TP2 C++

B3111 : Edern HAUMONT et Nicolas SIX

Mercredi 11 novembre 2015

Table des matières

1 Introduction

1.1 Choice of the data structure

Our application must support 20 million events from 1500 sensors. Consequently, we chose not to store event themselves as they arrive. However, at the launch of our program, several static arrays of entire values are created. Their indexes correspond to the information used to anwer to the user querries, for instance the sensor number, the hour,etc. Their last dimension corresponds to a sensor color. So when an event is added, we just increment one cell in each array.

2 Classes

2.1 DataHandler

All data additions and request are managed by the class DataHandler which contain itself the 4 data arrays.

2.1.1 Constants

This class use differents constants, first there are one error code used when the trafic char is not one of the four expected : 'V','J','R','N'

```
const unsigned int ERROR_INVALID_TRAFIC_UCHAR = 201;
```

DataHandler also use constants to define the size of the arrays used to store all the datas.

```
const int NUMBER_OF_COLORS = 4;
const int NUMBER_OF_SENSORS = 1500;
const int NUMBER_OF_MINUTES = 1440;
const int NUMBER_OF_HOURS = 24;
const int NUMBER_OF_DAYS = 7;
```

2.1.2 Atributs

To make our code more readable we decided to use the following type def in the DataHandler Class.

```
typedef unsigned int uint;
```

The argument used in this class are the following :

```
IdHash idHash;
uint sensors[NUMBER_OF_SENSORS][NUMBER_OF_COLORS];
uint days[NUMBER_OF_DAYS][NUMBER_OF_COLORS];
uint daysAndHours[NUMBER_OF_DAYS][NUMBER_OF_HOURS][NUMBER_OF_COLORS];
unsigned char *daysAndMin[NUMBER_OF_DAYS][NUMBER_OF_MINUTES][NUMBER_OF_COLORS];
```

idHash is an instance of our IdHash class, which is detailled latter. It's an hash tab that we use to link the sensors id and the position in our static tabs. sensors, days, daysAndHours and daysAndMin are four static array used to store statistics which will be used in our stats methods. The fourth array is a four dimantion one with the last one, coreponding to the sensors id, is alocated using a new in the construtor.

2.1.3 Public Methods

2.1.3.1 addData

```
int addData(const Data &data);
int addData(const char &trafic ,const uint &min,const uint &hours,\
        const uint &id ,const uint &day7);
```

description : Method used to increment corresponding cells of all arrays. Complexity : O(1).

Contract : The values given must be ritghtly build : min shoult be between 0 and 59, hours between 0 and 23 and day7 between 0, for monday, and 6, for sunday. To work properly trafic must be set to one of : 'V','J','R','N'.

2.1.3.2 sensorStats

```
int sensorStats(uint id);
```

description : Prints a sensor statistics using the array sensors. Complexity : O(1).

Contract : The id given in parameter must corespond to a sensors with an already added id else the stats will use the first added sensor.

2.1.3.3 jamStats

```
int jamStats(uchar day7);
```

description : Prints jam statistics by hour using the array daysAndHours. Complexity : O(1).

Contract : day7 must be between 0 and 6 ($NUMBER\_OF\_DAYS - 1$).

2.1.3.4 dayStats

```
int dayStats(uchar day7);
```

description : Prints a week day statistics using the array days. Complexity : O(1).

Contract : day7 must be between 0 and 6 ($NUMBER\_OF\_DAYS - 1$).

2.1.3.5 optimum

```
int optimum(uchar day7, uint bégginHours, uint endHours,\
                uint idTab[], uint tabSize);
```

description : optimum look for the best departure time in a given interval. Complexity : $O(tabsize * (endHours - beginHours))$.

Contract : day7 must be between 0 and 6 ($NUMBER\_OF\_DAYS - 1$). begginHours and endHours must be between 0 and 23 with begginHours smaller than endHours. idTab must be an array of size tabSize filled with already added sensors' id (if an id is not added the first added sensors will be used).

To compute the optimum departure time this methode consider that the time taken to go throuht a sensors is the most probable one at this time of the day acording to the added data. In case of equality on trafic information, the quiker will be used (optimistic).

2.2 IHash

2.2.1 Constants

2.2.2 Atributs

2.2.3 Public Methods

2.2.3.1 addId

```
unsigned addId(const unsigned & id);
```

description :

Contract :

2.2.3.2 getTabId

```
unsigned getTabId(const unsigned & id) const;
```

description :

Contract :

3 tests