**Langage de programmation : bases**

**HAUTE ÉCOLE DE NAMUR-LIÈGE-LUXEMBOURG**

**DA-IA – bloc 1**

Exercices 3B – Pointeurs

Objectifs

* Continuez à utiliser les éléments vus dans les modules précédents
* Utiliser les pointeurs

# Exercice 1 : mettre en majuscules

Commencez par créer un nouveau projet (et solution) en lui donnant comme nom EnMajuscules, par exemple. N’oubliez pas de faire en sorte que le fichier principale, celui qui contient la fonction main, ait l’extension .c.

Écrivez une fonction qui remplace, caractère par caractère, tous les éléments d’une chaine de caractères reçue en argument par son équivalent en majuscule.

Écrivez également la partie de la fonction principale qui déclare une chaine de caractères en l’initialisant à la valeur « Bonjour », qui appelle la fonction et qui affiche ensuite la chaine après remplacement via puts.

**Consignes :**

* notations tableau dans la fonction principale.
* notations pointeur dans la fonction de remplacement.

# Exercice 2 : taille du mot le plus long

Commencez par créer un nouveau projet (et solution) en lui donnant comme nom TailleMotLePlusLong, par exemple. N’oubliez pas de faire en sorte que le fichier principal, celui qui contient la fonction main, ait l’extension .c.

Écrivez une fonction qui renvoie la longueur du mot le plus long contenu dans une phrase reçue en argument. Le but est de le faire sans passer par des fonctions déjà définies dans une quelconque bibliothèque !

Écrivez également la partie de la fonction principale permettant de tester votre fonction.

Précisions, restrictions :

* Un seul espace sépare les mots.
* La phrase commence par un mot et ne contient aucun signe de ponctuation.

Mêmes **consignes** que dans l'exercice précédent.

# Exercice 3 : bibliothèque – liste chainée simple

Le but de cet exercice est de mémoriser les informations relatives aux différents livres de votre bibliothèque personnelle dans une liste chaînée simple.

Commencez par créer un nouveau projet (et solution) en lui donnant comme nom Bibliotheque, par exemple. N’oubliez pas de faire en sorte que le fichier principal, celui qui contient la fonction main, ait l’extension .c.

Lisez ensuite attentivement l’énoncé et assurez-vous d’avoir compris la structure des données et les fonctions à définir.

## Contexte

Pour chaque livre, on mémorise les informations suivantes :

* le titre,
* l'auteur,
* l’année de parution de la version originale,
* le numéro du tome s’il fait partie d’une série, ou 0 si non,
* le titre de la série s’il fait partie d’une série, ou « » (chaine vide) si non.

Le programme doit proposer un menu afin de pouvoir

* ajouter un livre,
* supprimer un livre,
* afficher la bibliothèque, ou
* quitter le programme.

**Les livres sont mémorisés sous la forme d’une liste chaînée simple, triée par ordre alphabétique sur les auteurs, puis par ordre croissant sur l’année de parution et enfin par ordre croissant sur le numéro du tome (s’il y a deux tomes parus la même année).**

**Pour cela, il faut prévoir d’utiliser des chaînons, composés chacun d’un livre et du pointeur vers le suivant. On veut pouvoir traiter (obtenir, afficher…) un livre indépendamment de la liste chainée.**

**Plusieurs messages sont à prévoir :**

* **un message précisant que l’allocation s’est mal passée,**
* **un message précisant que le livre est déjà dans la bibliothèque,**
* **un message précisant que le livre n’existe pas dans la bibliothèque,**
* **un message précisant que l’ajout d’un livre s’est bien passé,**
* **un message précisant que la suppression d’un livre s’est bien passée.**

## ****Quelques livres****

Voici quelques idées de livres à ajouter à votre bibliothèque :

{"Antoine de Saint-Exupery", "Le Petit Prince", 1943, 0, ""}

{"Christelle Dabos", "Les Fiances de l'hiver", 2013, 0, ""}

{"Antoine de Saint-Exupery", "Le Petit Prince", 1943, 0, ""}

{"Eoin Colfer", "Artemis Fowl", 2001, 0, ""}

{"John Green", "Nos etoiles contraires", 2012, 0, ""}

{"John Green", "Qui es-tu Alaska ?", 2005, 0, ""}

{"J.R.R.Tolkien", "La Communaute de l'Anneau", 1954, 1, "Le Seigneur des anneaux"}

