TD Informatique Algobox - Crypter une phrase avec le code César

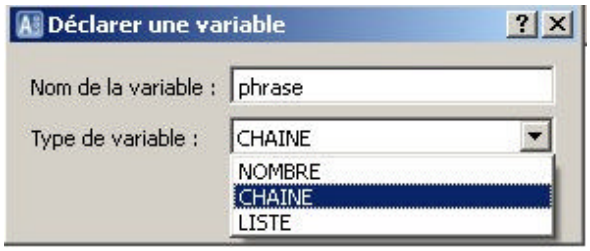
I – Utiliser la variable « chaine »

Dans algobox, il est possible de travailler avec des mots ou des phrases comme variable : une phrase est appelée « chaine » en référence à la chaine de caractères qu'elle représente.

💻 Ecrire un algorithme qui demande d'entrer une phrase et renvoie le nombre de caractères présent dans la phrase

Indications :

- Déclarez une variable « CHAINE » appelée « phrase »



- Déclarez une variable « NOMBRE » appelée « longueur_phrase »
- Le nombre de caractères dans une variable « CHAINE » nommée « phrase » est donnée par la fonction « phrase.length »

Correction :

```
1     VARIABLES
2       phrase EST_DU_TYPE CHAINE
3       longueur_phrase EST_DU_TYPE NOMBRE
4     DEBUT_ALGORITHME
5       AFFICHER "Entrez une phrase"
6       LIRE phrase
7       longueur_phrase PREND_LA_VALEUR phrase.length
8       AFFICHER "Le nombre de caractère dans votre phrase est : "
9       AFFICHER longueur_phrase
10    FIN_ALGORITHME
```

II – Action sur chaque caractère d'une chaine

II-a) Indexation de chaque caractère

Toute variable du type « CHAINE » est considérée comme un tableau a une seule ligne et autant de colonnes que de caractères présents dans la chaine. Le premier caractère est repéré par l'indice 0.

Exemple :la variable CHAINE : « le mot »

0	1	2	3	4	5
l	e		m	o	t

Il est possible d'extraire le contenu d'une chaine avec l'instruction **chaine.substr(position_premier_caractère_à_extraire,nombre_de_caractères_à_extraire).**

Attention : le premier caractère a pour position 0 (et pas 1)

***Exemple** : b prend la valeur a.substr(4,2) (b sera alors formé des 5ème et 6ème caractères de a ; a et b étant des variables du type CHAINE)*

💻 Ecrire un algorithme renvoyant la première lettre d'une phrase ou mot entré au clavier

Indications :

- Variables à déclarer : phrase est du type « CHAINE » ; lettre est du type « CHAINE »
- La première lettre de « phrase » sera donnée par l'instruction : phrase.substr(0,1)

Correction :

```
1     VARIABLES
2       phrase EST_DU_TYPE CHAINE
3       lettre EST_DU_TYPE CHAINE
4     DEBUT_ALGORITHME
5       LIRE phrase
6       lettre PREND_LA_VALEUR phrase.substr(0,1)
7       AFFICHER "La première lettre de votre phrase est : "
8       AFFICHER lettre
9     FIN_ALGORITHME
```

II-b) Concaténage de deux chaines

Il est possible d'ajouter (concaténer) des chaines

***Exemple** : MOT1 , MOT2 et MOT_NOUVEAU sont des variables du type « CHAINE »*
Si MOT_NOUVEAU prend la valeur MOT1+MOT2 alors l'affichage de MOT_NOUVEAU donnera la chaine « MOT1MOT2 »

💻 Ecrire un algorithme qui ajoute deux mots l'un à la suite de l'autre

Indications :

- Variables à déclarer : mot1, mot2 et mot_nouveau sont du type « CHAINE »

Correction :

```
1     VARIABLES
2       mot1 EST_DU_TYPE CHAINE
3       mot2 EST_DU_TYPE CHAINE
4       mot_nouveau EST_DU_TYPE CHAINE
5     DEBUT_ALGORITHME
6       LIRE mot1
7       LIRE mot2
8       mot_nouveau PREND_LA_VALEUR mot1+mot2
9       AFFICHER "Les deux mots concaténés donnent le mot "
10      AFFICHER mot_nouveau
11    FIN_ALGORITHME
```

II-c) Utiliser une boucle « POUR …DE… A… » pour parcourir une chaine .

