SIVANESAN Shivamshan 21209392

Advanced C++ Project: RATP Application

1. Create a file with a main function, and instantiate an object of your class. Does the code compile? Why doesn't it compile?

The code doesn't compile, this is due to the fact that the Generic station parser has a pure virtual function read_stations which needs to be implemented in the child class before thus causing the compilation error.

2. Override the Generic station parser::read stations function: In the class declaration, write the prototype of this protected function by adding the override keyword at the end. Implement this function?

The read_stations functions does the following things :
- We verify the filename could be opened or else we throw the runtime error
- We start reading the elements of the file starting from the second line,
- We then retrieve the strings that are stocked inside a string vector. For storing these elements in the station hash map, we use the station id as the key and use the struct Station to store the information of each station.

3. In your main function, instantiate your class and call the stations method of the Grade class. You use the static object provided in this class, depending on the database used?
Database used is the small database.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ g++ -Wall -Wextra -Werro
r -pedantic -pedantic-errors -O3 -g -std=c++11 main.cpp station.hpp station.
cpp "Grade 2019 g++ 8.2.0 UNIX.o" -o main.exe
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe
==========================> Grade 1 <==========================
Stations: seems ok
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$
```

4. Change your class to inherit from the parent class Generic connection parser.
Does the code compile, why?

The code doesn't compile due to the presence of a pure virtual function in the parent class which need to be implemented in the child class in order for it to compile.

5. Implement the read connections function.

For the read connections, we do the same 2 steps as the read_stations. Since the connections hash map is an unordered map with the key as the starting stations and the values is another unordered map. The key of this unordered map is the ending stations and the values is the time to go from the starting to ending stations.
We transform each string to integer before storing it inside the connection hash map.

6. Instantiate your class and call the connections method of the Grade class.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ g++ -Wall -Wextra -Werro
r -pedantic -pedantic-errors -O3 -g -std=c++11 main.cpp station.hpp station.
cpp "Grade 2019 g++ 8.2.0 UNIX.o" -o main.exe
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe
============================> Grade 1 <============================
Stations: seems ok
============================> Grade 2 <============================
Connections: seems ok
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$
```

7. Change your class to inherit from the Generic mapper parent class. Does the code compile, why?

No, this due to the fact the the parent contains two pure virtual function which needs to be implemented in the child class.

8. Implement the compute travel function, having as input arguments numbers corresponding to the station identifiers. This function returns a vector of std::pairs, containing the station ID and the associated cost of the connection.

The algorithm works in the following manner :
- Find out if the departure and arrival stations exist in the stations present in the hash map
- We assume that all nodes are not yet visited and that they all have equal start time values INF except the start node which is 0.
- As long as all nodes are not visited, we take the smallest node (at the beginning it will be the start node)
- This node will then be removed from the unvisited nodes and will be put in the visited node.
- For the chosen parent node we take their child nodes:
    - We check that if this child node is not already visited
    - If this child node has a time lower than the current distance
    - Then the child node in question takes the new smaller distance
    - We also keep in memory the path(prev) the parent node we took to get to the child node (prev[child node] = parent node)
    - Then we add the accumulated time to go from this parent node to the child node in question
- Finally we use the prev(unordered map) to take the stations that allow us to leave from the starting station to the arrival station

To the stations nodes which will get us to the end stations, we use the unordered map by looping from the end station until we get to the start station.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ g++ -Wall -Wextra -Werro
r -pedantic -pedantic-errors -O3 -g -std=c++11 main.cpp station.hpp station.
cpp -o main.exe
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe
Station : 3 Accumlated time : 0s
Station : 8 Accumlated time : 103s
Station : 10 Accumlated time : 270s
```

9. Implement the compute and display travel function, allowing to display the path calculated by the

function to connect the two stations you want to connect. The notation depends on the clarity of the display, as the client must easily follow clear instructions to travel.

This function uses the compute_travel function from above and prints out the chosen shortest path.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe
Best way from C ( line Ligne de la station C) to J ( line Ligne de la statio
n J) is:
Walk to C, line Ligne de la station C
Take line Ligne de la station H,  Ligne de la H (Terminus undefined -> Termi
nus undefined) - Aller
From C to H (103 secs)
Walk to H, line Ligne de la station H
Take line Ligne de la station J,  Ligne de la J (Terminus undefined -> Termi
nus undefined) - Aller
From H to J (167 secs)
After 270 secs, you have reached your destination!
```

