INF1005C – PROGRAMMATION PROCÉDURALE
Travail dirigé no 5
Enregistrements et fichiers binaires

Objectif : Permettre à l'étudiant de manipuler des fichiers binaires et des enregistrements, ainsi que de maîtriser les concepts d'accès direct et séquentiel.

Durée : Deux séances de laboratoire.

Remise du travail : Avant 23h30 le mardi 20 novembre 2018.

Travail préparatoire : Lecture des exercices et de la documentation fournie et rédaction des algorithmes.

Directives : N'oubliez pas les entêtes de fichiers ni, aux endroits appropriés, les commentaires dans le code. Vous devez ajouter un en-tête pour chaque fonction. Dans l'écriture de l'entête d'une fonction, ne pas oublier de donner la description IN/OUT pour chacun des paramètres. Il est interdit d'utiliser les variables globales, sauf celles en lecture seule (constantes). Suivez en tout temps le guide de codage.

Documents à remettre : sur le site Moodle des travaux pratiques, vous remettrez l'ensemble des fichiers .cpp et .hpp compressés dans un fichier .zip en suivant la procédure de remise des TDs.

Mise en contexte

Vous collaborez avec une équipe d'étudiants pour une compétition de drones dans laquelle vous devez détecter des cibles au sol avec votre drone. Votre avion survole de façon autonome la zone en prenant des photos avec la caméra embarquée qui transmet les images à une antenne parabolique au sol. L'antenne a bien sûr un angle d'ouverture limité, ce qu'il l'oblige à suivre l'avion en vol. Pour ce faire, l'ordinateur contrôlant l'antenne (un Raspberry Pi) reçoit la position GPS et l'altitude de l'avion.

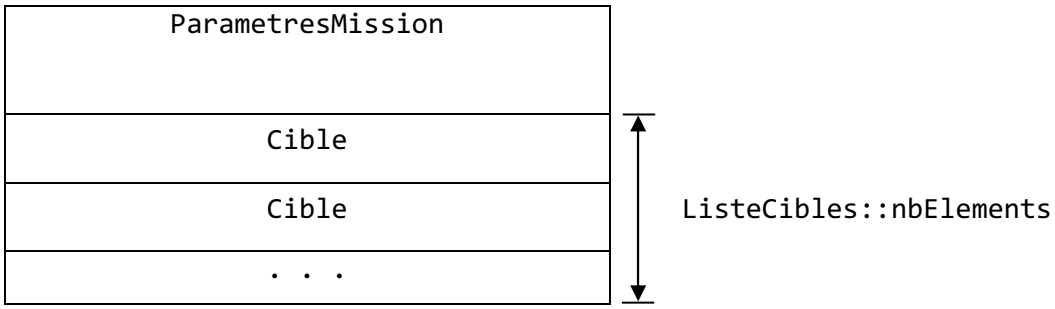
Pour tester le programme de détection de cibles, vous devez écrire un petit programme qui va lire les rapports de détection.

Format du fichier

Dans ce TD, on utilise un journal de détection écrit en binaire. Le fichier est composé d'une sorte d'entête (données qui apparaissent une seule fois au début du fichier) qui contient les paramètres de la mission (structure `ParametresMission` du TD) suivi d'un

tableau de cible (une séquence de structures `Cible` du TD). Tous les éléments sont contigus dans le fichier.

Voici, en résumé, le format du fichier binaire :



Noter que l'entête du fichier (`ParametresMission`) ne contient pas le nombre de cibles, mais qu'après avoir lu le fichier, le nombre de cibles lues devra être conservé dans le champ `nbElements` de `ListeCibles`.

