Logo, company name

Description automatically generated

**C++ Individual Project Reflection**

CS 313: Intermediate Computer Programming

Cyril Kujar | 62012024

Instructors: Dr. Robert Sowah & David Ebo Adjepon Yamoah

Faculty Interns: Owusu-Banahene Osei

28th November 2022

Initial thoughts on how to approach the route-finding problem and solve this challenge came with a variety of approaches in mind. Since we had previously tackled this problem but using java, we had an idea of a general procedure we could use to solve this problem. However, since C++ is a whole new language, we realized that the way in which we implemented certain things in Java might not work out the same way in this new language.

We first began by reading the data files provided and storing each of its contents in an unordered map. The reason for this was because we needed a way to access the data in the files while the program was running so storing them temporarily in these structures helped us do this. For each file read, a corresponding object was created with its parameters being all the necessary values of that object which may be needed throughout the program. For example, an airport object would contain its name, city, IATA code, latitude, longitude, etc. Since our program needed to lookup certain values such as the airports in a particular city quickly, we thought an unordered map would be useful since it supports the easy lookup of key-value pairs.

The keys for the maps were carefully chosen to ensure that at each point in the program, there would be a way to access the data needed quickly. For example, a key for the airports map was the city name and country name with its value being a list of airports in that city country. This helped to extract all start and destination locations when the file input is entered. Another key used was the unique airport IATA code with its value being the associated airport object. This helped to get the airport associated with the IATA codes derived when a path was found from one point to another which in turn was used to calculate the distance between points using the airports’ latitude and longitude. The sections discussed so far were mostly worked on by Bernd. In the routes file, we tracked all the flights that leave a particular airport by storing the source airport IATA as a key and a list of route objects containing the necessary details which leave the source airport as values. This structure in a way represented a graph/adjacency list of routes. We also kept track of a list of all flights that go from one location to another in order to determine the flights that can be taken once a valid path is found.

After storing all the data in respective data structures, to find a valid path from one point to another, we implemented the breadth-first search algorithm to search through the graph of routes and find the goal or destination. Once the goal was found, the path was reconstructed by tracing the parent of each node in the path found through a parent map. For each path found by the program, the total distance covered by that path was computed using the haversine formula and these distances were compared to obtain the optimal path. These sections were mostly worked on by me.

In implementing this solution, a few challenges were faced due to the different syntax in C++. We were not very familiar with the various libraries and data structures in the language along with their functionalities. This led us to do a lot of research into how these libraries and structures work in order to become conversant with their functionalities. Overall, this challenge was a great learning experience and allowed us to better grasp C++ concepts around OOP, file reading and the syntax in general.