INF203 - Travaux pratiques, séance 6
Chaînes de caractères, opérations sur des ensembles

Gestion d'un ensemble de chaînes de caractères

Pour cette première partie, le répertoire de travail est le répertoire ***Chaines***. Nous souhaitons écrire un certain nombre de fonctions qui nous permettront de gérer un ensemble S de chaînes de caractères, chacune d'elles étant associée à un entier unique dans $\{0, \ldots, |S| - 1\}$. Il s'agit d'un ensemble : une chaîne de caractères ne pourra être présente dans l'ensemble qu'au plus une fois.

Lisez le fichier ***chaines.h***. **[a]** Quel est le nom du type permettant de représenter un ensemble de chaînes de caractères ? De quelle manière sont représentées les chaînes de caractères dans un tel ensemble ? Quel est le cardinal maximal d'un tel ensemble ? Y a-t-il une limite à la taille d'une chaîne de caractères qu'il est possible de mémoriser dans un tel ensemble ? ■

Notez bien que le fichier ***chaines.h*** contient une déclaration, un certain nombre de prototypes, et une description des fonctions associées à ces prototypes. La déclaration de ces fonctions se trouve dans ***chaines.c*** mais pour l'instant, le corps de ces fonctions est vide et elles ne font donc rien. Lisez le contenu du fichier ***test_ chaines.c***. Le comportement attendu de ce programme est le suivant : le programme crée un ensemble de chaînes avec ses arguments (à partir du deuxième), puis recherche le premier argument dans cet ensemble. Voici 3 exemples de trace d'exécution attendue :

```
./test_chaines truc bidule machin chose
Copie successive des arguments dans l'ensemble : bidule - machin - chose -
Ensemble de chaines dans lequel la recherche est faite : { bidule  machin  chose }
Recherche de truc dans cet ensemble : -- Chaine absente de l'ensemble

./test_chaines truc bidule machin truc chose
Copie successive des arguments dans l'ensemble : bidule - machin - truc - chose -
Ensemble de chaines dans lequel la recherche est faite : { bidule  machin  truc  chose }
Recherche de truc dans cet ensemble : -- Chaine trouvee en position 2

./test_chaines truc bidule machin truc chose machin chose
Copie successive des arguments dans l'ensemble : bidule - machin - truc - chose - machin - chose -
Ensemble de chaines dans lequel la recherche est faite : { bidule  machin  truc  chose }
Recherche de truc dans cet ensemble : -- Chaine trouvee en position 2
```

[b] Utilisez le ***Makefile*** fourni pour compiler le programme contenu dans ***test_ chaines.c*** et vérifiez que son exécution ne fournit pas le résultat attendu. ■

Pour que le programme contenu dans ***test_ chaines.c*** fonctionne, il va falloir compléter la déclaration des fonctions de manipulation d'un ensemble de chaînes de caractères.
Complétez le fichier ***chaines.c***. Il est recommandé de compléter les fonctions selon l'ordre indiqué ci-dessous, et de vérifier au fur et à mesure en exécutant le programme ***test_ chaines*** que votre implémentation fonctionne.
Écrivez d'abord `init_chaines` en donnant au cardinal de l'ensemble la valeur 0.
Écrivez ensuite une version de `ajouter_chaine` qui ajoute la chaîne à l'ensemble **sans vérifier** qu'elle n'est pas déjà présente.
Puis écrivez `nombre_chaines`, `trouver_chaine` et `recuperer_chaine`.
Enfin modifiez `ajouter_chaine` pour tester si la chaîne est présente ou non dans l'ensemble (par exemple en utilisant `trouver_chaine`).

[c] Joignez le texte de ***chaines.c*** à votre compte rendu. Indiquez tous les jeux de test effectués pour s'assurer que le programme est correct. ■

la commande de la semaine : `wc`.

[TP6] La commande `wc` permet de compter le nombre de caractères, de mots, de ligne d'un fichier donné en argument, ou de l'entrée standard s'il n'y a pas d'argument.
Essayez successivement :

```
wc -c Candide_chapitre1.txt
wc -w Candide_chapitre1.txt
wc -l Candide_chapitre1.txt
wc Candide_chapitre1.txt
```

[d] À quoi correspondent les différentes options (`-c, -w, -l`) de `wc` ? Qu'affiche `wc` utilisé sans option ? ■

Exécutez la commande :
`grep chateau Candide_chapitre1.txt | wc -l`
[e] Quel est le résultat de cette commande ? Que signifie-t-il ? ■

Essayez d'afficher en une seule commande le nombre de lignes comportant à la fois Cunegonde et Candide dans ***Candide_chapitre1.txt***.
[f] Quelle commande utilisez-vous ? Combien de lignes répondent à la question ? ■

Représentation d'un ensemble, d'un sous-ensemble

Le répertoire de travail, pour la suite est le répertoire ***Clients***. Le fichier ***gestion.c*** contient un programme qui permet à l'utilisateur d'initialiser un ensemble de clients (d'une banque de cacahuètes) et de comptes associés (nombre de cacahuètes possédées par le client), de modifier les montants de ces comptes à l'aide de trois opérations, qui sont définies dans le fichier ***operations.c***, puis de sauvegarder l'état des comptes dans un (nouveau) fichier.

[g] Dans quel fichier est définie le type `ensemble_clients` qui représente un ensemble des clients et leur compte ? Faites un dessin représentant une variable de ce type pour un ensemble de 3 clients de votre choix. ■

Dans les fichiers fournis, il y a un certain nombre d'erreurs que vous allez devoir corriger.

