

# Manipulation de chaînes de caractères



## I - Que voulons-nous que l'ordinateur fasse ?

Nous voudrions créer une chaîne de caractères à partir des éléments d'une autre chaîne en vu de coder/décoder des messages.

### a. Qu'est-ce qu'une chaîne de caractère

C'est un ensemble ordonné, écrit entre guillemets et contenant un certain nombre d'éléments.

Par exemple, on crée la chaîne C1

```
C1:="wxcvbn12345* !+##-"
```



#### Attention !

Notez que les éléments de la chaîne peuvent être tout à fait quelconques et sont considérés par l'ordinateur comme des sigles sans significations particulière.

On peut créer également une chaîne vide

```
V:=""
```

à ne pas confondre avec la chaîne contenant un espace

```
W:=" "
```



#### Attention !

Une espace n'est pas le vide...

### b. Opérations sur les chaînes

Comme une chaîne est ordonnée, ses éléments ont chacun un rang.

Regardez par exemple ce que donne :

```
C1[2];  
C1[1];  
W[0];
```



#### Attention !

XCAS commence à compter à partir de zéro !

On peut obtenir la taille d'une chaîne avec la commande `size` qui donne le nombre d'éléments de la chaîne

```
size(C1)
```

```
size(V)
```

```
size(W)
```

Si on a oublié un terme, on peut le rajouter à la fin de la liste avec la commande `concat` (chaîne, élément)

```
concat(C1,u)
```

On dit qu'on *concatène* la chaîne C1 avec la chaîne "u".

## II - Reste de la division euclidienne

Nous allons bientôt traiter abondamment la division, mais vous savez déjà que, par exemple, le reste de la division euclidienne de 37 par 5 est...

```
irem(37,5)
```

## III - Pour...allant de...à...faire...

On veut calculer la somme des entiers naturels de 0 à 50<sup>a</sup>.

En fait, nous voulons calculer  $1 + 2 + 3 + 4 + \dots + 49 + 50$

Nous allons pour cela créer une *boucle*, mécanisme essentiel en informatique.

Nous allons ainsi comprendre le fonctionnement des « cases mémoire ».

Notons S notre somme. Au départ elle est nulle.

```
S:=0
```

Cette ligne signifie qu'une case-mémoire est appelée S et qu'elle contient pour l'instant 0.

Il faut commencer par ajouter 1. S devient S + 1.

En fait, la case-mémoire S contient maintenant 1 et non plus 0.

On ajoute alors 2. S devient S + 2.

La case-mémoire S contient maintenant 3 et non plus 1.

On ajoute alors 3. S devient S + 3.

La case-mémoire S contient maintenant 6 et non plus 3.

On ajoute alors 4. S devient S + 4.

La case-mémoire S contient maintenant 10 et non plus 6.

Bon, ça risque d'être un peu long... Alors on va expliquer à l'ordinateur que

– au départ S vaut 0

– POUR j ALLANT DE 1 À 50 FAIRE S :=S+j

– on s'arrête quand j vaut 50.

en XCAS, ça s'écrit

```
S:=0;
```

```
for(j:=0;j<=50;j:=j+1){S:=S+j}
```

## IV - Code ASCII

La mémoire de l'ordinateur conserve toute donnée sous forme numérique. Pour coder chaque touche du clavier, on utilise depuis les années 60 le code ASCII (American Standard Code for Information Interchange).

On dit que c'est un code 7 bits, c'est-à-dire que chaque caractère est codé par une liste de 7 chiffres égaux à 0 ou 1 : combien de caractères différents peut-on ainsi coder?...

Par exemple les codes 65 à 90 représentent les majuscules, les codes 97 à 122 représentent les minuscules.

Une fonction XCAS donne ces nombres : `asc("caractère")`

<sup>a</sup>. Il existe une fonction XCAS qui le fait directement, mais là n'est pas le problème...

```
asc("a")
```

```
asc("Where is Brian ?")
```

On peut également revenir au caractère en partant du code avec char (codes)

```
char(85)
```

```
char(66,114,105,97,110,32,105,115,32,105,110,32,116,104,101,32,107,105,99,104,101,110)
```



## V - Le programme...

On dispose d'une chaîne :

```
c1:="abcdefgh";
```

Le but de notre jeu passionnant est de créer une chaîne comprenant ces mêmes lettres mais en majuscules!

