- LLes chaînes de caractères

I) Le type chaîne de caractères

Une chaîne de caractère est constituée de caractères. On peut la représenter dans une expression sous la forme de la suite de caractères, placée entre apostrophe (simple ou double).

```
-->'Une chaîne de caractères'
 ans  = Une chaîne de caractères
-->"Une autre"
 ans  = Une autre
```

Il faut doubler les apostrophes pour les considérer dans une chaîne

```
-->'L''été'
 ans  = L'été
```

II) Opérations sur les chaînes de caractères

La fonction *length*() appliquée à une chaîne fournit le nombre de caractères de celle-ci.

```
-->s='Une chaîne de caractères'
 s  = Une chaîne de caractères
-->length(s)
 ans  =  24.
```

La fonction de *concaténation* de chaînes de caractères est représentée par le symbole +.

```
-->s+' plus longue'
 ans  = Une chaîne de caractères plus longue
```

La fonction *string*() permet de convertir une valeur numérique en une chaîne de caractères

```
-->a=string(2+3)
 a  = 5
-->b=string(1/3)
 b  =  .3333333
-->a+b
 ans  = 5 .3333333
```

La fonction *evstr*() permet de convertir une chaîne de caractères en une expression :

```
-->c='sqrt(3)/2'
 c  = sqrt(3)/2
-->d=evstr(c)
 d  =   .8660254
```

La fonction *part*() permet d'extraire une sous chaîne grâce à un nombre ou un tableau

```
-->part('abcdefg',2)
 ans  = b
-->part('abcdefg',[1 2 6])
 ans  = abf
```

Le fonction *strindex*() permet de rechercher une chaîne à l'intérieur d'une autre chaîne. La fonction fournit les indices de début de toutes les occurrences de la chaîne recherchée.

```
-->strindex('Mississipi','ss')
 ans  = !   3.    6. !
-->strindex('Mississipi','i')
 ans  = !   2.    5.    8.    10. !
```

III) Définition d'un caractère

1) Les caractères en Scilab

Scilab utilise une convention qui lui est propre pour représenter les caractères. Il est possible de connaître cette représentation au moyen de la fonction *str2code* :

```
-->str2code('abc')
 ans  =
!   10. !
!   11. !
!   12. !
```

Réciproquement, il est possible de synthétiser une chaîne de caractères à partir d'un tableau contenant les codes des caractères que l'on désire obtenir :

```
-->code2str([10 11 12])
 ans  = abc
```

De manière plus générale, on pourra utiliser cette représentation d'une chaîne de caractère chaque fois que l'on désirera manipuler, transformer cette chaîne. Ainsi, on remarquera que les lettres minuscules sont représentées par les entiers consécutifs de 10 à 35 et que les lettres majuscules sont représentées par l'opposé de ces mêmes codes :

```
-->str2code('azAZ')
 ans  =
!   10. !
!   35. !
! - 10. !
! - 35. !
```

2) La représentation ASCII (American Standart Codes for Information Interchange)

La codification la plus utilisée dans le monde est la représentation ASCII dont le tableau suivant donne le code ASCII standart, comportant 128 caractères numérotés de 0 à 127. On utilise en Europe une extension du code ASCII, l'ISO-latin qui comporte 256 caractères, les codes de 192 à 255 contenant la plupart des caractères spécifiques des langues indo-européennes (lettres accentuées).

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
	!	"	#	$	%	&		(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7

56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c	d	e	f	g

104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Scilab permet la conversion vers l'ASCII (et réciproquement) au moyen de la fonction *ascii*() ; cette opération est utile lorsqu'il s'agit de manipuler des textes externes à Scilab (par exemple, lus dans un fichier).

```
-->ascii('abc')
 ans  =  !  97.   98.   99. !
-->ascii([97 98 99])
 ans  = abc
```

3) Manipulation de caractères

Lecture de caractères

La fonction *readc_*() permet la lecture d'une chaîne de caractères (et donc d'un seul caractère). L'exécution du programme courant est suspendu jusqu'à ce qu'une chaîne de caractère (éventuellement vide) suivie de la touche entrée soit donnée par l'utilisateur.

```
-->c='o';
-->while c=='o' do
-->
-->
-->disp('Voulez-vous continuer ? (o/n) ')
-->c=readc_();
-->end
```

Exercices

Chiffres Romains

Il s'agit de créer la fonction *dec2rom*() permettant de donner l'écriture en chiffres romains d'un nombre entier.

Lettre	Valeur
M	1000
D	500
C	100
L	50
X	10
V	5
I	1

On n'oubliera pas que si 3 se note III, 4 se note IV, 400 CD, …

Réciproquement, créer la transformation inverse dans la fonction *rom2dec*().

