TP2 C++

Objectif : Ce TP a comme objectif la réalisation d'une application simple nécessitant quelques classes. Comme pour le premier TP C++, les notions suivantes seront étudiées : interface de classe et de réalisation de classe, la syntaxe du langage C++, la spécification et la conception d'une application, les tests unitaires et les tests fonctionnels.

Sujet - Cahier des Charges

Les routes de la ville de Lyon sont équipées de capteurs fournissant des informations sur le trafic routier. Actuellement nous disposons de 1500 capteurs, chaque capteur ayant un identifiant numérique. Les capteurs produisent l'état du trafic chaque minute, sous la forme d'un caractère représentant une couleur (vert, rouge, orange, noir) : 'V', 'J', 'R', 'N'. Nous vous demandons d'implémenter une application avec les fonctionnalités suivantes :

1. lecture des valeurs produites par un capteur ;
2. pour un capteur donné, trouver le pourcentage de temps qu'il passe dans un état de trafic particulier ;
3. pour la ville de Lyon, trouver les statistiques de trafic pour un jour de la semaine ;
4. pour la ville de Lyon, afficher les pourcentages d'embouteillages par jour de la semaine et par heure (un embouteillage correspond aux états 'N' ou 'R') ;
5. (*) pour un ensemble de segments connus formant un trajet, et pour une plage horaire connue, trouvez le moment du départ qui permet de minimiser le temps passé dans un embouteillage.

Les données de test se situent dans l'intervalle mai 2015 - septembre 2015.

Description des entrées/sorties

L'interface du programme sera réalisée en ligne de commande.

Lecture d'une valeur

La lecture d'une valeur de capteur est simulée avec une commande respectant la syntaxe :

ADD <id> <yyyy> <mm> <dd> <h> <m> <d7> <trafic>

où <ID> représente l'identifiant du capteur et <trafic> l'un des caractères 'V', 'J', 'R' ou 'N'. <yyyy> <mm> <dd> <h> <m> décrivent le timestamp (année, mois, jour, heure, minute). <d7> représente le jour dans la semaine (entre 1 et 7), pour vous éviter de la calculer.

Statistiques de trafic pour un capteur

La commande pour afficher les statistiques de trafic pour un capteur particulier :

STATS_C <idCaptor>

La réponse aura la forme :

V <value>%
J <value>%
R <value>%
N <value>%

où <value> est un nombre entier, représentant le pourcentage de temps passé dans un état particulier, pendant toute la durée de fonctionnement (durée active) d'un capteur. Les données arrivent en ordre chronologique par capteur. Un capteur est considéré comme actif chaque minute où il a émis une donnée.

Embouteillages par jour de la semaine et par heure

La commande pour afficher les statistiques sur les embouteillages par jour de la semaine et par heure :

JAM_DH <D7>

où <D7> identifie le jour de la semaine (1 à 7).
La réponse aura la forme :

<D7> 0 <value>%
<D7> 1 <value>%
...
<D7> 23 <value>%

où <value> est un nombre entier.

Statistiques de trafic pour un jour de la semaine

La commande pour afficher les statistiques de trafic pour les jours de la semaine :

STATS_D7 <D7>

où <D7> identifie le jour de la semaine (1 à 7).
La réponse aura la forme :

V <value>%
J <value>%
R <value>%
N <value>%

où <value> est un nombre entier.

Moment de départ optimal

Pour un ensemble de segments connus formant un trajet, et pour une plage horaire $<H_{start}>$-$<H_{end}>$, la commande pour afficher le moment de départ optimal (qui minimise le temps passé sur le parcours) :

OPT <D7> <H_start> <H_end> <seg_count> <seg_1> ... <seg_n>

où <D7> identifie le jour de la semaine (1 à 7), $<seg_{count}>$ est le nombre de segments formant le trajet et $<seg_1>$, ..., $<seg_n>$ les ID's de chaque segment. Les segments sont donnés dans l'ordre, et le temps de parcours de chaque segment est de 1 minute (état 'V'), 2 minutes (état 'J'), 4 minutes (état 'R'), 10 minutes (état 'N').
Le résultat :

<D7> <H> <M> <value>

où <value> représente le temps de parcours estimé (en minutes).

Fin de l'application

L'application se termine à l'apparition de la commande :

EXIT

Contraintes

Les seules classes que vous pouvez réutiliser sont celles concernant les entrées sorties (via *iostream*), la gestion des chaînes de caractères (via *<string>*) et éventuellement des fonctions pour la gestion du temps.
Quelques informations sur les tests que nous allons faire sur votre application : nous utilisons au maximum 1500 capteurs et 20.000.000 événements (opérations ADD).
Les événements arrivent en ordre chronologique.

Documents fournis et outils

Quelques exemples d'entrées/sorties vous seront proposés (via moodle2.insa-lyon.fr). Nous vous conseillons de développer votre application avec Eclipse (commande : eclipse-cpp).

Livrables attendus

1. Un document de spécification et de conception, format pdf, contenant les spécifications complètes de vos classes (principales définitions et choix généraux, ≤ 6 pages).
2. Vos codes sources (*.h, *.cpp, makefile) à déposer sur /sources.
3. Les tests que vous avez conçu et implémentés, à déposer sur /tests. Un document décrivant (d'une manière très succincte, ≤ 2 pages) ces tests.

Tous ces livrables seront déposés dans un fichier .zip sur moodle2.insa-lyon.fr, dans l'activité associée au cours *Algorithmique, Programmation Orientee Objet - C++*. Le fichier aura le nom B3XYY.zip, où X est votre groupe, YY votre numéro de binôme.

Notation

La note associée à ce TP sera dépendante de :

- (50%) la correction de votre code ; la plate-forme http ://servif-algo.insa-lyon.fr/domjudge sera utilisée pour vous permettre de valider automatiquement votre solution ; la correction du code est une condition nécessaire pour valider le TP. La fonctionnalité (*) n'est pas obligatoire pour valider le TP, mais elle est nécessaire pour avoir la note maximale (un TP correct sans cette option sera noté sur 16). Attention : chaque envoi d'une solution incorrecte sur la plate-forme vous pénalise de 0,5 points ! Il est important de bien tester votre solution avant de l'envoyer.
- (10%) les performances de votre code ; des tests seront réalisés sur des volumes de données importantes, et dans ce cas la performance de vos algorithmes va jouer un rôle important ; ces tests seront réalisée sur la plateforme précédente, en utilisant de vraies données ;
- (20%) le document de spécification et de conception ;
- (10%) la procédure de non-régression que vous avez mis en place (tests) ;
- (10%) le respect d'un guide de style pour le code C++.