Mini-projet 3

Jean Simard

Jeudi 6 novembre 2008 – Vendredi 7 novembre 2008

1 Rendre le TP

Vous commencerez par commenter votre programme dans un fichier `README`. Vous n'oublierez pas d'indiquer votre nom dans ce fichier et tous les détails que vous jugerez nécessaires.

Après avoir mis vos fichiers dans un répertoire nommé `Simard_Jean` (vous remplacerez par votre nom et votre prénom), vous compresserez le répertoire dans un fichier `TP12-Simard_Jean.tar.bz2` (remplacez également par votre nom et votre prénom).

```
tar -cvjf TP12-Simard_Jean.tar.bz2 ./Simard_Jean/
```

Ensuite, envoyez ce fichier dans un mail ayant pour sujet (en remplaçant les noms et prénoms).

```
[CCI] TP12 Simard Jean
```

Tâchez de respecter précisément le format des noms de fichiers, de répertoires et de mail.

2 Rappel sur les fichiers

Pour écrire dans un fichier, nous allons utiliser le type `FILE`. Pour être plus précis, un simple pointeur sur ce type sera suffisant.

Afin d'ouvrir un fichier ou de créer ce fichier le cas échéant, nous allons utiliser la fonction `fopen`. Cette fonction prend en argument le nom du fichier puis une chaîne de caractère indiquant la manière dont le fichier est ouvert : en écriture ou en lecture. D'autres options sont disponibles mais ne seront pas traitées ici. La fonction renvoie un pointeur sur un `FILE`. Par exemple, pour ouvrir un fichier `TP12.txt` en lecture (on ne veut pas le modifier), on écrira

```
FILE * pFichier = NULL;
pFichier = fopen( "TP12.txt", "r" );
```

Ensuite, on va pouvoir fermer le fichier à l'aide de la fonction `close`. Cette fonction prend simplement le pointeur sur un `FILE` en entrée.

```
fclose( pFichier );
```

Entre le moment où le fichier s'ouvre et le moment où il se ferme, nous allons pouvoir effectuer des opérations de lectures ou des opérations d'écriture en fonction du mode d'ouverture du fichier.

Pour écrire dans un fichier, nous utiliserons la variante `fprintf` de la fonction `printf`. Cette fonction va écrire dans un fichier au lieu d'afficher à l'écran. Elle prend un argument supplémentaire qui sera le pointeur de fichier.

```
int i = 7;
fprintf( pFichier, "i = %d\n", i );
```

Nous venons donc d'écrire `i = 7` dans le fichier en ajoutant un passage à la ligne à la fin de la ligne.

Imaginons maintenant que nous désirions lire ce même fichier. Nous utiliserions alors une variante de la fonction `scanf` qui est la fonction `fscanf`. De la même façon que la fonction `fprintf`, cette fonction prend simplement un argument supplémentaire pour indiquer le fichier.

```
fscanf( pFichier, "i = %d\n", &i );
```

Ici, au lieu de lire les caractères entrés au clavier comme dans la fonction `scanf`, on lit les caractères dans le fichier.

3 Carnet d'adresses de courriel

Nous allons ici créer un programme permettant la gestion d'un carnet d'adresses de courriel. Pour la création d'un tel programme, plusieurs fonctions peuvent être utiles. Je vais donc proposer plusieurs fonctions possibles.

3.1 La structure

Nous allons travailler dans cet exercice avec des listes simplement chaînées. L'ensemble des contacts seront contenus dans cette liste simplement chaînée.

Exercice 1 *Écrire une structure pour qu'elle puisse contenir trois chaînes de caractères : le nom, le prénom et l'adresse de courriel. On pourra utiliser une chaîne de caractères de grande taille temporaire pour lire les noms, prénoms et adresses de courriel (comme déjà vu dans les TP précédents). Cette structure devra constituer les maillons de notre liste simplement chaînée.*

3.2 Le menu

Pour ce programme, nous avons avoir besoin d'un menu déroulant pour choisir les différentes opérations qu'on va pouvoir effectuer.

Exercice 2 *Écrire la fonction* `main` *de telle manière qu'elle permette d'afficher un menu pour l'utilisateur. Ce menu affichera les différentes opérations proposées par le programme. Il faudra une structure* `switch` *qui appellera les différentes opérations possibles en fonction du choix de l'utilisateur. Après chaque opération, il faudra que le menu s'affiche de nouveau.*

3.3 Création d'un contact

Nous pouvons tout d'abord écrire une fonction permettant de créer un nouveau contact. En effet, cette fonction va permettre de remplir notre liste chaînée.

