INFORMATIQUE 1

Initiation à la programmation

I - Avant de commencer

XCAS est un logiciel libre de calcul formel. En particulier, vous allez pouvoir le télécharger gratuitement en toute légalité sur

`http://www-fourier.ujf-grenoble.fr/~parisse/irem.html`

Nous verrons plus tard ce que veut dire calcul formel. Pour aujourd'hui, nous commencerons doucement puisque le but de notre séance est de créer un programme qui remplie une *chaîne de caractères* à partir d'une autre.

II - Allumons l'ordinateur

Je pense que vous en êtes capables... Vous lancez XCAS.
Comme nous sommes très fort, nous choisirons le module **Université**.
Nous irons ensuite dans le menu de configuration pour choisir une police de taille 14 au lieu de celle de 20 un peu trop grosse.
Nous voici prêts à programmer.

III - Que voulons-nous que l'ordinateur fasse ?

Nous voudrions créer une chaîne de caractères à partir des éléments d'une autre chaîne.

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*␣!+#-"
```

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:="␣"
```

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]
```

XCAS commence à compter à partir de zéro !

```
W[0]
```

```
V[0]
```

Une espace n'est pas le vide...

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".

IV - Procédure

Une procédure est un programme qui a un nom, un début et une fin et qui dépend d'un certain nombre de paramètres fournis par l'utilisateur.
Par exemple

```
plus(a,b):={  // c'est le titre du programme qui dépend de la donnée
                 de deux variables a et b

 ...          // il faudra rentrer des instructions ici

 }            // marque la fin du programme
```

V - Fonction

Il faudrait trouver un moyen de faire calculer, par exemple, $a+b$ pour différentes valeurs de a et b que nous entrerons.
Il faut donc expliquer à l'ordinateur qu'il doit transformer a et b en $a+b$. On symbolise cette action par une flèche

```
(a,b)->a+b
```

cela veut dire pour XCAS qu'il doit transformer deux nombres quelconques a et b en $a+b$.
Pour mieux utiliser ce mécanisme, on lui donne un nom. Choisissons par exemple **f**. Pour faire comprendre à xcas que cette lettre **f** désigne la transformation, on utilise :=

```
f:=(a,b)->a+b
```

Alors, pour calculer par exemple le « transformé » de (4,2) par **f** on rentre

```
f(4,2)
```

Ben oui, $4+2=6$...

VI - 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)
```

VII - Pour...allant de...à...faire...

On veut calculer la somme des entiers naturels de 0 à 50[a].
En fait, nous voulons calculer $1+2+3+4+\cdots+49+50$
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}
```

Nous y sommes presque....

VIII - Variables locales

On peut être amené, à l'*intérieur* d'un programme, à introduire une variable intermédiaire comme ici j.
On l'introduit par

```
local j;
```

IX - 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")

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

```
asc("Where␣is␣Brian␣?")
```

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

[a] Il existe une fonction XCAS qui le fait directement, mais là n'est pas le problème...

```
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)
```

X - 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 Alt et P
- 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 ))  };
  ```

- on ferme

  ```
  }
  ```

XI - Comment l'utiliser ?

Pour obtenir "maman" en majuscule tapez

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

XII - 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...

XIII - Morituri te salutant

Nous sommes maintenant armés informatiquement pour affronter la programmation du chiffrement de César, mais nous avons besoin d'une petite introduction historico-mathématique : c'est l'objet du prochain cours...