IUT TOULON VAR

Département Génie Electrique et Informatique Industrielle

Vous devez traiter au moins un problème au choix parmi les quatre sujets proposés.

Vous devez rendre le fichier source commenté de votre programme et en faire une démonstration

S1

Algorithmique et langage C
Sujets des Travaux pratiques
Série n°3

Nom : **Prénom:**

Groupe:

Algorithmique et langage C

Sujets série n°3

1. Un compresseur de fichier rudimentaire (***)

Le but de cet exercice est de programmer un compresseur de fichier basé sur la méthode RLE (Run Length Encoding). Cette méthode consiste à remplacer dans un fichier des suites de caractères identiques par 3 caractères. Un caractère ! qui indique le début d'une séquence de remplacement, un caractère chiffre (compris entre 4 et 9) qui indique le nombre de caractères remplacés suivi du caractère effectivement remplacé. Des suites de caractères inférieures à 4 caractères ne sont pas modifiés car le gain serait nul. Un caractère ! dans le fichier original est traduit par deux !! dans le fichier compressé.
Par exemple avec le fichier ci-dessous, le fichier compressé obtenu est :

!bonjour!!	!!bonjour!!!!
aaaazzzzzzz	!4a!7z
jkiiiiii	jk!6i
ss	ss
sssssssssssdfffff	!9sssd!5f
1111111111111111	!91!71
xxxxxggg!gnnnnnnn!	!5xggg!!g!7n!!
qqqxx	qqqxx
lm!	lm!!

Des séquences de plus de 9 caractères identiques consécutifs sont traduites par deux (ou plus) séquences compressées. Ainsi la séquence de 11 s est traduite par !9sss (9 fois s + ss) et la séquence de 16 1 est traduite par !91!71 (9 fois 1 + 7 fois 1).

1/ Ecrire un programme qui compare deux fichiers dont les chemins sont entrés au clavier ☼.
2/ Ecrire un programme qui compresse par la méthode décrite un fichier dont le chemin est entré au clavier. Le nom du fichier compressé est le nom du fichier de départ auquel on rajoute l'extension rle. Votre programme indiquera le gain obtenu ☼☼☼.
3/ Ecrire le programme de décompression correspondant. Vérifier son fonctionnement avec votre programme de comparaison ☼☼.
Essayez vos programme avec différents format de fichier (txt,pdf,exe,bmp,jpeg,etc...) et notez les performances obtenues.
4/ Le gain maximal, de 6 caractère, est obtenu pour une séquence de 9 caractères identiques consécutifs. Les caractères ! présents dans le fichier d'origine pénalisent l'algorithme car il doivent être doublés. On peut facilement améliorer l'algorithme de deux manières :

- en choisissant un caractère qui n'apparaît pas dans le fichier d'origine (ou le caractère le moins fréquent lorsque toutes les valeurs existent) pour indiquer le début d'une séquence de remplacement. Le caractère choisit est simplement indiqué en premier caractère du fichier compressé.
- -en choisissant un caractère supplémentaire pour des séquences de 10 à 99 caractères identiques consécutifs. Ce caractères est également choisit parmis les caractères peu fréquents du fichier d'origine et est indiqué en deuxième caractère du fichier compressé.

Améliorez vos programmes et notez le gain obtenu pour différents formats de fichier☼☼.

2. Un "générateur" de (moins mauvais) mot de passe (**)

Les meilleurs mots de passe sont ceux générés aléatoirement. Mais ce sont aussi les plus difficiles à retenir. Un mot du langage courant est facile à retenir mais ne résistera pas longtemps à une attaque par dictionnaire. Un moyen d'obtenir un mot de passe facile à retenir et compliquant (un peu) l'attaque par dictionnaire est d'utiliser un mot du langage courant et de le couper à une position aléatoire par un caractère aléatoire imprimable et non alphabétique*. Par exemple, à partir du mot "bonjour" entré par l'utilisateur on propose le mot "bonjo!ur".
On pourra ensuite proposer un mot passe contenant deux caractères non alphabétiques. Toujours à partir de "bonjour", on obtiendrait "bonj#o'ur" ou "6bonjo&ur". (les positions des deux caractères non alphabétiques ne devant pas être consécutives)

*On définit un caractère alphabétique comme un caractère appartenant à ['a','z']u['A','Z'].
*Les caractères dont le code est inférieur 0x20 ne sont pas "imprimable".
Ce programme sera réalisé à l'aide de fonctions.
Chaque fonction sera testée indépendamment avec une fonction main qui l'appelle. Ensuite, toutes les fonctions seront utilisées pour le programme final.

1/ Ecrire une fonction qui retourne un nombre aléatoire compris entre une valeur minimum nmin et une valeur maximum nmax fournie en paramètre.
Le prototype sera : `int Alea(int nmin, int nmax);` ☼

2/ Ecrire une fonction qui retourne un caractère ascii aléatoire imprimable et non alphabétique. Chaque caractère non alphabétique devra avoir la même probabilité d'apparition.
Le prototype sera : `char CarAlea(void);` ☼☼

3/ Ecrire une fonction qui insére à la position p dans une chaîne ch, un caractère x.
Le prototype sera : `void Insere(char * ch , char x, int p);` ☼☼

4/ Ecrire le programme complet en utilisant les fonctions précédentes ☼.

3. Jeux : Retrouver des mots mélangés (**)

A partir d'un dictionnaire sous forme de fichier texte contenant un mot par ligne, on vous demande de présenter à l'utilisateur un mot choisit aléatoirement dont les lettres ont été mélangées aléatoirement. L'utilisateur aura un certain nombre d'essais pour deviner le mot original.
Le nombre d'essais pourra être fonction de la longueur du mot à deviner.