Exercice 3 *Écrire une fonction afin de créer un nouveau contact et de l'ajouter dans la liste chaînée. Cette fonction devra demander à l'utilisateur un nom, un prénom puis une adresse de courriel correspondante. À partir de ces informations, on ajoutera ce nouveau contact dans la liste chaînée (au début, à la fin ou à l'endroit qui vous arrange). Il faudra donc utiliser la fonction* `malloc` *pour allouer l'espace mémoire nécessaire à la création d'un nouveau contact.*

3.4 Libérer l'espace mémoire

Nous créons une liste simplement chaînée en créant de nouveaux contacts au-fur-et-à-mesure que l'utilisateur les enregistre. À la fin du programme, nous allons donc devoir libérer cette mémoire.

Exercice 4 *Écrire une fonction qui libère la mémoire de la liste chaînée. Pour cela, il faudra écrire une fonction supplémentaire pour libérer l'espace mémoire d'un seul maillon de la liste. Il faudra donc utiliser la fonction* `free` *afin de libérer cette mémoire. Attention, il ne faudra pas oublier de libérer les chaînes de caractères dans chaque maillon avant de libérer la mémoire d'un maillon.*

3.5 Afficher les contacts

Nous sommes en train de créer un carnet d'adresses de courriel. Il peut donc être utile d'afficher les noms et les adresses correspondantes.

Exercice 5 *Écrire une fonction qui affiche tous les contacts présents dans le carnet d'adresses.*

3.6 Enregistrer le carnet d'adresses

Dans ce programme, nous sommes capables de créer une liste chaînée au sein du programme. Mais lorsque nous allons quitter le programme, la liste n'existera plus. C'est pour cette raison que nous devons pouvoir enregistrer le carnet d'adresses dans un fichier.

Exercice 6 *Écrire une fonction qui enregistre les noms, prénoms et adresses de courriel dans un fichier. La fonction demandera à l'utilisateur de donner le nom du fichier. Pour le format du fichier, on écrira le nom, le prénom et l'adresse de courriel sur trois lignes distinctes. Une ligne vide sera écrite pour séparer deux contacts dans le fichier.*

3.7 Charger un carnet d'adresses

Si nous sommes capables d'enregistrer un carnet d'adresses dans un fichier, nous devrions également pouvoir lire un fichier pour créer un carnet d'adresses.

Exercice 7 *Écrire une fonction qui chargera les noms, prénoms et adresses de courriel depuis un fichier. La fonction demandera à l'utilisateur de donner le nom du fichier. Cette fonction devra créer la liste chaînée contenant toutes les informations présentes dans le fichier.*

3.8 Suppression d'un contact

Dans ce carnet, nous devrions pouvoir supprimer un contact. Pour effectuer cette opération, vous allez devoir utiliser la fonction `strcmp` de la bibliothèque `string`. Cette fonction va vous permettre de comparer deux chaînes de caractères. La fonction prend deux arguments qui seront les deux chaînes de caractères à comparer et renvoie la valeur zéro si les deux chaînes sont identiques.

Exercice 8 *Écrire une fonction qui permette de supprimer une contact du carnet d'adresses. Cette fonction devra demander à l'utilisateur le nom du contact à supprimer. Il est possible que plusieurs contacts correspondent au nom dans le carnet. C'est pour cette raison que le programme devra proposer de confirmer ou non la suppression pour chaque contact dont le nom correspond.*

3.9 Modification d'un contact

Dans le cas où l'utilisateur effectue une erreur de saisie ou modifie l'adresse de courriel d'un contact, il doit pouvoir modifier un contact.

Exercice 9 *Écrire une fonction qui permette de modifier un contact. Cette fonction devra demander à l'utilisateur le nom du contact à modifier. Il est possible que plusieurs contacts correspondent au nom dans le carnet. C'est pour cette raison que le programme devra proposer de modifier ou non chaque contact dont le nom correspond.*

Lorsque le programme propose de modifier un contact, il lui propose de modifier soit le nom, soit le prénom, soit l'adresse de courriel. Il doit donc également proposer de ne pas modifier le contact pour le cas où plusieurs contacts correspondent au nom donné par l'utilisateur.