**INF3105 – Structures de données et algorithmes**
UQÀM, Département d'informatique

# TRAVAIL PRATIQUE #1 : SYSTÈME DE GESTION DE CARGAISON DE CAMIONS D'UN ENTREPÔT

## OBJECTIFS

- S'initier à la programmation en C++.
- Pratiquer l'encapsulation et concevoir des types abstraits de données.
- Gérer la mémoire.
- Implémenter et appliquer des structures de données simples en C++.
- Résoudre un problème simple.
- Analyser la complexité temporelle d'un algorithme.

## PROBLÉMATIQUE

Vous devez écrire un programme C++ nommé `tp1` qui gère un processus de cargaison des camions d'un entrepôt. Un grand entrepôt possède plusieurs bâtiments d'entreposage. Le programme reçoit en entrée le nombre total de boîtes à transporter, la capacité maximale d'un camion, les positions des bâtiments d'entreposage impliqués dans la cargaison courante ainsi que le nombre de boîtes disponibles à chaque point de cargaison. Pour commencer, il faut rechercher un point possédant **le plus grand nombre de boites**. Les coordonnées de ce bâtiment deviennent la position courante du camion. Pour chaque cargaison, le programme doit afficher les positions des bâtiments d'entreposage situés à la distance la plus proche de la position du camion, et le nombre de boîtes restantes aux points de service. On suppose qu'à partir de chaque édifice, vers le camion, le transport de toutes les boîtes disponibles est possible. Le nombre de boites à charger dans le camion ne doit pas dépasser la capacité maximale de celui-ci.

**Hypothèses simplificatrices.**

Le camion reste toujours à sa position spécifiée, ce sont plutôt les monte-charges qui se déplacent pour aller chercher et charger les boîtes. Un monte-charge peut livrer toutes les boîtes disponibles à la fois. On maximise le chargement du camion seulement au début en plaçant le camion à la position du point de service possédant le maximum de boîtes. Ensuite, on cherche à minimiser la distance parcourue par les monte-charges. Dans le cadre de la résolution de ce problème, vous avez besoin d'utiliser un algorithme de tri. Vous devez implémenter les deux algorithmes de tri : l'un simple et l'autre efficace, en créant les deux versions de votre solution.

**Exemple :**

Supposons qu'un camion doit transporter 15 boîtes de marchandise... La capacité du camion est mesurée en termes de boîtes et pour cet exemple, elle sera donc aussi de 15. Dans la liste des bâtiments de cargaison, il y a 4 points de services localisés aux coordonnées spécifiées avec le nombre de boîtes disponibles. Les données initiales et les résultats trouvés sont affichés plus bas.

```
15 15
2 (45.515399, -73.561996) 5 (45.5092,-73.5682)
8 (45.4383,-73.8205) 4 (45.4977,-73.714)
```

```
Truck position: (45.4383,-73.8205)
Distance:0          Number of boxes:0          Position:(45.4383,-73.8205)
Distance:10611.3    Number of boxes:0          Position:(45.4977,-73.714)
Distance:21193.7    Number of boxes:2          Position:(45.5092,-73.5682)
```

## STRUCTURE DU PROGRAMME

Pour bien amorcer ce travail, il est fortement recommandé de commencer avec le squelette de départ fourni dans tp1.zip. Vous pouvez modifier ces fichiers autant que vous désirez. Toutefois, vous devez préserver la syntaxe d'appel du programme et ses formats d'entrée et de sortie. Cela est nécessaire pour la correction automatique.

### Syntaxe d'appel du programme tp1

Le programme `tp1` doit pouvoir être lancé en ligne de commande avec l'une des deux syntaxes suivantes :

```
./tp1 camion_entrepot.txt
./tp1 < camion_entrepot.txt
```

où:

- le fichier `camion_entrepot.txt` contient un nombre de boîtes à charger, la capacité du camion et un état d'un entrepôt, soit une liste des bâtiments de cargaison (coordonnées) avec le nombre de boîtes disponibles.

Si `camion_entrepot.txt` est passé en argument (`argv[1]`), alors votre programme doit lire depuis le fichier `nomfichier` au moyen d'un flux de lecture de type `std::ifstream`. Sinon, votre programme doit lire dans l'entrée standard (`stdin`) au moyen du flux d'entrée `std::cin`. À noter que le squelette fourni implémente déjà cela.

Les résultats produits par votre programme doivent être écrits dans la sortie standard (`stdout`) à l'aide du flux de sortie C++ `std::cout`.

## Fichier d'entrée

Un fichier d'entrée (`camion_entrepot.txt`) est structuré de la façon suivante.

Sur la première ligne, on y retrouve un nombre de boîtes à transporter et la capacité du camion.

