DEUG MIAS2. Institut Galilée
Option : Langages formels et compilation. 1999–2000
TP 1.

Ce sujet très élémentaire a pour but de vous faire manipuler des mots. Turbo–Pascal propose le type `string` qui est très similaire au type `mot` qui nous intéresse.

- La déclaration d'une variable `s` : `string` équivaut à celle d'un tableau

`s : array[0..255] of char.`

La longueur `l = length(s)` de `s` est codée par la composante d'indice `0` :

`l = ord(s[0])`

et, pour chaque `n` dans le domaine `[1..l]`, `s[n]` est le `n`–ième caractère de `s`.

- Nous prendrons la définition suivante du type `mot` :

```
type mot = record
                  ch :  array[1..Max] of char ;
                long :  integer
           end ;
```

où `Max` est une constante que vous choisirez en fonction de vos besoins.

I. Primitives relatives aux mots.

Ecrire les primitives suivantes :

`procedure motvide( var u : mot ) ;` construire le mot vide, c'est–à–dire, schématiquement : `u` := (),

`procedure adjoindre( var u : mot ; x : char ) ;` adjoindre le caractère `x` à la fin du mot `u` c'est–à–dire : `u` := `ux`,

`procedure lire( var u : mot ) ;` donner à `u` la valeur correspondant à une chaîne de caractères lue à l'écran, *cette procédure est la seule où doit intervenir une variable de type* `string`,

`procedure retirer( var u : mot ) ;` retirer le dernier caractère d'un mot c'est–à–dire : si `u = vx` alors `u` := `v`,

`function longueur( u : mot ) : integer ;` renvoyer la longueur d'un mot,

`function nieme( u : mot ; n : integer ) : char ;` renvoyer le `n`–ième caractère du mot `u`,

`procedure ecrire( u : mot ) ;` écrire à l'écran la chaîne de caractères correspondant au mot `u`.

II. Applications.

Ecrire des programmes n'utilisant que les primitives que vous venez de définir (les lectures et écritures se feront toutes au clavier et à l'écran). Par exemple :

- Inverser un mot (c'est–à–dire, transformer un mot en son image miroir).
- Tester si un mot est un palindrome (c'est–à–dire, tester s'il est égal à son inverse).
- Concaténer deux mots.
- Comparer deux mots.
- Tester si un mot `v` est un sous–mot du mot `u` : si le test est positif, il devra permettre de localiser la première occurrence de `v` dans `u`.
- Tester si un mot est correctement parenthésé.