INF1005C - PROGRAMMATION PROCÉDURALE
Travail dirigé No. 4
Les fonctions

Objectifs : Permettre à l'étudiant d'explorer la notion de fonctions.

Durée : Deux séances de laboratoire.

Remise du travail : Dimanche 2 juin 2019 avant 23 h 30.

Travail préparatoire : Lecture des exercices et rédaction des algorithmes.

Documents à remettre: Sur le site Moodle des travaux pratiques, on peut remettre l'ensemble des fichiers .cpp compressés des exercices dans un fichier .zip en suivant **la procédure de remise des TDs.**
Retard : Les retards ne sont pas tolérés, un retard méritera la note 0.

Correction : Les deux exercices seront corrigés.

Directives particulières

- Dans chaque programme vous pouvez ajouter d'autres fonctions à celles décrites dans l'énoncé, ainsi que d'autres structures (et sous-structures), pour améliorer la lisibilité et suivre le principe DRY (Don't Repeat Yourself). À chaque endroit où vous remarquez une duplication de code (vous avez écrit les mêmes choses plus d'une fois) et qu'il n'est pas possible de l'éliminer avec ce qui a été vu en cours, indiquez-le en commentaire.
- Il est interdit d'afficher directement ou indirectement dans les fonctions décrites si la description n'indique pas d'affichage.
- Respecter le guide de codage, les points pertinents pour ce travail sont donnés en annexe à la fin.
- N'oubliez pas de mettre les entêtes de fichiers (guide point 33), et pour chaque fonction (guide point 89).
- Ne pas oublier de donner la description IN/OUT pour chacun des paramètres.
- Compiler avec /W4. Lors de la compilation avec Visual Studio, votre programme ne devrait pas afficher d'avertissements (« warnings »).

Exercice 1 – Course automobile

Un fichier *champions.txt* contient des données concernant les champions du monde de Formule 1 durant quinze années.
Pour chaque pilote, le fichier réserve 4 lignes contenant:

- le nom du pilote,
- le pays, suivi du nombre de courses, suivi du nombre de victoires,
- la liste des années où il a été champion du monde,
- la liste des constructeurs avec lesquels il a été champion du monde.

Format précis du fichier :

- Lorsqu'il y a plusieurs entiers de suite, ils sont séparés par des espaces, et il n'y a jamais d'espace après le dernier entier (c'est directement la fin de ligne, soit '`\n`').
- Les textes peuvent contenir des espaces, les espaces au début d'un texte ne sont jamais importants.
- Les textes qui ne sont pas seuls sur leur ligne sont toujours suivi d'un point-virgule '`;`', il n'y a jamais d'espace après le '`;`' du dernier texte d'une ligne (c'est directement la fin de ligne, soit '`\n`').

Créez une structure `Pilote` qui contiendra les données d'un pilote à savoir :

- Son nom et son pays,
- Le nombre de courses et le nombre de victoires,

- Un tableau des années ou le pilote a été champion et le nombre d'années,
- Un tableau des constructeurs avec lesquels le pilote a été champion et le nombre de ceux-ci.

Note1 : Les paires ‹ tableau, nombre d'éléments › sont des bons candidats pour des sous-structures qui aideront la lisibilité du programme (donc vous devez créer de telles structures).
Écrivez les fonctions suivantes, pour lesquelles vous devez déterminer le type de la valeur de retour et la liste des paramètres :

1. `lireFichier()` qui stocke les données du fichier (donc des pilotes) dans un tableau de structures (voir structures décrites ci-haut incluant la Note1).
2. `afficherListeChampions()` qui affiche la liste des champions avec toutes leurs informations.
3. `listePilotesPays()` qui retourne un tableau contenant la liste des pilotes (incluant toutes leurs informations; encore, voir structures décrites ci-haut) qui appartiennent à un pays donné (passé en paramètre). La fonction retourne aussi le nombre total de victoires de tous les pilotes de ce pays (malgré que cette autre valeur de retour puisse réduire la lisibilité, le but est de vérifier que vous savez comment avoir plus d'une valeur de retour).
4. `meilleurPilote()` qui retourne le meilleur pilote de la période (celui qui a gagné le plus de courses selon les données disponibles).
5. La fonction `main()` doit proposer un menu qui permet à l'utilisateur de choisir la fonction à exécuter, entrer les paramètres nécessaires au clavier et voir les résultats à l'écran (tel qu'indiqué dans les directives du TD, aucune des autres fonctions décrites ci-dessus ne doit faire d'affichage). Le tableau est conservé entre les choix de l'usager, pour ne pas relire le fichier à chaque fois; en fait la fonction lireFichier ne devrait pas être dans le menu, mais être appelée une seule fois en début de programme.