{"J.R.R.Tolkien", "Les Deux Tours", 1954, 2, "Le Seigneur des anneaux"}

{"J.R.R.Tolkien", "Le Retour du roi", 1955, 3, "Le Seigneur des anneaux"}

{"J.R.R.Tolkien", "Bilbo le Hobbit", 1937, 0, ""}

{"Lois Lowry", "Le Passeur", 1993, 0, ""}

{"Pierre Bottero", "La Quete d'Ewilan - L'integrale", 2003, 0, ""}

{"Suzanne Collins", "Hunger Games", 2008, 1, "Hunger Games"}

{"Suzanne Collins", "L'embrasement", 2009, 2, "Hunger Games"}

{"Suzanne Collins", "La revolte", 2010, 3, "Hunger Games"}

## ****Étape 1 : définir les structures****

Définissez une structure d’étiquette livre pour mémoriser les informations sur un livre. Son synonyme est Livre.

Définissez ensuite une structure d’étiquette chainonLivre qui contient un livre et le pointeur vers le chainon suivant. Son synonyme est ChainonLivre.

## Étape 2 : déclarer le pointeur de début

L’adresse du premier chainon de la liste chaînée est dans pBibliotheque. Dans la fonction principale, déclarer le pointeur début de liste : pBibliotheque et initialisez-le (à NULL).

## Étape 3 : définir les énumérations

Commencez par définir une énumération d’étiquette menu qui associe une constante nommée à chaque option du menu. Son synonyme est Menu.

Pour afficher le menu, vous pouvez copier la fonction suivante.

void afficherMenu(void) {

printf("%d. Ajouter un livre\n", AJOUTER\_LIVRE);

printf("%d. Supprimer un livre\n", SUPPRIMER\_LIVRE);

printf("%d. Afficher les livres\n", AFFICHER\_LIVRES);

printf("%d. Quitter\n", QUITTER);

}

Elle ne sera utilisée qu’en fin d’exercice.

Ensuite définissez une énumération d’étiquette codeMessage qui associe une constante nommée à chacun des 5 messages à afficher. Son synonyme est CodeMessage.

Voici la fonction qui permet d’afficher le message correspondant à chaque énumérateur.

void afficherMessage(CodeMessage codeMessage) {

switch (codeMessage) {

case ERREUR\_MEMOIRE: puts("Plus de place memoire !"); break;

case LIVRE\_EXISTANT: puts("Le livre existe deja !"); break;

case LIVRE\_INEXISTANT: puts("Le livre n'existe pas !"); break;

case AJOUT\_EFFECTUE: puts("Le livre est ajoute !"); break;

case SUPPRESSION\_EFFECTUEE: puts("Le livre est supprime !"); break;

}

}

## Étape 4 : afficher un livre

Définissez une fonction qui permet d’afficher les informations concernant un livre reçu en argument.

Dans le cas d’un livre hors-série, l’affichage doit ressembler à ceci :

Antoine de Saint-Exupery -> Le Petit Prince (1943)

Dans le cas d’un livre faisant partie d’une série, l’affichage doit ressembler à ceci :

Suzanne Collins -> Hunger Games - Tome 3 : La revolte (2010)

Analysez convenablement les exemples ci-dessus pour déterminer

ce qui est affiché dans quel cas, et

où chaque information est affichée.

Pour tester cette fonction, vous pouvez utiliser les données qui vous ont été données ci-dessus, par exemple en les mettant dans un tableau de structures et en affichant chaque élément, via la fonction définie, dans une boucle.

Dans ce cas, l’affichage est le suivant.