Palindromes

Ecrire une fonction *palindrome*() reconnaissant si la chaîne de caractères qui lui a été passé comme paramètre est un palindrome, c'est-à-dire une phrase qui se lit de la même manière (à la ponctuation près), à l'endroit et à l'envers. (Vous pourrez supposer, dans un premier temps, que la chaîne a été transformée en une chaîne constituée des majuscules non accentuées des caractères de la chaîne initiale et que les espaces ainsi que la ponctuation de cette chaîne ont été supprimés).

Quelques palindromes sont bien connus :

- Esope reste ici et se repose.
- Elu par cette crapule.
- A man, a plan, a canal : Panama.
- Georges Pérec (1936 - 1982) est l'auteur d'un palindrome de 1 247 mots et plus de 76 000 caractères, qui débute ainsi :

Trace l'inégal palindrome. Neige. Bagatelle, dira Hercule. Le brut repentir, cet écrit né Pérec …

et se termine par :

… S'il porte, sépulcral, ce repentir, cet écrit ne perturbe le lucre : Haridelle, ta gabegie ne mord ni la plage, ni l'écart.

Généralités

D'une manière générale, le nombre $a_n a_{n-1}...a_2 a_1 a_0$, s'il est exprimé en base B, ne pourra comporter que des chiffres a_i compris entre 0 et B - 1. Ainsi, en base 10 nous disposons des 10 chiffres arabes de 0 à 9, en base 2 nous ne pourrons utiliser que les chiffres 0 et 1 ; en base 16 :0, 1, 2, ..., 9 et nous continuerons avec les lettres A, B, C, D, E, F qui auront les valeurs dix, onze, douze, treize, quatorze, quinze dans notre base usuelle.

Au nombre $a_n a_{n-1}...a_2 a_1 a_0$ sera associé la valeur $a_n*B^n + a_{n-1}*B^{n-1} + ... + a_2*B^2 + a_1*B + a_0$ calculée en base 10.

Expression en base 10 d'un nombre exprimé en base B

Comme nous sommes habitués à calculer en base 10, nous pouvons effectuer tout changement de bases en effectuant un passage par la base 10. La seule exception à cette règle est pour les passages entre les bases 2 et 16.

Les notations se feront à partir de la règle :

$$(a_n a_{n-1}...a_2 a_1 a_0)_{\mathbf{B}} = (a_n*B^n + a_{n-1}*B^{n-1} + ... + a_2*B^2 + a_1*B + a_0)_{\mathbf{10}}$$

Les calculs pourront être exprimés à l'aide de la méthode de Hörner (gage de vitesse et de précision) :

$$a_n*B^n + a_{n-1}*B^{n-1} + ... + a_2*B^2 + a_1*B + a_0 = ((((a_n*B + a_{n-1})*B + ...) + a_2)*B + a_1)*B + a_0$$

Expression en base B d'un nombre X exprimé en base 10

D'un nombre $X = ((((a_n*B + a_{n-1})*B + ...) + a_2)*B + a_1)*B + a_0$, nous déduisons simplement que lorsqu'on divise (division cartésienne ou entière) X par B, on obtient un reste égal à a_0 et un quotient entier égal à

$$Y = (((a_n*B + a_{n-1})*B + ...) + a_2)*B + a_1.$$

De même, en effectuant la division entière de Y par B, nous obtenons un reste égal à a_1 et un quotient égal à $Z = ((a_n*B + a_{n-1})*B + ...) + a_2$. En opérant les divisions successives, nous obtenons tous les chiffres du nombre (on arrête les divisions lorsque le quotient obtenu est nul).

Passage entre les bases 2 et 16

En base 16	En base 2	En base 10
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Ce tableau montre tout l'intérêt de la base 16. Comme exprimer les nombres en base 2 est particulièrement fastidieux et sujet à erreur, l'idée à été de regrouper par 4 et de convertir chaque groupe en 1 chiffre hexadécimal. Les chiffres de la base 16 permettent en effet de trouver le chiffre correspondant à chaque groupe de 4 chiffres binaires.

Le passage de la base 2 à la base 16 se fera donc par regroupement de 4 chiffres binaires consécutifs (en partant de la droite et en allant vers la gauche) et de la traduction en base 16 à l'aide du tableau décrit ci-contre.

Exemple : $(111011001)_{\mathbf{2}} = 0001\ 1101\ 1001 = (1D9)_{\mathbf{16}}$

Le passage de la base 16 à la base 2 se fera par traduction d'un chiffre hexadécimal en 4 chiffres binaires à l'aide du tableau ci-contre.