Note2 : Un des noms des pilotes contient des accents, voir la « Foire aux questions » sur le site Moodle du cours pour savoir ce qu'il faut ajouter au programme pour qu'il soit affiché correctement.

Exercice 2 – Tableaux de nombres

On souhaite créer une petite bibliothèque de fonction pour la manipulation de tableaux de nombres entiers. Les tableaux à 2 dimensions ont un nombre de colonnes qui est connu à la compilation. On vous conseille fortement d'utiliser les *span* de la librairies GSL pour passer vos tableaux à vos fonctions. Allez lire le documents *Boucles sur intervalle, cppitertools et span* qui est disponible sur Moodle.

1. Écrivez une fonction qui retourne une valeur aléatoire entre 2 bornes (inférieure et supérieur).
2. Écrire une fonction qui remplit un tableau à une dimension de valeurs aléatoires (en faisant appel à la fonction du point 1); la taille du tableau à remplir peut être d'une taille différente d'un appel à l'autre de cette fonction, et on ne veut pas de taille maximale définie à la compilation.
3. Même question que ci-dessus, mais pour un tableau à 2 dimensions (le nombre de ligne peut varier, mais tel qu'indiqué dans la question, le nombre de colonnes est fixé à la compilation).
4. Écrivez une fonction qui reçoit un nombre entier et un tableau d'entiers à 2 dimensions et qui retourne le nombre d'éléments qui sont plus petits, égaux et plus grands à ce nombre.
5. Écrivez une fonction qui détermine si un tableau *A* est inclus dans un autre tableau *B* (une dimension). Si le tableau *A* est inclus dans le tableau *B*, la fonction retourne l'indice dans *B* où se trouve l'inclusion. Un tableau *A* est dit inclus dans un tableau *B* à la position *p* si $B_{p+i} = A_i$ pour tout indice *i* de *A*.
6. Écrivez un programme principal qui teste les 5 fonctions précédentes, affichant les résultats.

****

Annexe 1 : Points du guide de codage à respecter

Les points du **guide de codage** à respecter **impérativement** pour ce TD sont les suivants :
(voir le guide de codage sur le site Moodle du cours pour la description détaillée de chacun de ces points)

Nouveaux points depuis le TD3 :

2 : noms des types en UpperCamelCase
5 : noms des fonctions en lowerCamelCase
48 : aucune variable globale (les constantes globales sont tout à fait permises)
50 : mettre le & près du type
89 : entêtes de fonctions; y indiquer clairement les paramètres [out] et [in,out]

Points du TD3 :

3 : noms des variables en lowerCamelCase
21 : pluriel pour les tableaux (int nombres[];)
22 : préfixe n pour désigner un nombre d'objets (int nElements;)
24 : variables d'itération i, j, k mais jamais l
27 : éviter les abréviations (les acronymes communs doivent être gardés en acronymes)
29 : éviter la négation dans les noms
33 : entête de fichier
42 : include au début
46 : initialiser à la déclaration
47 : pas plus d'une signification par variable
51 : test de 0 explicite (if (nombre != 0))
52, 14 : variables vivantes le moins longtemps possible
53-54 : boucles for et while
58-61 : instructions conditionnelles
62 : pas de nombres magiques dans le code
67-78, 88 : indentation du code et commentaires
83-84 : aligner les variables lors des déclarations ainsi que les énoncés
85 : mieux écrire du code incompréhensible plutôt qu'y ajouter des commentaires