Antoine de Saint-Exupery -> Le Petit Prince (1943)

Christelle Dabos -> Les Fiances de l'hiver (2013)

Eoin Colfer -> Artemis Fowl (2001)

J.R.R.Tolkien -> Bilbo le Hobbit (1937)

J.R.R.Tolkien -> Le Seigneur des anneaux - Tome 1 : La Communaute de l'Anneau (1954)

J.R.R.Tolkien -> Le Seigneur des anneaux - Tome 2 : Les Deux Tours (1954)

J.R.R.Tolkien -> Le Seigneur des anneaux - Tome 3 : Le Retour du roi (1955)

John Green -> Qui es-tu Alaska ? (2005)

John Green -> Nos etoiles contraires (2012)

Lois Lowry -> Le Passeur (1993)

Pierre Bottero -> La Quete d'Ewilan - L'integrale (2003)

Suzanne Collins -> Hunger Games - Tome 1 : Hunger Games (2008)

Suzanne Collins -> Hunger Games - Tome 2 : L'embrasement (2009)

Suzanne Collins -> Hunger Games - Tome 3 : La revolte (2010)

## Étape 5 : ajouter un livre à la bibliothèque

Définissez une fonction permettant d'ajouter un livre dans la bibliothèque. La fonction reçoit le livre à ajouter et pBibliotheque en arguments.

Dans un premier temps, ajouter les éléments en début de liste sans vérifier que le livre existe déjà. On ajoutera la fonction de recherche et ce qu’il faut d’autre pour tenir compte du tri dans les étapes suivantes.

Cette fonction d’ajout renvoie un code de message (LIVRE\_EXISTANT ou AJOUT\_EFFECTUE).

Comme il faut également pouvoir récupérer la valeur de pBibliotheque (qui change en cas d'ajout en début de liste), il faut passer ce pointeur par adresse…

Le prototype de la fonction doit donc être similaire à la suivante.

CodeMessage ajouterLivre(Livre livre, ChainonLivre\*\* ppBibliotheque);

C’est dans cette fonction que vous devez allouer la mémoire au nouveau chainon. Si cette allocation ne se passe pas comme espéré, veillez à quitter la fonction directement en renvoyant le code de message adéquat.

Si l’allocation s’est bien passée, vous pouvez recopier les informations concernant le livre dans le chainon et ensuite effectuer le chainage. Pensez à renvoyer le code adéquat lorsque tout s’est bien passé.

Dans la fonction principale, ajoutez quelques livres à votre bibliothèque, par exemple en parcourant le tableau de structures que vous avez déjà utilisé pour tester l’affichage et en ajoutant chacun de ces livres à la bibliothèque.

Pour vérifier si l’ajout s’est bien passé, vous pouvez simplement adapter la boucle d’affichage déjà écrite de façon à afficher le contenu de la bibliothèque comme demandé. *Les notations « tableau » doivent devenir des notations « pointeur ».*

N’oubliez pas de tester le cas d’un livre qui serait déjà présent… *Il suffit de mettre deux fois le même livre dans le tableau de structures.*

## Étape 6 : afficher toute la bibliothèque

Écrivez une nouvelle fonction qui permet d’afficher toute la bibliothèque. Elle reçoit le pointeur pBibliotheque.

Ne refaites pas ce qui a déjà été fait ! Reprenez la dernière boucle que vous avez écrite, celle qui parcourt la liste chainée et affiche chaque livre, et recopiez-la dans le corps de la fonction.

Faites ensuite appel à cette fonction, dans la fonction principale, en lieu et place de la précédente boucle d’affichage, pour vérifier que tout s’affiche toujours comme demandé.

## Étape 7 : obtenir un livre

Définissez une fonction permettant de demander un livre à l'utilisateur, et de le renvoyer à la fonction appelante. Attention à ne demander que les informations nécessaires selon que le livre est ou n’est pas dans une série.

Testez cette fonction, dans la fonction principale, en y faisant appel et en affichant ensuite le livre récupéré.

## Étape 8 : tenir compte du tri

Pensez au fait que la liste chaînée est triée comme suit :

**par ordre alphabétique sur les auteurs**

**⮱ par ordre croissant sur l’année de parution**

**⮱ par numéro de tome**

**Au lieu d’ajouter les instructions directement dans la fonction d’ajout, il est judicieux de définir une fonction de recherche sur base de ces 3 critères. En effet, elle sera également utile pour la suppression…**

**Écrivez une fonction de recherche qui reçoit le livre à rechercher (une variable de type** Livre**) et le pointeur** pBibliotheque.