Les autres lignes contiennent une liste des 2 champs:

- nombre de boîtes au point de service;
- point de service spécifié par les coordonnées (longitude, latitude);

Pour faciliter la lecture (le *parsing*) au moyen d'un flux de lecture std::istream, des tabulations ou des espaces blancs sépareront les champs.

À titre d'exemple, voici un extrait du fichier d'entrée `camion_entrepot.txt` fourni.

```
15 15
2 (45.515399, -73.561996) 5 (45.5092,-73.5682)
8 (45.4383,-73.8205) 4 (45.4977,-73.714)
```

## Sortie

La recommandation de cargaison calculée par le programme doit être écrite dans la sortie standard (`stdout`) à l'aide du flux de sortie C++ `std::cout`.

```
Truck position: (45.4383,-73.8205)
Distance:0              Number of boxes:0               Position:(45.4383,-73.8205)
Distance:10611.3        Number of boxes:0               Position:(45.4977,-73.714)
Distance:21193.7        Number of boxes:2               Position:(45.5092,-73.5682)
```

## Le format de sortie

Première ligne : "`Truck position:`", espace, suivi par un point de service représenté par les coordonnées;

Sur chaque ligne suivante : "`Distance:`" suivi par la valeur de la distance calculée, un espace(tabulation) suivi par la chaîne "`Number of boxes:`", la valeur des boîtes restantes à chaque point de service impliqué, un espace(tabulation), la chaine "`Position:`" suivi par les coordonnées. Les espaces pourraient être remplacés par le symbole tabulation. Il faut afficher les points de services impliqués dans la cargaison.

Il est très important de respecter ce format de sortie, car des scripts de correction seront utilisés.

Durant le développement, il est recommandé d'utiliser le flux standard d'erreur C++ `std::cerr` pour afficher des messages de débogage et les messages d'erreurs. Ainsi, si vous oubliez d'enlever ces messages de débogage, ces derniers ne seront pas redirigés dans le fichier de sortie.

L'exécution de la commande :

```
./tp1 camion_entrepot.txt
```

doit produire le résultat (`res+.txt`) :

```
Truck position: (45.4383,-73.8205)
Distance:0              Number of boxes:0               Position:(45.4383,-73.8205)
Distance:10611.3        Number of boxes:0               Position:(45.4977,-73.714)
Distance:21193.7        Number of boxes:2               Position:(45.5092,-73.5682)
```

# CONTRAINTES

## Librairies permises

Vous devez implémenter et utiliser vos propres structures de données. Pour l'instant, l'utilisation des conteneurs de la librairie standard de C++ (*Standard Template Library*) n'est pas permise. Ce sera permis plus tard dans le cours.

## Environnement de développement

Relisez les Politiques et les directives sur les outils informatiques dans le cours INF3105.

## Taille des équipes

Vous pouvez faire ce travail en équipe de 1 ou 2. Toutefois, tous les membres de l'équipe doivent contribuer à l'ensemble du travail et non à seulement quelques parties. Le travail d'équipe vise à favoriser les discussions et l'entraide. Le travail d'équipe ne vise pas à réduire la tâche. Ainsi, se diviser la tâche en deux n'est pas une méthode de travail d'équipe appropriée dans ce cours. Tous les membres de l'équipe doivent être en mesure de comprendre et d'expliquer l'ensemble du travail. La participation inadéquate d'une étudiante ou d'un étudiant peut être considérée comme du plagiat. Le professeur et le correcteur pourront sélectionner quelques équipes au hasard afin de vérifier que tous les membres sont capables d'expliquer l'ensemble du travail.

---

# ANALYSE TEMPORELLE

Vous devez réaliser deux types d'analyses de la complexité temporelle des algorithmes pour les deux versions de votre solution avec les différents algorithmes de tri : une analyse expérimentale (empirique) et une analyse théorique (asymptotique). Les résultats de ces analyses doivent être inclus dans le rapport.

# REMISE

Vous devez remettre électroniquement le TP1 **au plus tard le 26.02.2024 à 23h59** au plus tard. La remise papier doit être effectué soit en classe au professeur au début du cours qui suit la date de remise ou être placé dans la chute à travaux du Département d'informatique (PK-4151).

### Remise électronique

Vous devez remettre un seul fichier archivé contenant tous vos fichiers sources, incluant un fichier Makefile, via Moodle.

Vous pouvez soumettre votre TP autant de fois que vous voulez. Seule la dernière soumission sera considérée.

