction (qui peut être quelconque à l'exception d'une double quote, ainsi que de tous les caractères suivants jusqu'à la fin du string ou une lettre capitale.

```
FHelloHTrouvé
```

Quoted ; il est alors constitué de tous les caractères entre le double-quote qui suit immédiatement l'instruction et le double-quote suivant. Il est possible d'inclure un double-quote dans la string en le doublant. Cette notation est indispensable si l'argument contient une lettre capitale ailleurs qu'en première position

```
F"HELLO"HTrouvé
```

Exemple

Notre extracteur s'écrit en syntaxe compacte:

```
NF{NMF {C
```

Syntaxe Stringbol

La syntaxe Stringbol est beaucoup plus lisible, mais aussi beaucoup plus verbeuse. Le but est de ressembler le plus possible à du COBOL.

Exemple

Notre extracteur s'écrit en syntaxe Stringbol :

```
ADVANCE ONE CHARACTER.
ADVANCE UNTIL FINDING "{".
ADVANCE ONE CHARACTER.
RECORD CURRENT POSITION INTO MARKER 1.
ADVANCE UNTIL FINDING " {".
COPY FROM MARKER 1 TO CURRENT POSITION INTO OUTPUT.
```