Il s'agit du procédé itératif le plus courant en algorithmique, on demande de faire une action pour i allant de telle valeur initiale à telle valeur finale …

💻 **Ecrire un algorithme faisant écrire à l'écran tous les entiers de m à n**

Indications :

Un nombre peut-être transformé en chaine avec l'instruction nombre.toString()
***Exemple** : si i est du type « NOMBRE », alors i.toString() est une chaine représentant le caractère i.*

- Variables à déclarer : i, m et n sont des variables du type « NOMBRE » ; suite est une variable du type « CHAINE »
- La boucle à utiliser : POUR i allant de m à n, faire afficher la variable « suite » qui prend la valeur i.toString()

Correction :

```
1     VARIABLES
2       i EST_DU_TYPE NOMBRE
3       suite EST_DU_TYPE CHAINE
4       n EST_DU_TYPE NOMBRE
5       m EST_DU_TYPE NOMBRE
6     DEBUT_ALGORITHME
7       AFFICHER "Entrer l'entier de départ"
8       LIRE m
9       AFFICHER "Entrer le dernier entier"
10      LIRE n
11      POUR i ALLANT_DE m A n
12        DEBUT_POUR
13        suite PREND_LA_VALEUR i.toString()
14        AFFICHER suite
15        FIN_POUR
16    FIN_ALGORITHME
```

💻 **Etudier l'algorithme suivant et indiquez son action (vérifiez en testant) :**

```
1     VARIABLES
2       x EST_DU_TYPE NOMBRE
3       phrase EST_DU_TYPE CHAINE
4       cryptage EST_DU_TYPE CHAINE
5       longueur EST_DU_TYPE NOMBRE
6     DEBUT_ALGORITHME
7       LIRE phrase
8       longueur PREND_LA_VALEUR phrase.length
9       POUR x ALLANT_DE 0 A longueur-1
10        DEBUT_POUR
11        cryptage PREND_LA_VALEUR cryptage+phrase.substr(longueur-x-1,1)
12        FIN_POUR
13      AFFICHER cryptage
14    FIN_ALGORITHME
```

Réponse :

III – coder une lettre avec son code ASCII.

CODE	LETTRE	CODE	LETTRE
065	A	097	a
066	B	098	b
067	C	099	c
068	D	100	d
069	E	101	e
070	F	102	f
071	G	103	g
072	H	104	h
073	I	105	i
074	J	106	j
075	K	107	k
076	L	108	l
077	M	109	m
078	N	110	n
079	O	111	o
080	P	112	p
081	Q	113	q
082	R	114	r
083	S	115	s
084	T	116	t
085	U	117	u
086	V	118	v
087	W	119	w
088	X	120	x
089	Y	121	y
090	Z	122	z

Chaque lettre de l'alphabet, mais aussi les autres caractères comme les points, virgules, espaces etc sont traduit en langage machine par un nombre repéré dans la table ASCII. Ci-contre, une version simplifiée de cette table qui donne les codes des lettres de A jusqu'à Z , puis de a jusqu'à z.

💻 **Ecrire un algorithme donnant le code ASCII d'une lettre entrée au clavier.**

L'instruction machaine.charCodeAt(pos) permet d'obtenir le nombre égal au code ascii de la lettre figurant à la position pos dans la chaine machaine (Attention : le premier caractère a pour position 0).

Correction :

```
1     VARIABLES
2       lettre EST_DU_TYPE CHAINE
3       code_ASCII EST_DU_TYPE NOMBRE
4     DEBUT_ALGORITHME
5       LIRE lettre
6       //Cet algorithme détermine le code ascii d'une lettre
7       code_ASCII PREND_LA_VALEUR lettre.charCodeAt(0)
8       AFFICHER code_ASCII
9     FIN_ALGORITHME
```

💻 **Ecrire un algorithme donnant le caractère correspondant à un code ASCII entré au clavier**

l'instruction String.fromCharCode(nombre) renvoie une chaine contenant le caractère dont le code ascii est égal à nombre.