1. Il manque une directive `include` dans ***clients_in.c***. Une façon de s'en rendre compte est d'essayer de créer ***clients_in.o***.
   [h] Quelle commande de compilation saisissez-vous pour créer ***clients_in.o*** à partir de ***clients_in.c*** ? Que signifie le message d'erreur ? Comment faites-vous pour y remédier ? ■

2. Essayez de compiler de la même façon ***gestion.c***. Quel est le problème ? Sur le modèle de ***clients.h***, créez les fichiers ***.h*** manquants, et vérifiez que vous pouvez maintenant compiler ***gestion.c***. N'oubliez pas d'inclure également ces fichiers ***.h*** dans les fichiers ***.c*** correspondants.
3. Le fichier ***Makefile*** qui vous est fourni est également incorrect.

   — vérifiez/corrigez les lignes de dépendance des fichiers ***.o*** : on rappelle que ***fich.o*** dépend uniquement de ***fich.c*** et de tous les fichiers ***.h*** *utilisateur* inclus dans ***fich.c***.
   — vérifiez/corrigez les lignes de création des ***.o***. **[i]** De quelle forme est la ligne de création de ***fich.o*** ? ■

   — **[j]** À partir de quels fichiers ***.o*** est fabriqué l'exécutable ***gestion*** ? Pourquoi ne met-on pas les fichiers ***.h*** ou ***.c*** sur la ligne de dépendance de ***gestion*** ? ■

   — Corrigez la ligne de dépendance de ***gestion*** ainsi que la ligne de création de ***gestion***. Vérifiez que votre ***Makefile*** fonctionne. Relisez si nécessaire le manuel en ligne de la commande `touch`. Modifiez (en une seule commande) les dates de dernière modification de tous les fichiers ***.c*** et ***.h*** du répertoire.
   [k] Quelle (ligne de)commande saisissez-vous pour celà ? ■
   Recompilez le programme avec `make`, et vérifiez que l'ensemble de la compilation se déroule bien.

[l] Dessinez le graphe de dépendance sur votre compte rendu. ■

Vous disposez d'un fichier de clients ***Fichier_Clients***. Modifiez-le en ajoutant quelques clients de votre choix, et en respectant le format du fichier. Exécutez le programme ***gestion*** en donnant comme fichier de clients d'entrée ***Fichier_Clients***, et comme fichier de sortie un nom de votre choix, et en n'utilisant que les requêtes D(épôt) et R(etrait), puis Q(uitter).
[m] Quel est le contenu du fichier de sortie ? Comment l'expliquez-vous ? ■

Le problème vient du fait que la fonction supposée écrire les clients dans un fichier ne fait rien. Complétez cette fonction, en vous inspirant de la fonction qui affiche les clients à l'écran. N'oubliez pas que dans le fichier de clients à écrire, il faut écrire le nombre de clients sur la première ligne. Testez cette fonction puis **[n]** joignez son texte à votre compte rendu. ■

En vous aidant du texte des fonctions `depot` et `retrait`, complétez le fichier ***operations.c*** en écrivant la fonction `virement` qui permet de transférer un montant d'un compte vers un autre. Testez cette fonction.
[o] Que se passe-t'il si un client se fait un virement à lui-même ? ■
[p] Un virement peut se décomposer en un retrait et un dépôt, sur deux comptes donnés. Voyez-vous une raison pour avoir défini une fonction `virement` plutôt que d'avoir choisi d'exécuter successivement les fonctions `depot` et `retrait` quand l'utilisateur emploie la requête V(irement) ? ■

Le programme ***sous_ensembles.c*** tel qu'il vous est fourni construit un sous-ensemble de l'ensemble des clients : les clients dont le compte est négatif. Ce sous-ensemble est représenté par le tableau de "booléens" `sous_ens_clients`.
Pour mémoriser cet ensemble dans un fichier, on a prévu la fonction `ecrire_les_elements`. Le format du fichier à écrire sera le même que celui des fichiers de clients déjà lus ou écrits au cours de ce TP : sur une première ligne, le nombre d'éléments, puis, chacun sur une ligne, le nom et le compte des clients appartenant à l'ensemble. Écrivez cette fonction dans le fichier ***clients_out.c***.
[q] Comment faites-vous pour compter le nombre d'éléments de l'ensemble à écrire dans le fichier ? ■

Ajoutez le profil de la nouvelle fonction dans ***clients_in.h***. Ajoutez les directives `include` pour les fichiers d'interface indispensables, en tête du programme ***sous_ensembles.c***.
[r] Donnez la liste des fichiers d'interface que vous avez inclus en tête du programme. ■

Ajoutez au ***Makefile*** :
- la ligne de dépendance de ***sous_ensembles.o***, et la ligne de fabrication de ce fichier,
- la ligne de dépendance de l'exécutable ***sous_ensembles*** et la ligne de fabrication de ce fichier.

Compilez avec la commande `make sous_ensembles`.
Testez votre programme avec un fichier d'entrée d'une dizaine de clients de votre choix (dont certains avec des comptes négatifs).

Exercice complémentaire :

Ajoutez à ***sous_ensembles.c***, sur le modèle de `clients_debiteurs`, des fonctions qui calculent le sous-ensemble des clients dont le nom commence par `'H'`, le sous-ensemble des clients dont le nom fait moins de 5 lettres ... ou tout autre critère de votre choix. Modifiez le programme pour afficher ces sous-ensembles dans des fichiers différents. *Indication : il faudra modifier le nombre d'arguments de la ligne de commande.* Testez !
[s] Joignez votre programme ***sous_ensembles.c*** complété à votre compte rendu. ■