Décomposons l'action au ralenti :

- On ouvre une fenêtre de programmation en tapant simultanément sur  et 
- Une ligne pour introduire le programme

```
majuscule(c):={
```

On l'appelle `majuscule` et il dépend de la donnée d'une variable `c` qui sera pour nous une chaîne de caractère en minuscules.

- une ligne pour introduire les variables locales.

Comme on ne les connaît pas a priori, on laisse un vide qu'on remplira au fur et à mesure :

```
local ;
```

- une ligne pour créer notre chaîne des majuscules.

Pour l'instant elle est vide car on n'a pas analysé la chaîne entrante :

```
C:="";
```

- le cœur du programme :

```
for(j:=0;j<size(c);j:=j+1){ C:=concat(C,char( asc(c[j])-32 )) };
```

Expliquez ce que l'ordinateur va faire.

- on ferme

```
}
```

## VI - Comment l'utiliser ?

Pour obtenir "maman" en majuscule tapez

```
majuscule("maman")
```

## VII - Sans parachute...

À vous d'imaginer une procédure **minuscule("chaîne")** qui transforme un texte majuscule en minuscule.

Vous pouvez même aller jusqu'à créer une procédure qui laisse en majuscule la première lettre.

La prochaine étape sera de mettre un espace puis une majuscule après un point...

## VIII - Morituri te salutant

Nous sommes maintenant armés informatiquement pour affronter la programmation du chiffrement de César.

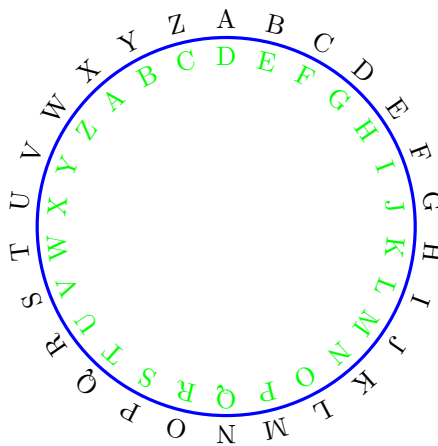
### a. Un peu d'histoire

Comme le disait Suétone (70-127) dans La vie des 12 Césars :

On possède enfin de César des lettres à Cicéron, et sa correspondance avec ses amis sur ses affaires domestiques. Il y employait, pour les choses tout à fait secrètes, une espèce de chiffre qui en rendait le sens inintelligible (les lettres étant disposées de manière à ne pouvoir jamais former un mot), et qui consistait, je le dis pour ceux qui voudront les déchiffrer, à changer le rang des lettres dans l'alphabet, en écrivant la quatrième pour la première, c'est-à-dire le D pour l'A, et ainsi de suite.

### b. Un peu de bricolage

Que vous inspire ce dessin :



### c. Un peu de mathématiques

Codons chaque lettre par un nombre :  $A \mapsto 0, B \mapsto 1, \text{etc.}$

Comment modéliser la chiffrement de César ?

Y a-t-il un problème ?

Qu'ont en commun 29 et 3 ?

### d. Un peu d'informatique

Les américains ont mis au point le code ASCII : ils ont donc oublié de coder nos lettres accentuées...

De plus, parmi toutes les touches codées, seules celles contenant un certains nombre de caractères nous intéressent. Que fait cette procédure :

```
code:= (c)->{
if (c=='é') return(100) ;
if (c=='è') return(99) ;
if (c=='à') return(98) ;
if (c=='ç') return(97) ;
if (c=='ù') return(96) ;
if (c=='ê') return(95) ;
.
.
.
return(asc(c)-32);
};
```

À vous de terminer cette procédure en vous occupant des majuscule puis d'imaginer la procédure decode...

Il faudra ensuite créer une procédure codant et décodant selon le chiffre de César. En voici le cœur :

```
messcode:=concat(messcode, decode(iirem(c1e+code(message[j]),106)));
```

À vous de reconstituer le reste...