TP N°4
Cryptographie

Objectifs :

- ✓ Manipulation de caractère et code ASCII
- ✓ Allocation de mémoire statique et dynamique
- ✓ Tableaux

Préambule

Après avoir ouvert virtualBox et vous être logué sous l'environnement CentOS5, on vous demande de créer un répertoire **`tp4`** dans votre répertoire de travail **`algo.`** Vous vous placerez ensuite dans le répertoire **`tp4`** et créerez un fichier nommé `tp4.c`. Vous resterez dans ce même fichier pour tout le TP.

Le chiffrement par décalage est historiquement une des premières méthodes que l'homme à utiliser pour crypter un message. Basé sur un principe très simple, ce procédé de chiffrement était utilisé par Jules César dans ses correspondances secrètes, d'où l'appellation courante « **chiffrement de César** ».

Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet.

La longueur du décalage, 3 dans l'exemple évoqué, constitue **la clé du chiffrement** qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles (y compris la clé nulle, qui ne modifie pas le texte).

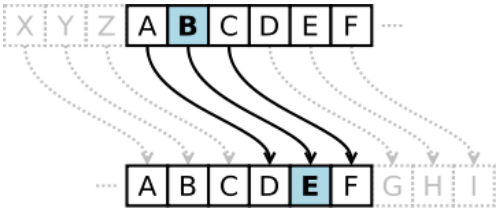


Figure 1: Exemple de chiffrement de César avec un décalage (ou clé) de 3

1. Chiffrement

On vous demande de créer une fonction `char chiffrement(char c, int cle)` qui prend en paramètre d'entrée un caractère à chiffrer ainsi qu'une clé de chiffrement et qui retourne le caractère chiffré en utilisant le principe du chiffre de César. Attention on ne chiffrera que les lettres majuscules, les autres caractères ne seront pas chiffrés (la fonction retourne le caractère à l'identique). Vous sauvegarderez votre fonction dans le fichier `tp4.c` et vous la testerez dans une fonction `main` en utilisant les exemples suivants :

```
VERCINGETORIX  cle = 3
YHUFLQJHWRULA

EN L'OCCURRENCE L'IMBECILLITE EST UN DILEMNE ETYMOLOGIQUE ! cle = 20
YH F'IWWOLLYHWY F'CGVYWCFFCNY YMN OH XCFYGGY YNSGIFIACKOY !
```

2. Déchiffrement

On vous demande à présent de créer une fonction `char dechiffrement(char c, int cle)` qui prend en paramètre d'entrée un caractère chiffré ainsi qu'une clé de chiffrement et qui retourne le caractère déchiffré en utilisant le principe du chiffre de César. Attention on ne déchiffrera que les lettres majuscules, les autres caractères ne seront pas déchiffrés (la fonction retourne le caractère à l'identique Vous sauvegarderez votre fonction dans le fichier `tp4.c` et vous la testerez dans une fonction `main` en utilisant les exemples précédents.

3. Cryptanalyse du chiffre de César

Le chiffrement par décalage est malheureusement très simple à être cassé. Une méthode classique consiste à analyser la fréquence de chacune des lettres du message chiffré et établir une correspondance avec la langue à laquelle on soupçonne le texte clair d'appartenir.

Pour chaque lettre q de l'alphabet, on détermine la fréquence d'apparition de cette lettre au sein du message chiffré en ne considérant que les lettres utiles du message, à savoir les lettres majuscules.

En français, la fréquence d'apparition des lettres est connue et est illustrée en Figure 2 :

