INF1005C
Programmation procédurale
Session automne 2014
Travail dirigé No. 6
Allocation dynamique et représentation des données

Objectifs :	Permettre à l'étudiant de manipuler l'allocation dynamique et statique. Afficher des caractères hors ASCII. Maitriser la représentation interne des données.
Durée :	2 séances de laboratoire.
Date limite de remise :	Jeudi 4 décembre 2014 à 23h30. Remise uniquement via Moodle.
Documents à remettre :	Remettre seulement l'ensemble des fichiers **.cpp** et **.h** dans un dossier compressé au format **.zip**
Directives :	- N'oubliez pas les en-têtes de fichiers et de fonctions ni, aux endroits appropriés, les commentaires sur le code. - Vous devez supporter les caractères accentués et/ou symboles spéciaux. - Vous devez éliminer ou expliquer tout avertissement donné par le compilateur et l'analyseur. - Respectez le guide de codage et le principe DRY (don't repeat yourself); vous pouvez ajouter autant de fonctions que vous voulez pour que le programme soit facile à lire et sans duplication de code. **- Il est interdit d'utiliser les variables globales.**

Exercice 1 :

A. Description des structures

Une petite bibliothèque utilise un programme pour la manipulation et la gestion de sa collection. Chaque livre est mémorisé dans un enregistrement de type `Livre` tel que suit :

```
struct Livre {
       char cote[8];
       wchar_t titre[50];
       ??? anneeEdition;
       ??? nExemplaires;
       ??? nDisponibles;
};
```

- cote : Une chaine de caractères alphanumériques de taille fixe (7 caractères), en ASCII.
- titre : Titre du livre (49 caractères au maximum), en Unicode.
- anneeEdition : Entier compris entre 1600 et 3000 représentant l'année d'édition du livre.
- nExemplaires: Entier représentant le nombre d'exemplaires possédés par la bibliothèque. 200 au maximum.

- nDisponibles: Entier représentant le nombre d'exemplaires actuellement disponibles à la bibliothèque.

L'ensemble des livres est ensuite enregistré dans un fichier binaire où les livres sont contenus l'un après l'autre.
Pendant que le programme marche, l'ensemble des livres est chargé dans une structure `Collection` tel que suit :

```
struct Collection {
	int nLivresAlloues;
	int nLivres;
	Livre** livres;
};
```

- nLivresAlloues: Entier représentant le nombre maximal de livre que la collection peut actuellement contenir (la taille du tableau alloué).
- nLivres: Entier représentant le nombre de livre présentement contenus dans la collection. Ce chiffre doit donc toujours être inférieur ou égal à `nLivresAlloues`.
- livres: Pointeur d'une structure `Livre`. Ce pointeur pointe vers un tableau de taille `nLivresAlloues` pour les livres dans la collection.

B. Travail à effectuer

Vous devrez ensuite rédiger un programme qui permet la manipulation d'un ensemble de livres. Ces livres ont des noms en Unicode, et une des collections fournies contient des noms grecs pour vérifier si ça fonctionne correctement (dans la fenêtre de console, assurez-vous d'utiliser une police de caractères Unicode, tel que "Consolas" ou "Lucida Console"; aller dans les "propriétés" de la fenêtre de console pour changer la police).

Pour voir le travail général à effectuer, voici ce que doit permettre le programme. La fonction **main()** affiche un menu offrant les options suivantes :

1. Charger collection.
2. Afficher collection.
3. Retrouver livre par son titre.
4. Ajouter un livre.
5. Sauvegarder la collection.
6. Quitter

- Choix 1 : Demande à l'utilisateur d'entrer le nom du fichier binaire contenant les livres et charge la collection en mémoire (dans une structure Collection).
- Choix 2 : Affiche l'ensemble des livres contenus dans la collection. Le format d'affichage est laissé à votre discrétion.
- Choix 3 : Demande à l'usager d'entrer une partie du titre d'un livre recherché et affiche les informations sur les livres trouvés. N'oubliez pas de libérer l'espace après l'affichage. Le format d'affichage est laissé à votre discrétion.
- Choix 4 : Demande à l'utilisateur d'entrer les informations sur un livre et ajoute ce dernier à la collection actuellement chargée.
- Choix 5 : Sauvegarde la collection actuellement chargée dans un fichier binaire. Le nom du

fichier doit être demandé à l'utilisateur.
- Choix 6 : Met fin au programme.
Les choix 2, 3 et 4 ne devraient pas être accessibles si aucune collection n'est chargée.
Conseil : Utilisez une fonction pour l'affichage du menu et une fonction par option (voir la description des fonctions ci-bas). La fonction pourrait afficher les choix et lire l'entrée de l'utilisateur avant de la retourner. Il sera ensuite possible d'utiliser un `switch` dans la fonction **main()** en utilisant le choix (entier) retourné par la fonction d'affichage du menu.
Note : N'oubliez de libérer l'espace alloué avant de charger une nouvelle collection.