10. Instantiate your class and call the dijkstra method of the Grade class, with the boolean argument set to false.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ g++ -Wall -Wextra -Werro
r -pedantic -pedantic-errors -O3 -g -std=c++11 main.cpp station.hpp station.
cpp "Grade 2019 g++ 8.2.0 UNIX.o" -o main.exe
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe "data/station
s.csv" "data/connections.csv" 3 10
============================> Grade 1 <============================
Stations: seems ok
============================> Grade 2 <============================
Connections: seems ok
============================> Grade 3 <============================
First tests
Starting id unknown
```

For the test with dijikstra, I tried to test the program with the small database but it threw an error stating that the starting station id is unknown even though the station number exist within the stations hash map.

11. Implement an overload of functions that estimate the minimum path to connect the stations via their names, and not their identifiers. It would be nice to be error resistant (case sensitive, misspelling, etc) when the user enters the names.

We overload the compute_travel function which in this case takes strings as arguments. First, we verify that the stations names exist within the hash map if not we throw an error. If the stations exist, we then get it’s id associated to the station in question and finally we use the compute_travel function take integers as arguments which means we don’t need to re implement the compute travel function again.

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ g++ -Wall -Wextra -Werro
r -pedantic -pedantic-errors -O3 -g -std=c++11 main.cpp station.hpp station.
cpp -o main.exe
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe data/stations
.csv data/connections.csv C J
Best way from C ( line Ligne de la station C) to J ( line Ligne de la statio
n J) is:
Walk to C, line Ligne de la station C
Take line Ligne de la station H,  Ligne de la H (Terminus undefined -> Termi
nus undefined) - Aller
From C to H (103 secs)
Walk to H, line Ligne de la station H
Take line Ligne de la station J,  Ligne de la J (Terminus undefined -> Termi
nus undefined) - Aller
From H to J (167 secs)
After 270 secs, you have reached your destination!
```

This is the output given for the large database with the ids 1722 and 2062 :

```
shiv@shiv-PC:~/Desktop/Prog_c/Projet_Sivanesan_Cpp$ ./main.exe data/s.csv data/c.csv 1722 2062
Best way from Saint-Lazare ( line 3) to Bastille ( line 1) is:
Walk to Saint-Lazare, line 3
Take line 14, (SAINT-LAZARE <-> OLYMPIADES) - Aller
From Saint-Lazare to Saint-Lazare (180 secs)
Walk to Saint-Lazare, line 14
Take line 14, (SAINT-LAZARE <-> OLYMPIADES) - Aller
From Saint-Lazare to Madeleine (122 secs)
Walk to Madeleine, line 14
Take line 14, (SAINT-LAZARE <-> OLYMPIADES) - Aller
From Madeleine to Pyramides (134 secs)
Walk to Pyramides, line 14
Take line 14, (SAINT-LAZARE <-> OLYMPIADES) - Aller
From Pyramides to Châtelet (116 secs)
Walk to Châtelet, line 14
Take line 1, (CHATEAU DE VINCENNES <-> LA DEFENSE) - Retour
From Châtelet to Châtelet (180 secs)
Walk to Châtelet, line 1
Take line 1, (CHATEAU DE VINCENNES <-> LA DEFENSE) - Retour
From Châtelet to Hôtel de Ville (104 secs)
Walk to Hôtel de Ville, line 1
Take line 1, (CHATEAU DE VINCENNES <-> LA DEFENSE) - Retour
From Hôtel de Ville to Saint-Paul (Le Marais) (113 secs)
Walk to Saint-Paul (Le Marais), line 1
Take line 1, (CHATEAU DE VINCENNES <-> LA DEFENSE) - Retour
From Saint-Paul (Le Marais) to Bastille (150 secs)
Walk to Bastille, line 1
Take line 1, (CHATEAU DE VINCENNES <-> LA DEFENSE) - Aller
From Bastille to Bastille (0 secs)
After 1099 secs, you have reached your destination!
```

As we can see that some times the the same line appears multiple times, this is to indicate we take the same line but to different stations, the different stations are indicated through the ‘from’.