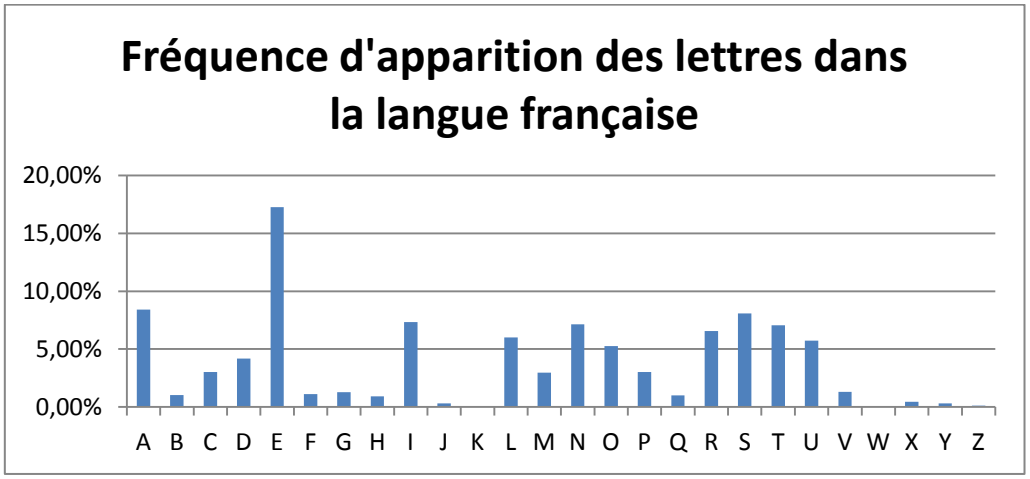


Figure 2: Fréquence moyenne d'apparition des lettres dans la langue française

En cherchant par exemple, la lettre la plus fréquente dans notre texte chiffré, on peut découvrir de combien a été décalée la lettre E, et par conséquent de connaitre la clé du chiffrement.

On vous demande de créer la fonction suivante :

`float* analyseFreq(char* texte,int N)` : calcule la fréquence d'apparition des lettres de l'alphabet dans un texte de longueur N. Le résultat est stocké dans un tableau de flottant de 26 cases, où chaque case du tableau correspond à la fréquence d'apparition (en %) d'une lettre (indice 0 = lettre A, indice 1 = lettre2,…, indice 25 = lettre Z). Attention ne sont considérées que les lettres majuscules de l'alphabet (A,B,… Z), les autres caractères sont ignorés..

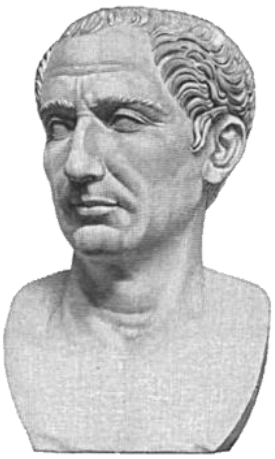
On vous demande ensuite de créer la fonction suivante :

`int calculCle(float* tabFreq)` : prend en entrée le tableau de fréquence d'apparition des lettres calculé précédemment et calcule la valeur du décalage utilisé pour chiffrer le texte correspondant

Dans le programme principal, on vous demande enfin de tester les fonctions précédentes avec le texte chiffré suivant :

```
ZNVGER PBEORNH, FHE HA NEOER CREPUR, GRANVG RA FBA ORP HA SEBZNTR. ZNVGER
ERANEQ CNE Y'BQRHE NYYRPUR, YHV GVAG N CRH CERF PR YNATNTR : RG OBAWBHE
ZBAFVRHE QH PBEORNH. DHR IBHF RGRF WBYV! DHR IBHF ZR FRZOYRM ORNH! FNAF
ZRAGVE, FV IBGER ENZNTR FR ENCCBEGR N IBGER CYHZNTR IBHF RGRF YR CURAVK QRF
UBGRF QR PRF OBVF. N PRF ZBGF YR PBEORNH AR FR FRAG CNF QR WBVR; RG CBHE
ZBAGERE FN ORYYR IBVK, VY BHIER HA YNETR ORP YNVFFR GBZORE FN CEBVR. YR
ERANEQ F'RA FNVFVG RG QVG : ZBA OBA ZBAFVRHE, NCERARM DHR GBHG SYNGGRHE IVG
NH QRCRAF QR PRYHV DHV Y'RPBHGR : PRGGR YRPBA INHG OVRA HA SEBZNTR FNAF
QBHGR. YR PBEORNH UBAGRHK RG PBASHF WHEN ZNVF HA CRH CYHF GNEQ, DH'BA AR
Y'L CERAQENVG CYHF.
```

Quelle est la clé de chiffrement et quel est le texte en clair ?



4. Annexe : rappels sur les chaîne de caractères

Une chaîne de caractères (appelée string en anglais) est une suite de caractères, c'est-à-dire un ensemble de symboles faisant partie du jeu de caractères, défini par le code ASCII (American Standard Code for Information Interchange) que vous trouverez en Figure 3. Il s'agit de la norme de codage de caractères en informatique la plus connue, la plus ancienne et la plus largement compatible.