Exemple :

dictionnaire.txt :
.....
.....
voile
voilure
voiture
voiturette
.....
.....

Exécution :
Devinez ce mot : ervouti
Vous avez 5 essais :
essai 1 : *vroum*
Non !
Essai2 : *etrouvi*
Non !
Essai3 : *voiture*
Bravo !
On rejoue (o/n) : *N*
Au revoir ....

Un dictionnaire est disponible à : http://arlotto.univ-tln.fr/cours_de_c/tpserie3/dico.txt

Imaginer un système de scores basé sur le temps mis pour répondre.
Un fichier des meilleurs scores sera créé et tenu à jour par votre programme.

4. Crypter , Décrytper (*) et Casser un code simple (***).

Un code très simple mais néanmoins assez utilisé lorsque la sécurité est peu critique et le codage par OU exclusif avec une clé (mot de passe) constituée d'une chaîne de caractères.
On utilise la propriété de réversibilité d'un OU exclusif :

Si code = clair ⊕ clé alors clair = code ⊕ clé (en C a⊕b est noté a^b)

La même opération est alors utilisée pour coder et décoder un message.

Par exemple, pour coder le message "May the force be with you" avec la clé "Yoda", on code chaque caractère du message avec un caractère de la clé en recommençant avec le premier caractère de la clé lorsque on atteint le dernier :

```
YodaYodaYodaYodaYodaYodaY
May the force be with you
```

Le message codé sera : 'Y'⊕'M' , 'o'⊕'a', 'd'⊕'y', 'a'⊕' ', 'Y'⊕'t', 'o'⊕'h','d'⊕'e','a'⊕' ',.....
Bien sûr, le message codée n'est pas constitué uniquement de caractères ascii imprimables, mais ce n'est pas un problème car il n'est destiné à être lu tel quel. Cette méthode s'applique à n'importe quel type de fichier. Le décodage se fait en appliquant le même principe.

Ecrire un programme qui crypte un fichier par cette méthode. Votre programme demandera un mot de passe de 8 à 16 caractères.

Ecrire un programme qui, après avoir demandé le mot de passe, décrypte un fichier crée par le programme précédent.

On va montrer qu'il est assez facile si l'on dispose d'un fichier assez volumineux de retrouver le mot de passe par une méthode statistique en supposant connue la longueur de la clé et en supposant que le message original contient en majorité du texte.
En effet, la faiblesse de l'algorithme est que pour une clé de longueur N, un octet sur N est toujours codé avec le même caractère de la clé.
Ainsi avec l'exemple précédent,

le caractère 'Y' code les octets de rang 0,4,8,12,....
le caractère 'o' code les octets de rang 1,5,9,13,....

Le caractère de rang i de la clé code les octets de rang i, i+N, i+2N,....

Pour les mots de la langue française on sait que le caractère le plus fréquent est le e. Mais en fait lorsqu'on écrit un texte c'est l'espace qui est le caractère le plus fréquent.
Il suffit donc de rechercher le caractère le plus fréquent parmi les caractère de rang 0,N,2N,.... de fichier crypté et d'en faire un ou exclusif avec le code ascii de l'espace pour retrouver le premier caractère de la clé. En procédant de même pour les caractères de rang 1,1+N,1+2N,...., on trouve le deuxième caractère de la clé. Et ainsi de suite jusqu'à dernier caractère de la clé. Ensuite le décodage est trivial....

Algorithmique et langage C

Sujets série n°3

Ecrire un programme qui décrypte un message codé sans connaître la clé, mais en demandant sa longueur (entre 8 et 16 caractères).
Pour les essais prendre un fichier texte assez gros (au moins 3 à 4ko).

Participer au challenge d'algorithmique GEII (****) :

A l'adresse http://arlotto.univ-tln.fr/cours_de_c/tpserie3 , vous trouverez un fichier **challenge1.pdf** codé avec l'algorithme ci-dessus. Le fichier clair est un fichier au format adobe pdf. La longueur de la clé de vous est pas donnée.

Le premier étudiant qui envoie un mail à arlotto@univ-tln.fr avec le mot de passe, le fichier décodé et le code source du programme utilisé gagnera un gros paquet de bonbons.

Pour ceux qui trouveraient cet exercice trop simple, un fichier **challenge2.pdf** a été crypté par une méthode légèrement différente :
Avant le cryptage, chaque caractère du fichier original est répété trois fois (expansion). Puis un caractère sur trois choisi aléatoirement est remplacé par une valeur aléatoire (brouillage). Ensuite le fichier est crypté par la même méthode que précédemment mais avec une clé de longueur triple (cryptage).
La clé entrée par l'utilisateur est prise une fois à l'endroit, une fois à l'envers et une fois avec la casse des caractères alphabétiques inversée.

Exemple :

texte original :	bonjour monsieur
expansion :	bbbooonnnjjjoooouuurrr mmmooonnnsssiiieeeuuurrr
brouillage :	bzbRoo=jjoouu#ur"r G mmPo>onn)ss'ii&e(eu1uxrr
clé: "SecreT5"	SecreT55TerceSsECRt5SecreT55TerceSsECRt5Sec

La longueur de clé est comprise entre 4 et 9.
Malgré son apparente complexité, ce code n'est pas vraiment plus difficile à casser.
Le paquet de bonbon sera tout de même plus gros.....