### Remise papier

La partie papier est constituée du/d'un(e)/de:

1. **Formulaire de remise imprimé** [ PDF | OpenDocument | Word ], **dûment rempli et signé.**
2. **Auto-évaluation** indiquant si votre programme fonctionne correctement, partiellement ou aucunement.
3. **Analyse de la complexité temporelle (pire cas) en notation grand O**
    - La complexité temporelle doit être exprimée en fonction de la taille du problème :
        - *n* indique nombre de bâtiments impliqués dans la cargaison;
4. **Code source.**
    - Imprimez tout votre code source.
    - L'impression doit être faite à l'aide d'un éditeur de texte mettant la syntaxe C++ en évidence, comme GEdit ou Notepad++.
    - N'imprimez pas depuis un logiciel de traitement de texte comme Word.

- Le code source doit être présenté dans un ordre convenable. Par exemple, un fichier d'entête (.h) devrait venir immédiatement le fichier source (.cpp) correspondant.

Brochez le tout. Évitez les trombones. Il n'est pas nécessaire d'insérer le tout dans une grande enveloppe.

Vous pouvez aussi remettre la partie papier dans la chute à travaux tout juste à gauche de la première porte du PK-4151. Vous pouvez aussi remettre au début d'un cours.

Les correcteurs vous feront des commentaires constructifs directement sur la partie papier qui vous sera ensuite retournée.

# ÉVALUATION

Ce travail pratique vaut 10% de la note finale.

## Grille de correction

| Critère | Description | Pondération |
|---|---|---|
| A. | **Respect des directives pour la remise**<br><br>• Fichiers sources seulement (ex.: .h, .cpp, Makefile). Aucun fichier source manquant. Aucun fichier binaire (.o, exécutable). Aucun fichier test.<br>• Remise par Moodle. Pas de remise par courriel.<br>• Compilable avec `make` sans modification.<br>• Exécutable sans modification. | / 1 |
| B. | **Appréciation générale**<br><br>• Structure du programme + Qualité du code :<br>  ◦ Découpage du programme (tout n'est pas dans la fonction `main()`).<br>  ◦ Choix des types de données; identificateurs (noms) significatifs, lisibilité du code, pertinence des commentaires; etc.<br>  ◦ Justesse de l'usage du mot clé `const`, des références (&) et des pointeurs (*).<br>• Encapsulation :<br>  ◦ Respect des principes de l'abstraction;<br>  ◦ Cachez le maximum de la représentation des objets en rendant un maximum d'attributs privés;<br>  ◦ Évitez autant que possible les bris d'abstraction.<br>  ◦ Utilisation appropriée des modificateurs d'accès `public`, `protected` et `private`, et du mot clé `friend`, etc.<br>• Gestion de la mémoire : | / 1 |

<table>
<tr><td></td><td>
<ul><li>Toute la mémoire allouée dynamiquement doit être correctement libérée au moment approprié et avant la fin du programme.</li></ul>
</td><td></td></tr>
<tr><td>C.</td><td>Fonctionnement correct.<br>Le programme produit les bonnes recommandations. Une recommandation non optimale est considérée mauvaise même si elle est très proche de l'optimale.<br>L'efficacité n'est pas directement évaluée. Cependant, l'efficacité peut être indirectement évaluée lorsqu'un programme ne parvient pas à produire des résultats dans des délais raisonnables. Votre programme doit également consommer une quantité de mémoire raisonnable.</td><td>/ 5</td></tr>
<tr><td>D.</td><td>Exactitude de l'auto-évaluation.
<table>
<tr><td></td><td colspan="4">Vous déclarez que votre programme fonctionne ...</td></tr>
<tr><td></td><td>Correctement</td><td>Partiellement</td><td>Aucunement</td><td>(Déclaration absente)</td></tr>
<tr><td>À la correction, votre programme fonctionne correctement</td><td>0.5</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>À la correction, votre programme fonctionne partiellement</td><td>0.25</td><td>0.5</td><td>0</td><td>0</td></tr>
<tr><td>À la correction, votre programme ne fonctionne aucunement</td><td>0</td><td>0</td><td>0.5</td><td>0</td></tr>
</table>
Attention : les tests fournis ne sont pas nécessairement ceux qui seront utilisés pour la correction. De nouveaux tests pourront être utilisés.</td><td>/ 0.5</td></tr>
<tr><td>E.</td><td>Analyse de l'algorithme.<br><ul><li>Complexité temporelle en notation grand O.</li><li>Ordre de grandeur simplifié. Ex: O(2n) ==> O(n).</li><li>Cheminement et justification.</li></ul></td><td>/ 2</td></tr>
<tr><td>F.</td><td>Efficacité</td><td>/ 0.5</td></tr>
<tr><td></td><td>Total :</td><td>10/ 10</td></tr>
</table>

Pour les cas problématiques, jusqu'à 2 points peuvent être retranchés pour la qualité de la présentation.