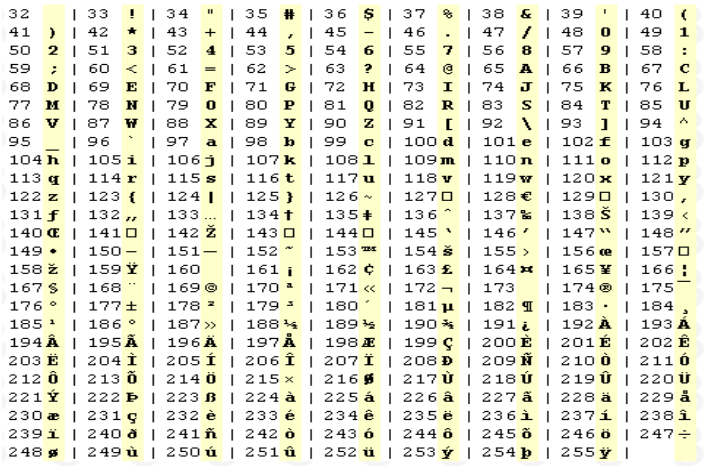


Figure 3 : Table ASCII 8 bits

En langage C, une chaîne de caractères est un tableau, comportant plusieurs données de type char, dont le dernier élément est le caractère nul '\0', c'est-à-dire le premier caractère du code ASCII (dont la valeur est 0). Ce caractère est un caractère de contrôle (donc non affichable) qui permet d'indiquer une fin de chaîne de caractères. Ainsi une chaîne composée de n éléments sera en fait un tableau de n+1 éléments de type char.

La chaine de caractère « Hello » sera donc représentée par un pointeur `maChaine_pc` de la façon suivante :

Adresse	Valeur (Caractère)	Valeur (code ASCI décimal)
maChaine_pc	H	72
maChaine_pc+1	e	101
maChaine_pc+2	l	108
maChaine_pc+3	l	108
maChaine_pc+4	o	111
maChaine_pc+5	\0	0

Tableau 1 : Chaine de caractères en Langage C

`\0` : deux caractères tapés, \ suivi de 0, ne représente en langage C qu'un seul caractère. \est un caractère spécial permettant d'introduire des caractères spéciaux. Vous connaissez déjà `\n` => saut de ligne, `\t` => introduction d'une tabulation. Si vous avez besoin d'un caractère ASCII de code octal N, vous pouvez taper `\N`.

On peut déclarer et initialiser directement une chaine de caractères de la façon suivante (l'allocation mémoire est faite en même temps) :

```
char*  maChaine_pc = "Hello";
```

ou bien on peut déclarer la chaine (et allouer la mémoire nécessaire) puis initialiser les caractères uns à uns de la façon suivante :

```
char maChaine_pc[6];
maChaine_pc[0] = 'H';
maChaine_pc[1] = 'e';
maChaine_pc[2] = 'l';
maChaine_pc[3] = 'l';
maChaine_pc[4] = 'o';
maChaine_pc[5] = '\0'; // signifie fin de chaine
```

L'opération `maChaine_pc[i]`, ou bien `*( maChaine_pc +i)` permet d'accéder au i-ème caractère en partant de 0. Pour que `maChaine_pc` soit une chaine de caractères, il suffit de trouver la symbole `\0` parmi les N caractères qu'il contient. La chaîne s'arrête à `\0`.

On peut également allouer la mémoire de la chaine façon dynamique avec la fonction `malloc()`.

```
char* maChaine_pc = malloc(6*sizeof(char));
/* … */
free(maChaine_pc) ;
```

Pour écrire une phrase à l'écran on utilisera la commande suivante :

```
printf("%s", maChaine_pc )
```

```
>> Hello
```

Où `maChaine_pc` est une variable de type pointeur sur caractère `( char * )` et `%s` est le format de chaîne de caractère.

Pour lire une phrase à l'écran on pourra utiliser la commande suivante :

```
scanf("%s", maChaine_pc )
```

Où `maChaine_pc` est encore une variable de type pointeur sur caractère `( char * )`. A la fin `scanf` ajoute automatiquement le caractère spécial `\0`.

Attention, `scanf` s'arrête dès qu'il y a un espace ou une tabulation. Il faudra donc utiliser `gets` pour récupérer aussi les espaces.

Pour copier une chaine de caractère dans une autre on peut utiliser la fonction `strcpy()` :

```
char maChaine2_pc[6];
strcpy(maChaine2_pc, maChaine_pc);
printf("%s", maChaine2_pc) ;
```

```
>> Hello
```