Exemple : $(3AF)_{\mathbf{16}} = 0011\ 1010\ 1111 = (1110101111)_{\mathbf{2}}$

Exercice :

Vous pourrez utiliser dans les fonctions à écrire la chaîne de caractères donnée en tout début du programme principal : lettres = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'. Cette chaîne a comme particularité que tous ses caractères peuvent être utilisés dans l'écriture d'un nombre en une base inférieure à 36 mais peuvent aussi être obtenus suivant leur indice de position dans la chaîne lettres. Ainsi, le nombre $(1A)_{\mathbf{16}}$ donné comme la chaîne de caractères st='1A' a comme valeur en base 10 : (strindex(lettres, part(st,1))-1)*16 + (strindex(lettres, part(st,2))-1).

1) Ecrire une fonction de_base_en_b10 qui reçoit en paramètres une chaîne de caractères qui est l'expression d'un nombre en une base quelconque et comme autre paramètre la base depuis laquelle le nombre doit être transformé et renvoie l'entier résultat.

2) Ecrire une fonction de_b10_en_base qui reçoit en paramètres un nombre en base 10 et comme autre paramètre la base dans laquelle ce nombre doit être transformé. Le résultat sera la chaîne de caractère donnant l'expression du nombre dans la base demandée.

3) Ecrire une fonction de_base1_en_base2 transformant un nombre d'une base 1 en une base 2.

Sur les enveloppes utilisées par la Poste, les codes postaux écrits en chiffres par l'envoyeur sont codés par des bandes colorées imprimées dans la partie inférieure.
Les codages des cinq chiffres du code postal sont écrits de droite à gauche, jointivement. La table de conversion est la suivante :

0: ..||||	1: .|.|||	2: .||.||	3: .|||.|	4: |..|||
5: |.|.||	6: |.||.|	7: ||..||	8: ||.|.|	9: |||..|

où le . représente un espace de code ASCII 32
et | est le caractère de code ASCII 124

On suppose que ce codage est contenu dans le tableau Tableau de taille 10 contenant les chaînes de la table de conversion.
Ainsi Tableau(3) est la chaîne '.|||.|'

Ecrire la fonction **[c]=codage_postal(n)** permettant d'écrire le codage c correspondant au code postal donné par *n*.

En vue d'un décodage, on doit s'assurer tout d'abord de la validité de la lecture du code (par un scanner ou lecture optique). Ecrire la fonction **[b]=Verification(c)** retournant la valeur vraie si le code est possible et la valeur faux sinon sachant que le codage de chaque chiffre doit comprendre 4 barres et nécessairement une barre en première position (dans son écriture qui est de droite à gauche ! *cf* table de conversion).

Ecrire la fonction **[n]=decodage_postal(c)** permettant, après s'être assuré de la fiabilité du codage, de déterminer le code postal.

Cryptographie

1ère partie

Jules César, pendant la guerre des Gaules, eu l'idée simple mais efficace, de transmettre des messages en décalant chacune des lettres de 3 crans dans l'alphabet. Ainsi, *a* devient *d*, *b* devient *e*, etc...
Depuis, tout système par décalage, quel que soit le nombre de crans choisi, est appelé "alphabet de Jules César".
1) Créer une fonction **[b]=codage_cesar(a,n)** avec un décalage donné par **n** et permettant de coder une phrase constituée de lettres en majuscules et de blancs. (une seconde version pourra transformer la chaîne en majuscules)
2) Créer une fonction de décodage permettant, par exemple, de lire la phrase, suite à un décalage de 7 crans :
"SL KLCVPY LZA ALYTPUL".

2ème partie

Il s'agit de construire deux fonctions, l'une servant à coder un texte, l'autre le décrypter.
Pour transmettre un message, on peut utiliser le système de cryptage à clé secrète suivant :

1ère étape : On supprime tout d'abord les caractères qui ne sont pas des lettres et on transforme les lettres restantes en leurs majuscules.
A chaque lettre du message en clair, on associe son rang dans l'alphabet (précédé de 0 si ce numéro est inférieur ou égal à 9) : A $\rightarrow$ 01, B $\rightarrow$ 02, ..., Z $\rightarrow$ 26
Exemple, pour le message Le contact aura lieu demain, il devient successivement LECONTACTAURALIEUDEMAIN, puis :
12 05 03 15 14 20 01 03 20 01 21 18 01 12 09 05 21 04 05 13 01 09 14

2ème étape : la clé secrète est un mot, par exemple NATUREL, que l'on transforme en chiffres comme dans la première étape. Ici, 14 01 20 21 18 05 12.