Matériel fourni

Dans le fichier *CodeFourni.hpp* vous trouverez les déclarations des structures, constantes et variables qui vous sont fournies; leur implémentation se trouve dans *CodeFourni.cpp*. Vous devez écrire vos fonctions dans *CodeDemande.cpp* et mettre vos prototypes dans *CodeDemande.hpp*. Vous n'avez qu'à suivre les *TODO* dans chaque fonction. On vous donne aussi un fichier d'aide (la même documentation en deux formats, *Documentation.chm*, qui est un format page web indexé et compressé, et *Documentation.pdf*) qui contient toute la documentation du code fourni. Vous y retrouverez les descriptions détaillées des fonctions avec leur valeur de retour et paramètres ainsi que la description de ce que représentent les structures. Si jamais vous vous demandiez à quoi cela peut bien servir d'écrire des entêtes de fonctions, et bien en voici un bon exemple!

On vous donne aussi un journal de détection *Cibles.data* avec les cibles détectées.

Il y a plusieurs structures d'enregistrement avec lesquelles vous devez travailler. La structure `Position` donne une position GPS. `Cible` est une structure donnant la description d'une cible, c'est-à-dire sa position, son numéro et des observations à son sujet. La structure `ListeCibles` est une séquence de taille dynamique de `Cible`, c'est-à-dire avec une taille, une capacité et un tableau d'éléments. La structure `ParametresMission` représente l'entête du fichier binaire et `JournalDetection` représente la mission elle-même, avec des paramètres et une liste de données. Consultez

la documentation fournie pour plus de détails sur les structures, les constantes et les fonctions qui leur sont associées.

Attention : Pour la version .chm de la documentation, certaines configurations de Windows ne veulent pas ouvrir le fichier s'il est sur le réseau; dans le lab, copiez le fichier sur le disque Z: pour l'ouvrir (si vous ne voyez pas le logo de Polytechnique à l'ouverture de la documentation .chm, le fichier est incorrectement ouvert).

Présentation du travail à réaliser

Tout d'abord, lisez bien la documentation des structures fournies, car vous aurez à vous servir de celles-ci. On vous fournit entre autres des fonctions pour convertir les angles de degrés en radians et vice-versa et afficher à l'écran le contenu des différentes structures.

Vous avez neuf fonctions à implémenter, certaines plus courtes que d'autres. On vous fournit les squelettes dans *CodeDemande.cpp* et les prototypes dans *CodeDemande.hpp*. Pour chaque fonction, vous devez remplir les *TODO* qui vous sont donnés dans le code. Vous pouvez aussi ajouter des fonctions pour la lisibilité ou le principe DRY. Chaque fonction doit être testée (les *TODO* dans les fonctions `tests_partie1` et `tests_partie2`) à mesure. Vous devez ensuite écrire le `main` qui fera des opérations sur les cibles du fichier.

Il y a deux parties au TD. Les fonctions de la partie 1 peuvent être faites durant la première séance de laboratoire. Les fonctions de la partie 2 demandent l'utilisation d'allocation dynamique qui est une notion vue entre la première et la deuxième séance.

N'oubliez pas d'écrire votre documentation, c'est-à-dire les entêtes de fonctions selon le guide de codage.

Travail à réaliser – Partie 1

Ces fonctions ne demandent pas d'allocation dynamique et peuvent être faites durant la première séance.

- `ajouterCible()`
- `retirerCible()`
- `lireCibles()`

- `ecrireCibles()`
- `ecrireJournalDetection()`
- `ecrireObservation()`

Travail à réaliser – Partie 2

Ces fonctions demandent de faire de l'allocation dynamique et il est recommandé de les faire à la deuxième séance, donc après avoir eu le cours sur l'allocation.

- `allouerListe()`
- `desallouerListe()`
- `lireJournalDetection()`

Fonction principale du programme

On vous demande d'écrire un `main()` qui lit le journal, le met à jour et l'écrit dans un nouveau fichier.

Annexe 1 : Points du guide de codage à respecter

Les points du **guide de codage** à respecter **impérativement** pour ce TD sont les mêmes que pour le TD4 : (voir le guide de codage sur le site Moodle du cours pour la description détaillée de chacun de ces points)

Points du TD5 :

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