Pour réaliser ce programme vous devez :

1. Avec les informations que vous avez, choisissez les types pour `anneeEdition`, `nExemplaires` et `nDisponibles` de façon à minimiser la taille de la structure. Utilisez des types prédéfinis en C++ (pas de "bit fields" qui ne sont pas matière au cours).

2. Écrire la fonction **chargerCollection()** qui prend en paramètre le nom du fichier binaire contenant la collection de livres. La fonction retourne une nouvelle collection (de type `Collection`).
   La fonction détermine le nombre de livres enregistrés dans le fichier binaire, assigne aux champs `nLivresAlloues` et `nLivres` le nombre de livres, et ajoute 3 à la valeur `nLivresAlloues`.
   La fonction alloue dynamiquement un tableau de taille `nLivresAlloues` dans `Livre**`. Ensuite, la fonction alloue des `Livre`s pour lire le fichier et y stocker les informations, puis placer les pointeurs vers ces livres dans le tableau alloué.
   La fonction retourne un une collection vide si le chargement n'a pas réussi.
   Note : Aucun `cin/wcin` et `cout/wcout`.

3. Écrire la fonction **libererCollection()** qui prend en paramètre une collection.
   La fonction libère toute la mémoire dynamiquement allouée dans la structure `Collection`.
   Note : N'oubliez pas de modifier les champs `livres`, `nLivres` et `nLivresAlloues` comme il se doit. Aucun `cin/wcin` et `cout/wcout`.

4. Écrire la fonction **retrouverLivresParTitre()** qui prend en paramètres une chaine de caractères Unicode et une collection de livres. La fonction retourne l'ensemble des livres de la collection qui contiennent dans leur titre la chaine passée en argument. Noter que la structure `Collection` sert justement à représenter un ensemble de livres. Cette fonction ne devrait faire aucune copie de `Livre`, uniquement les pointeurs. Attention lors de la libération de l'ensemble des livres trouvés que cette fonction n'a pas copié les livres.
   Note : Aucun `cin/wcin` et `cout/wcout`.
   Aide : Vous pouvez utiliser la fonction **wcsstr(*chaineOùChercher, chaineÀTrouver*)** sur un wchar_t* (en utilisant .c_str() sur la wstring), ou **find()** sur une wstring (en convertissant le wchat_t* en wstring).

5. Écrire la fonction **ajouterLivre()** qui prend en paramètre une collection. La fonction utilise des informations lues au clavier (cote, titre, année et nombre d'exemplaires) et ajoute le livre à l'enregistrement en s'assurant qu'il reste assez d'espace.
   S'il ne reste plus assez d'espace, il faut réallouer un plus grand tableau dans le champ `Livre**` et y copier les pointeurs vers les livres déjà présents avant d'ajouter le nouveau (cette fonction ne devrait faire aucune copie de `Livre`, uniquement les pointeurs). Le nouveau tableau possèdera 3 cases de plus que l'ancien tableau.
   Notes : Vous assumez que les informations lues au clavier sont intègres (cote de 7 caractères, nombre entier pour l'année…etc.). Le titre peut contenir des caractères spéciaux Unicode.
   Vous pouvez ajouter des paramètres si vous le jugez nécessaire.
   N'oubliez pas de modifier les champs `nLivres` et `nLivresAlloues` selon les modifications apportées à la structure `Collection`, et de faire les désallocations si nécessaire.
   Aide : Vous pouvez utiliser la fonction **wcsncpy(*destination*, *source*, *nombreMaximalDeCaractères*)** sur un wchar_t* (en utilisant .c_str() sur la wstring).

6. Écrire la fonction **sauvegarderCollection()** qui prend en paramètres une collection et le nom du fichier binaire dans lequel la collection sera sauvegardée.
   La fonction enregistre, un après l'autre, tous les livres contenus dans la collection. Le fichier d'enregistrement doit avoir le nom passé en argument.
   La fonction retourne un booléen indiquant si l'enregistrement s'est correctement déroulé.

