Kwaku Agyako Osafo

CS313: Intermediate Computer Programming, Fall 2022

Reflection on C++ Project

Dr. Ebo Adjepon, Dr. Robert Sowah

29th November, 2022

If there’s one thing I learnt from doing this project, it’s this, line-by-line code translation is a bad idea.

C++ is different, to say the least. It certainly does not have python and java’s simplicity and functionalities. After all, it’s meant to give the programmer the reins so that he can control stuff like memory usage. This project was harder than I predicted. I thought we (Kirklyn and I) would just have to google the equivalent of Java functionalities in C++, only to find out those functionalities do not even exist in C++. For the Java project, I put object-orient programming to good use. My approach relied heavily on representing almost everything as an object. Kirklyn and I tried doing that in C++ and received a rude awakening. After two days of trying to crack our (me and Kirkyn’s) code, I conceded to the reality that had been staring us in the face. We had to rethink our approach.

With our first one, using four classes proved to be a major hurdle. These were Airport, City, Airline and File. We had Airline types in the Airport class and vice versa. C++ was having none of that. The compiler coughed out an “undeclared file type error.” Again, the changes made by our setter functions were not permanent so the data read from files could not be permanently stored in the member variables of the various objects. The only time we could register permanent changes was when we used a constructor. Passing by reference did not make the problem go away.

Our new approach involved the use of only one class, an airport class. Its member variables were *airportCode, airportID, airportName, parentAirportAirlineCode,* and *parentAirport.* All these variables were of type string. In the previous approach, we kept a lot of data that was irrelevant to the output. Examples were the airport’s name and the country it was situated in. We had two constructors. The first initialized *airportCode, airportID, parentAirportAirlineCode,* and *parentAirport.* The second initialized airport ID and airport code. For the sake of encapsulation, we made these variables private. This class’s functions were getters for these variables, and a path function for getting the airports visited and the airlines used during the flight. Everything else, mainly reading from the files and the search algorithm went into the main function.

The “airlines.csv” file was redundant, so we did not bother reading from it. We defined three functions in the main function. *createCitiesAndTheirAirportsMap, createAirportsMap, createRoutesMap, getStartCityAndDestinationCity, foundDestination,* and *search.* We had an *airportsStream* and *routesStream* variable of type *fstream* for reading from the airports and routes file. We had *startCity* and *destinationCity* variables of type string, a *destinationCityVector* variable that was empty only when the destination had not been found, and unordered maps called *routesMap*, *airportsMap* and *citiesAndTheirAirports.* One of our main goals for this project was to make our code readable and understandable. Using descriptive variable names helped with this. The *citiesAndTheirAirports* had a city name as its key and a vector of airports as its value. The *routesMap* had an airport code as its key and a vector of destination airports as its value. The *airportsMap* had an airport ID as its key and an airport object as its value. For our search, we used breadth-first. The comments and docs in the code will provide a better understanding of our process.

Yes, this project introduced me and Kirklyn to the demon of C++ but it was a good learning experience overall. He and I did not work independently of each other. No part of the code is his and neither is any part mine. Everything is ours.