**Cette fonction doit être utilisée non seulement pour déterminer si un livre est déjà présent ou pas, mais aussi pour déterminer la place à laquelle se trouve le livre recherché ou la place à laquelle on doit ajouter un nouveau livre.**

**Ainsi, elle doit permettre de récupérer**

**dans le cas d’une recherche qui permet de savoir si le livre existe ou pas,   
l’adresse du livre trouvé (**pLivre)**.**

**Dans le cas d’une recherche qui permet de savoir où insérer un nouveau livre,   
l’adresse du livre courant, celui qui doit suivre le nouveau livre (**pLivre**) et celle du livre qui précède le nouveau livre (**pLivreSauv**).**

**Une façon de faire est de renvoyer le pointeur sur le livre trouvé/courant (**pLivre**) et de passer le pointeur sur le livre qui précède (**pLivreSauv**) par adresse. Le prototype est le suivant.**

ChainonLivre\* livreRecherche(Livre livre, ChainonLivre\* pBibliotheque,

ChainonLivre\*\* pLivreSauv);

**Au niveau algorithmique, il faut commencer par rechercher l’auteur, ensuite l’année de parution (si l’auteur est présent) et enfin le numéro du tome (au cas où il y aurait eu deux tomes parus la même année).**

**Réfléchissez bien à la façon de procéder et vérifier l’ordre de vos conditions !**

**Pour tester cette fonction, ajoutez les instructions suivantes à votre la fonction principale :**

**déclarez une variable** livre **ainsi que les deux pointeurs nécessaires à l’appel de la fonction de recherche (**pLivre **et** pLivreSauv**),**

**affectez à la variable** livre **un livre qui fait partie de la bibliothèque (un des éléments du tableau de structure),**

**si ce n’est déjà fait, générez la liste chainée sur base du tableau de structures déjà déclaré et initialisé (*l’ordre des éléments à de l’importance puisque vous allez tester une recherche dans une liste chaînée triée*),**

**faites appel à la fonction de recherche en passant** livre **comme livre recherché (attention à la manière dont vous passez** pLivreSauv**),**

**affichez un message permettant de savoir si le livre est présent ou pas,**

**faites appel à la fonction d’obtention d’un livre pour garnir la variable** livre **de façon qu’elle ne corresponde à aucun livre de la bibliothèque,**

**faites à nouveau appel à la fonction de recherche en passant** livre **comme livre recherché,**

**affichez un message permettant de savoir si le livre est présent ou pas…**

**Maintenant que la fonction de recherche est prête, vous pouvez modifier la fonction d’ajout afin de tenir compte de l’ordre imposé. Réfléchissez à ce que vous faites : par exemple, il est inutile d’allouer de la mémoire si le livre est déjà dans la bibliothèque…**

## Étape 9 : supprimer un livre

Définissez une fonction permettant de supprimer un livre de la bibliothèque. Elle reçoit, entre autres, un livre. Elle renvoie un code de message indiquant le message à afficher le cas échéant.

Dans la fonction principale, procédez comme décrit précédemment pour tester la fonction que vous venez d'écrire.

## Étape 10 : un peu de convivialité

Arrivé à cette étape, vous avez quasi toutes les fonctions nécessaires pour faire un programme un peu plus convivial.

Voici un exemple de fonction principale. Les appels de fonctions en rouge font référence à deux fonctions que vous devez encore implémenter.

La fonction choixObtenu doit permettre à l’utilisateur de voir le menu et de choisir une option de ce menu. Sa signature est la suivante.

ChoixMenu choixObtenu(void);

La fonction libererListe doit libérer toute la mémoire allouée. Attention, chaque chainon de la liste doit être libéré… Faites attention à la façon de procéder !

void main(void) {

ChainonLivre\* pBibliotheque = NULL;

Livre livre;

CodeMessage codeMessage;

ChoixMenu choix;

choix = choixObtenu();

while (choix != QUITTER) {

switch (choix) {

case AJOUTER\_LIVRE:

livre = livreLu();

codeMessage = ajouterLivre(livre, &pBibliotheque);

afficherMessage(codeMessage);

break;

case SUPPRIMER\_LIVRE:

livre = livreLu();

codeMessage = supprimerLivre(livre, &pBibliotheque);

afficherMessage(codeMessage);

break;

case AFFICHER\_LIVRES:

afficherBibliotheque(pBibliotheque);

break;

}

choix = choixObtenu();

}

libererListe(&pBibliotheque);

}