3ème étape : on aligne l'un en-dessous de l'autre le message en chiffres et la clé en chiffre (répétée autant de fois que nécessaire) :
12 05 03 15 14 20 01 03 20 01 21 18 01 12 09 05 21 04 05 13 01 09 14
14 01 20 21 18 05 12 14 01 20 21 18 05 12 14 01 20 21 18 05 12 14 01
On additionne en colonne les nombres écrits ; lorsque la somme dépasse 26, on diminue le résultat obtenu de 26.
Dans l'exemple, on obtient ainsi le message (secret) suivant :
01 06 23 10 06 25 13 17 21 21 16 10 06 24 23 06 15 25 23 18 13 23 15

Ecrire la fonction **[b]=codage_crypto(a,cle)** qui effectue la tache suivante.
On peut tout d'abord supposer que toutes les lettres des chaînes sont transformées en leurs majuscules et que tous les blancs ont été supprimés.
Vous devrez construire un premier tableau faisant correspondre à chaque lettre de la chaîne **a** son rang dans l'alphabet.
Vous devrez ensuite créer une chaîne **c** constituée de la chaîne **cle** répétée autant de fois que nécessaire pour atteindre la longueur de la chaîne **a** et coder la chaîne **c** dans un autre tableau comme pour la chaîne **a**.
Vous pourrez ensuite effectuer le codage grâce aux tableaux ainsi construits :
- construction de la somme des tableaux au décalage de 26 près
- constitution de la chaîne correspondant à ces rangs dans l'alphabet

Création du programme de décodage : **[b]=codage_crypto(a,cle)**
On suppose dans ce cas que l'on connaît le texte à décrypter a (il est constitué de la suite des chiffres) et la clé du codage (cle).
Utilisez et modifiez les éléments de codage pour construire cette fonction de décodage.

Exercices Caractères et Chaînes de caractères

I. Manipulation de caractères

1. Ecrire la fonction **[n]=Pos_Car(car,c)** renvoyant la position de la première occurrence du caractère car dans la chaîne c, ou 0 s'il n'y figure pas. (Vous n'utiliserez par la fonction strindex)
2. Ecrire la fonction **[n]=Nb_Occurr (car,c)** retournant le nombre d'occurrences du caractère car dans la chaîne c.
3. Ecrire la fonction **[n]=Nbmots(c)** retournant le nombre de mots de la phrase c. (On supposera que les mots sont séparés par un seul espace et que la chaîne ne contient aucun espace au début ou à la fin. Dans un deuxième temps, une version plus générale pourra bien sûr être écrite)

II.–Utilisation de fonctions et procédures

4. Ecrire la fonction **[d]=Inverse(c)** retournant la chaîne c inversée. ('abc' devient 'cba')
5. Ecrire la fonction **[d]=Sans_Blanc(c)** supprimant tous les espaces de la chaîne c.
6. Ecrire la fonction **[d]=Sch(c,p,n)** retournant la sous-chaîne formée de n caractères consécutifs de la chaîne c à partir du $p^{ième}$.
7. Ecrire la fonction **[d]=Mot(n,c)** retournant le $n^{ième}$ mot de la chaîne c, ou la chaîne vide si c contient moins de n mots.
8. On sait que le numéro INSEE d'identification des personnes, dit "numéro de sécurité sociale", est de la forme : SAAMMDDCCCNNN + CLE. (S=sexe; AMDC=année, mois, département, commune de naissance) La clé est le complément à 97 du reste de la division du numéro INSEE de 13 chiffres par 97.

a) Ecrire la fonction **[c]=Num_Insee** renvoyant une chaîne de 13 caractères *numériques exactement* saisie au clavier, avec S égal à 1 ou 2, MM entre 01 et 12.

b) Ecrire la fonction **[m]=Cle(n)** retournant la clé du numéro INSEE n.

c) Ecrire la fonction **Message(n)** affichant, selon le numéro INSEE n, un message de la forme: "Bonjour, *Madame*! Vous êtes née en *Janvier 1959*. La clé de votre numéro INSEE est égale à *45*"

9. Un mot est un *mot carré* s'il est du type *uu* où *u* est une chaîne de caractères (sans espace).
exemples : *papa* ainsi que *boubou*

a) Ecrire une fonction **[b]=carre(m)** qui vérifie si une chaîne *m* est une chaîne carrée.
(on suppose que la chaîne *m* ne contient aucun espace).

b) Ecrire une fonction **[m]=generer_carre**; permettant de générer aléatoirement un mot carré de 8 lettres. (rappel ascii('A')=65).

c) Ecrire une fonction permettant de déterminer et d'afficher tous les mots carrés d'une chaîne de caractères *m*.

d) Ecrire un programme principal permettant de choisir d'exécuter l'une ou l'autre des procédures précédentes.