Correction :

```
1     VARIABLES
2       code_Ascii EST_DU_TYPE NOMBRE
3       lettrecorrespondante EST_DU_TYPE CHAINE
4     DEBUT_ALGORITHME
5       LIRE code_Ascii
6       lettrecorrespondante PREND_LA_VALEUR String.fromCharCode(code_Ascii)
7       AFFICHER "La lettre correspondante au code Ascii "
8       AFFICHER code_Ascii
9       AFFICHER " est la lettre ou le symbole : "
10      AFFICHER lettrecorrespondante
11    FIN_ALGORITHME
```

IV – Codage par la méthode de CESAR

Dans cette première partie, pour simplifier, nous ne crypterons par décalage qu'un mot écrit en lettre majuscule, c'est-à-dire avec des lettres dont le code ASCII est compris entre 65 et 090

Nous savons désormais transformer une lettre en son caractère ASCII dans une variable « CHAINE » : le principe est d'opérer le décalage sur le nombre correspondant au code ASCII, puis de revenir ensuite à une lettre …

Un outil important : l'utilisation des nombres modulo n

Eh oui, pour aller plus loin, il nous faut faire un peu de mathématiques ! quand même …

Le problème : les nombres que nous allons utiliser sont donc compris entre 65 et 90 (la plage des codes ASCII des lettres majuscules de l'alphabet), nous commencerons par simplifier les calculs en retranchant, le temps de l'algorithme, 65 à chaque nombre, de façon à travailler avec des nombres allant de 0 (pour A) à 25 (pour Z) : nous appellerons cette nouvelle table le « code ajusté »

Le principe du code Cesar, est d'opérer un décalage de n rangs pour chaque lettre du mot : il suffit d'ajouter le nombre n au code ASCII correspondant à la lettre ! C'est là qu'est l'os hélas …

En effet, si n, le décalage est par exemple 12, et si on souhaite crypter la lettre B dont le code ajusté est 1 : on fait 1 +12 = 13 qui est le code ajusté de N (78 – 65 = 13).

Mais si on souhaite crypter la lettre P dont le code ajusté est 15 on fait : 15 + 12 = 27 … on sort de la plage 0 … 25 ! il faut donc trouver une astuce pour « rester dans {0,1,2, …, 25}

Congruence modulo 26

Proposons la définition suivante : ***trouver la congruence d'un nombre a modulo 26, c'est remplacer ce nombre a par son reste r dans la division euclidienne de a par 26.***

Vocabulaire : a = r [26] se lit « a égale r modulo 26 » ou encore « a est congru à r modulo 26 »

Exemple 1 : prenons a = 23, on a 23 = 0x26 + 23 donc 23 = 23 [26]

Exemple 2 : prenons a = 35, on a cette fois 35 = 1x26 + 9 et donc 35 = 9 [26]

Considérons l'ensemble fini {0, 1, 2, 3, …,24, 25} , (NB : cet ensemble de 26 éléments se note Z/26Z et est ce que l'on appelle un anneau d'entiers), cet ensemble a la particularité d'être « stable » pour l'addition et la multiplication pour des résultats modulo 26, la stabilité indiquant que le résultat obtenu ne « sort pas » de l'ensemble {0, 1, 2, 3, …,24, 25}

Par exemple 12 + 20 = 32 dans l'addition traditionnelle : on sort de {0, 1, 2, 3, …,24, 25}, mais si on prend le résultat congru à 26, on obtient 12 + 20 = 6 [26] . En effet 32 = 1x26 + 6 donc 6 est bien le reste de la division de 32 par 26.

Ainsi dans notre ensemble {0, 1, 2, 3, …,24, 25} , si on parle en « modulo 26 » on a 12 + 20 = 6

De même : 17 + 23 = 14 et par exemple 3 x 18 = 2 … tous les résultats obtenus « restent » dans {0, 1, 2, 3, …,24, 25} , c'est le principe de la stabilité.

Dans notre algorithme, nous traduirons les résultats de nos calculs de décalage en résultats modulo 26, ce qui se fait par l'instruction « nombre%26 »

💻 **Etudier l'algorithme suivant et indiquez son action (vérifiez en testant) :**

```
1     VARIABLES
2       decalage EST_DU_TYPE NOMBRE
3       lettre EST_DU_TYPE CHAINE
4       lettre_cryptee EST_DU_TYPE CHAINE
5       n EST_DU_TYPE NOMBRE
6     DEBUT_ALGORITHME
7       AFFICHER "Entrez une lettre majuscule"
8       LIRE lettre
9       LIRE decalage
10      n PREND_LA_VALEUR (lettre.charCodeAt(0)-65+decalage)%26
11      lettre_cryptee PREND_LA_VALEUR String.fromCharCode(n+65)
12      AFFICHER "La lettre cryptée est : "
13      AFFICHER lettre_cryptee
14    FIN_ALGORITHME
```