C. Unicode

Le programme doit afficher et lire des chaînes Unicode pour le titre des livres. Une fois en mode Unicode, il n'est plus possible d'utiliser `cin/cout`, il faut plutôt utiliser `wcin/wcout` (ceux-ci fonctionnent de la même manière). Les chaines Unicode sont des `wstring` au lieu de `string`, les caractères sont des `wchar_t` au lieu de `char`, et les textes entre guillemets doivent être précédés d'un L pour indiquer qu'ils sont Unicode. Nous fournissons un exemple de programme dans le main(), utilisant aussi des fonctions fournies `versString` et `versWstring` pour convertir les une `wstring` en `string` et une `string` en `wstring`, respectivement.

D. Fichiers fournis

- Exercice1.cpp: Le programme à compléter.
- Structures.h: Les définitions des struct, à compléter.
- cppcheck.zip: Les fichiers nécessaires pour Cppcheck, à décompresser. (voir annexe)
- Collection1.bin: Fichier binaire pouvant être utilisé pour vos tests.
- Collection2.bin: Fichier binaire pouvant être utilisé pour vos tests.
- Collection3.bin: Fichier binaire pouvant être utilisé pour vos tests, avec des noms grecs.

Exercice 2 : Représentation interne des données

Dans cet exercice, vous devez écrire un programme qui lit un réel court (4 octets, soit un `float`) au clavier. Le programme découpe ensuite le réel en 4 blocs de 1 octet, puis affiche chaque octet séparément en format hexadécimal.

Par exemple, si on entre au clavier "-1.0", la sortie devrait être "00 00 80 BF" (chaque octet, dans le même ordre qu'en mémoire, affichés en format hexadécimal).

Aide: L'idée est de convertir le pointeur vers le `float` en un pointeur vers des octets, similairement à ce qu'on fait pour lire/écrire dans les fichiers binaires. On peut donc faire `((unsigned char*)&f)` où f est une variable float, pour obtenir un pointeur vers des octets non-signés.

Attention! Vous devez effectuer vous-même la traduction des blocs au format hexadécimal. Il est **interdit** d'utiliser `cout << hex`, les `stringstream`, `to_string`, `itoa`, `sprintf`, ou toute autre fonction qui pourrait faire la traduction en hexa automatiquement. Pour être certain que vous n'utilisez aucune traduction automatique, utilisez uniquement des `cout.put(`*caractère*`)` pour l'affichage du résultat, et uniquement des opérations de base du langage C/C++ pour calculer les caractères à afficher.

ANNEXE : Utilisation de l'outil CPPCHECK

Le but de ces outils est de vous aider à trouver certaines erreurs dans votre programme.
Dans les propriétés du projet, dans C/C++, il est possible de choisir le niveau d'avertissement. Par défaut, le niveau est à /W3, alors que pour ce travail on demande qu'il soit à /W4.
Ceci permet de détecter certains problèmes possibles, tel que l'utilisation d'une affectation au lieu d'une égalité dans une condition (comme dans « if (x = 4) » qui devait probablement être « if (x == 4) »), les variables potentiellement non initialisées, et les conditions constantes. Pour voir la liste des erreurs et avertissements, sélectionner le menu Affichage > Liste d'erreurs et s'assurer de sélectionner les avertissements.
Une recompilation (menu Générer > Compiler, ou Ctrl+F7) est nécessaire pour mettre à jour la liste des avertissements.

Cppcheck fait d'autres analyses, permettant dans certains cas de détecter des accès hors bornes dans les tableaux, les énoncés vides, des améliorations possibles au style de programmation (réduction de la portée d'une variable; initialisation à la déclaration), etc. Un .zip de Cppcheck vous est fourni, que vous pouvez décompresser dans votre compte. Pour exécuter cppcheck.exe à partir de l'environnement Visual Studio, il faut le configurer comme outil externe :
Dans le menu Outils, choisissez Outils externes…
Dans la boîte de dialogue, appuyez sur Ajouter et dans la zone de titre entrez un nom comme « Cppcheck ».
Comme commande, cliquez sur le « … » à côté pour avoir une fenêtre permettant d'aller chercher où vous avez mis cppcheck.exe.
Comme arguments entrez : --enable=all --inconclusive --template=vs $(ProjectDir)
Comme répertoire initial entrez : $(ProjectDir)
Activez l'utilisation de la fenêtre de sortie.
Puis faites OK.

Après cette configuration, Cppcheck sera affiché dans le menu Outils, qui vous permet de l'exécuter facilement. Si un avertissement est affiché dans la fenêtre de sortie, il suffit de double-cliquer sur l'avertissement pour aller à la ligne de code qui correspond.
Votre programme ne devrait avoir aucun avertissement, ni par le compilateur, ni par Cppcheck. Pour tout avertissement restant (s'il y en a) vous devez ajouter un commentaire dans votre code, à l'endroit concerné, pour indiquer pourquoi l'avertissement peut être ignoré.