Exercices à faire pendant le TP 10 et le TP 11

Codage et décodage d'un fichier

On s'intéresse au codage de textes, avec le principe suivant :

- Le texte à coder est découpé en une suite de segments de telle façon que chaque segment soit composé d'une ou plusieurs occurrences du même caractère.
- Chaque segment est codé par deux entiers n et c où n est le nombre de caractères contenus dans le segment, et c est le code ASCII du caractère qui apparait dans ce segment.
- Le code final est composé de la suite de tous les entiers obtenus, séparés par des espaces.

(ce codage met en oeuvre la "compression des répétitions", qui est à la base de plusieurs méthodes de compression d'images)

Par exemple, le texte : ***** bonne chance *****
est codé par la suite d'entiers :
5 42 1 32 1 98 1 111 2 110 1 101 1 32 1 99 1 104 1 97 1 110 1 99 1 101 1 32 5 42 1 10

1. On se donne un fichier de nombres entiers positifs, qui est le résultat du codage d'un texte.

Ecrivez un programme qui lit ce fichier et le décode pour afficher à l'écran le texte initial.

Pour cela :
- écrivez la fonction `void decoder (FILE *f)`, qui fait ce travail de décodage, et dans laquelle f est un pointeur vers le fichier à décoder ;
- puis écrivez la fonction `main`, qui contient la saisie du nom du fichier à décoder et appelle la fonction `decoder` .

2. Allez chercher sur la page Ametice de l'UE le fichier code.txt, et décodez-le avec votre programme. Si l'affichage est trop rapide, ralentissez le en ajoutant dans votre programme une "boucle vide" bien calibrée.
Ensuite, modifiez votre programme pour qu'il imprime dans un fichier dessin.txt le résultat du décodage de code.txt.

3. Programmez l'opération inverse : le codage d'un texte.
L'appel de cette fonction `codage` sur le fichier dessin.txt produit un fichier nv-code.txt, qui doit être identique à code.txt. Vérifiez-le en décodant nv-code.txt.

Représentation de fractales de Mira

Ecrivez un programme qui calcule et écrit dans un fichier les valeurs successives du couple $\{x_n, y_n\}$ obtenues selon les règles de récurrence :

$$\begin{aligned} x_{n+1} &= ax_n + \frac{2(1-a)x_n^2}{1+x_n^2} + by_n \\ y_{n+1} &= -x_n + ax_{n+1} + \frac{2(1-a)x_{n+1}^2}{1+x_{n+1}^2} \end{aligned}$$

Tracez avec Gnuplot l'ensemble des couples de valeurs $\{x_n, y_n\}$ en prenant, par exemple, les paramètres suivants : $x_0 = 12$, $y_0 = 0$, $a = 0,7$ et $b = 0,9998$.

Vous pouvez chercher sur internet d'autres équations de fractales. Puis écrivez le programme qui les dessine à l'écran.

Deviner un mot : le Jeu du Pendu

Ce jeu est initialisé par le choix d'un mot formé de caractères alphabétiques minuscules (au plus dix lettres), enregistré dans l'ordinateur mais gardé secret. Le joueur doit deviner ce mot.
Le joueur dispose d'un tableau qui est initialement vide. Il propose successivement des lettres, qui seront posées dans ce tableau si elles correspondent à des lettres du mot secret. A tout moment, s'il pense avoir deviné le mot secret, le joueur peut proposer un mot. S'il a trouvé le mot secret, il a gagné.

Le joueur peut perdre de deux façons : soit il propose un mot qui n'est pas le bon, soit il a proposé 10 lettres et n'a toujours pas trouvé le mot.

Nous allons organiser le programme de la façon suivante :

1. Le mot secret est enregistré dans la variable *mot_secret* de type tableau de N caractères. Le joueur dispose du tableau *tab* de N caractères, initialement rempli de caractères '_'.
2. Le joueur propose une lettre. Si elle est présente dans une ou plusieurs cases de *mot_secret*, elle est posée aux mêmes places dans *tab*.
3. Le joueur choisit de proposer :
   – soit un mot entier ; si c'est le mot secret, le joueur a gagné, sinon il a perdu la partie.
   – soit une lettre (retour au point 2).
4. Les étapes 2 et 3 sont répétées jusqu'à ce que le joueur gagne ou perde.

Voici un exemple d'exécution du programme, en supposant que le mot secret est "couvrir" :

```
le mot secret a 7 lettres
contenu du tableau : _ _ _ _ _ _ _
que voulez-vous proposer : 1. un mot     2. une lettre
votre reponse : 2
votre lettre : e
lettre absente !
que voulez-vous proposer : 1. un mot     2. une lettre
votre reponse : 2
votre lettre : r
contenu du tableau : _ _ _ _ r _ r
que voulez-vous proposer : 1. un mot     2. une lettre
votre reponse : 2
votre lettre : o
contenu du tableau : _ o _ _ r _ r
que voulez-vous proposer : 1. un mot     2. une lettre
votre reponse : 1
votre mot : couvrir
gagne !
```

On vous donne ci-dessous la fonction *enreg_secr*, qui est exécutée au début du programme : cette fonction permet d'enregistrer le mot secret, et renvoie le nombre de lettres de ce mot.

```
int enreg_secr (char mot_secret[])
{
   int  lg ;
   printf ("donner le mot secret : ");
   fgets (mot_secret, N, stdin) ;
   lg = strlen (mot_secret) ;
   return (lg-1) ;
}
```

(Remarque : la longueur réelle du texte saisi est `(lg-1)`, car `fgets` recopie également dans t le caractère "retour à la ligne" qui termine le texte saisi au clavier.).