💻 **EX1 - Transformez l'algorithme précédent pour crypter un mot entier écrit en majuscules**

💻 **Ex2 – Ecrire enfin un algorithme cryptant une phrase entière utilisant des majuscules ou des minuscules, des espaces ou des symboles. On effectuera le décalage de César uniquement sur les lettres.**

Correction EX 1 :

```
1     VARIABLES
2       decalage EST_DU_TYPE NOMBRE
3       n EST_DU_TYPE NOMBRE
4       i EST_DU_TYPE NOMBRE
5       longueur EST_DU_TYPE NOMBRE
6       mot EST_DU_TYPE CHAINE
7       mot_crypte EST_DU_TYPE CHAINE
8     DEBUT_ALGORITHME
9       AFFICHER "Entrez un mot en lettres majuscules"
10      LIRE mot
11      AFFICHER "Entrez le decalage"
12      LIRE decalage
13      longueur PREND_LA_VALEUR mot.length
14      POUR i ALLANT_DE 0 A longueur-1
15        DEBUT_POUR
16        n PREND_LA_VALEUR (mot.charCodeAt(i)-65+decalage)%26
17        mot_crypte PREND_LA_VALEUR mot_crypte+String.fromCharCode(n+65)
18        FIN_POUR
19      AFFICHER "Le mot crypté est "
20      AFFICHER mot_crypte
21    FIN_ALGORITHME
```

Correction EX 2 :

```
1     VARIABLES
2       decalage EST_DU_TYPE NOMBRE
3       code_ASCII EST_DU_TYPE NOMBRE
4       n EST_DU_TYPE NOMBRE
5       lettre_cryptee EST_DU_TYPE CHAINE
6       phrase EST_DU_TYPE CHAINE
7       longueur_phrase EST_DU_TYPE NOMBRE
8       i EST_DU_TYPE NOMBRE
9     DEBUT_ALGORITHME
10      //Cet algorithme crypte un message par un décalage monoalphabétique de
longueur n (Code César)
11      AFFICHER "Entrez une phrase ne comportant que des minuscules et des
majuscules sans accent, les espaces sont autorisés"
12      LIRE phrase
13      longueur_phrase PREND_LA_VALEUR phrase.length
14      decalage PREND_LA_VALEUR -1
15      AFFICHER "Entrez le décalage : nombre entier compris entre 0 et 25"
16      //Ici, on va s'assurer que le décalage entré au clavier est bien compris
entre 1 et 25
17      TANT_QUE (decalage <0 OU decalage>25) FAIRE
18        DEBUT_TANT_QUE
19        LIRE decalage
20        FIN_TANT_QUE
21      POUR i ALLANT_DE 0 A longueur_phrase-1
22        DEBUT_POUR
23        code_ASCII PREND_LA_VALEUR phrase.charCodeAt(i)
24        //à partir d'ici, on teste si le caractère de la phrase est une minuscule,
une majuscule ou un autre caractère
25        SI (code_ASCII>=65 ET code_ASCII<=90) ALORS
26          DEBUT_SI
27          n PREND_LA_VALEUR code_ASCII-65
28          n PREND_LA_VALEUR (n+decalage)%26
29          lettre_cryptee PREND_LA_VALEUR lettre_cryptee+String.fromCharCode(n+65)
30          FIN_SI
31          SINON
32            DEBUT_SINON
33            SI (code_ASCII>=97 ET code_ASCII<=122) ALORS
34              DEBUT_SI
35              n PREND_LA_VALEUR code_ASCII-97
36              n PREND_LA_VALEUR (n+decalage)%26
37              lettre_cryptee PREND_LA_VALEUR
lettre_cryptee+String.fromCharCode(n+97)
38              FIN_SI
39              SINON
40                DEBUT_SINON
41                lettre_cryptee PREND_LA_VALEUR
lettre_cryptee+String.fromCharCode(code_ASCII)
42                FIN_SINON
43            FIN_SINON
44        FIN_POUR
45      AFFICHER "La phrase cryptée est:"
46      AFFICHER lettre_cryptee
47    FIN_ALGORITHME
```