1) Ecrivez la fonction `void aff_tab_char (char t[], int n)` affiche sur une même ligne les n éléments du tableau t séparés par un espace. Cette fonction servira à l'affichage du contenu de *tab*.

2) Ecrivez une fonction
`void poser_lettre (char c, char mot_secret[], int lg, char tab[])`
qui parcourt les lg cases du tableau *mot_secret*, et qui, pour chaque case de *mot_secret* qui contient le caractère c, écrit c dans la case correspondante de *tab*. Cette fonction affiche ensuite le contenu de *tab*. Si la lettre n'a pas d'occurrence dans le mot secret, on affiche la réponse "lettre absente !" à l'écran.

3) Ecrivez une fonction `void essai_mot (char mot_secret[])`
qui demande à l'utilisateur de proposer un mot, et qui donne la réponse à cette proposition : soit "gagné ! ", soit "perdu ! ". Pour cela, utilisez la fonction $fgets$ comme dans la fonction *enreg_secr*, et la fonction *strcmp*.
Rappel : si x et y sont des chaînes de caractères, l'instruction `g = strcmp(x,y)` donne à g la valeur 0 si x et y sont identiques, et une valeur non nulle sinon.

4) Ecrivez une fonction
`void essai_lettre (char mot_secret[], int lg, char tab[])`
qui demande à l'utilisateur de proposer une lettre, puis appelle la fonction *poser_lettre*.

5) Ecrivez la fonction *main* qui orchestre le tout. Cette fonction :

1. appelle la fonction *enreg_secr* ;
2. affiche le nombre de lettres du mot secret, initialise *tab*, et affiche le contenu de *tab* ;
3. demande à l'utilisateur ce qu'il veut proposer : un mot ou une lettre ;
4. selon la réponse de l'utilisateur, la fonction appelle *essai_mot* ou *essai_lettre*, puis revient au point 3 si la partie n'est pas terminée ;
5. affiche à l'écran le résultat de la partie, avec un message approprié.

6) On dispose maintenant du fichier "fic-100-mots-secrets.txt", composé de 100 lignes, tel que chaque ligne contient un mot composé de 6 à 10 lettres, et on change la procédure de démarrage du jeu : on tire au hasard un entier k de [1,100], et le mot à deviner est celui qui se trouve sur la *k_ième* ligne du fichier. Le fichier est à télécharger sur la page Ametice de l'UE.

Ecrivez une fonction `int enreg_secr_bis (char mot_secret[])`
qui correspond à cette nouvelle procédure de démarrage. Puis modifiez la fonction *main*.

Le jeu du Mastermind

Vous allez programmer le célèbre jeu de Mastermind qui consiste à rechercher une combinaison secrète de quatre couleurs choisies parmi six couleurs. Nous considérons les six couleurs suivantes : rouge (R), bleu (B), vert (V), jaune (J), noir (N) et orange (O).

Principe du jeu :

1. L'ordinateur choisira de façon aléatoire une combinaison secrète de quatre couleurs parmi les six couleurs possibles. Une couleur peut être présente zéro, une ou plusieurs fois. Soit par exemple la combinaison secrète à rechercher : RJRB.
2. L'utilisateur du programme va faire un premier essai en proposant une combinaison de quatre couleurs, par exemple : VBRJ.
3. Le programme compare l'essai à la combinaison secrète, puis affiche le nombre x de couleurs qui figurent à la bonne place dans l'essai, ainsi que le nombre y de couleurs qui sont présentes mais qui ne figurent pas à la bonne place dans l'essai. Pour l'essai VBRJ, les valeurs de ces deux nombres sont $x = 1$ et $y = 2$.
4. Les étapes 2 et 3 sont répétées jusqu'à ce que l'utilisateur trouve la combinaison secrète, c'est-à-dire lorsque $x = 4$.

Principe de programmation :

- Ecrire une fonction `initial` sans arguments qui génère une combinaison aléatoire C de quatre couleurs . C sera la combinaison secrète à trouver. Cette fonction calculera également le nombre de fois où chacune des six couleurs apparaît dans la combinaison C et le stockera dans un tableau T de six entiers. La chaîne de caractères C et tableau T seront déclarés comme variables globales.
- Ecrire une fonction `essai` qui permet à l'utilisateur du programme de proposer une combinaison E. Cette fonction aura pour arguments la chaîne de caractères E ainsi qu'un tableau F de six entiers permettant de comptabiliser le nombre de fois où chacune des six couleurs apparaît dans la combinaison E.
- Ecrire une fonction `resultat` qui compare l'essai E à la combinaison C, qui affiche les valeurs des nombres x et y définis ci-dessus (voir le principe du jeu) et qui retourne la valeur de x. Cette fonction a pour arguments la chaîne de caractères E et le tableau F.
- Ecrire la fonction `main` : elle fait appel à la fonction `initial`, puis fait appel aux fonctions `essai` et `resultat` jusqu'à ce que l'essai E soit identique à la combinaison recherchée C.

Conseils pour le calcul de x et y :

Le calcul de x dans la fonction `resultat` ne présente pas de difficulté, il suffit de comparer les chaînes de caractères E et C . En revanche, le calcul de y dans la fonction `resultat` est plus complexe ; pour le faire, comparez le contenu des tableaux F et T .