Harjoitustyö alkoi aluksi katsomalla harjoitustyön työohje. Työohjeesta päättelin, että minkälainen rakenne olisi hyvä ja toimiva kaikille funktioille. Myös pyrin katsomaan toteutattivia funktioita, että miten saan paljon käytettäville funktioille mahdollisimman hyvän tietorakenteen. Päädyin kahteen eri ”struct” rakenteeseen. Erikseen on pysäkkien ”struck”, joka sisältää pysäkin nimen, koordinaatin ja alueen mihin se kuuluu. Huom. aluetta ei välttämättä ole olemassa, joten se on älykäs osoittaja. Myös alueilla on oma ”struct” rakenne, jossa on alueen nimi ja tämän vanhempi eli parent, joka on toteutettu älykkäillä osoittimilla. Rakenteessa on myös lapsia ”set” rakenteessa älykkäillä osoittimilla.

Seuraavaksi piti päättää tietorakenne näille, missä dataa säilytetään. Päädyin ”unordered_map”:iin, koska se on nopeampi kuin map ja järjestystä ei tarvitse pitää yllä. Ohjelmassa on toteutettu kaksi ”unordered_map”, josta toinen sisältää pysäkin ID:n ja pysäkkien ”strucktin”. Toinen taas alueen ID:n ja alueen ”strucktin”. Nyt sain kaiken datan säilytettyä helposti ja ovat ikäänkuin puurakenteessa.

Toteutin aluksi lähes kaikki funktiot toimivaksi ja sen jälkeen aloin parantelemaan niitä. Esimerkiksi tein pari ”vektor” säiliötä, joista toinen pitää pysäkkejä koordinaattien mukaisessa järjestyksessä ja toinen pysäkkien nimien perusteella aakkosjärjestyksessä. Nämä siksi, koska oli huomattavasti nopeampaa uudelleen järjestää nämä, kuin katsoa ”unordered_map”:sta jokainen kerta. Nämä tietenkin ovat tehty niin, että poistot, lisäykset ja muutokset ovat otettu huomioon.

Olen käyttänyt myös harjoitustyössä paria eri muuta ”struct” rakennetta, joka pitää sisällään boolean operaattorin, jolla voidaan verrata kahta eri pysäkkien niiden etäisyyksien mukaan. Tämä on ikäänkuin apu järjestämis funktiolle, koska eihän c++ tiedä, miten käyttäjä haluaa järjestää ”unordered_map”:in tavarat, koska siellä on vain ID ja ”struct”.

Ohjelmaan tein myös muutamia apu funktioita:

- twoPointDistance = palauttaa kahden pisteen välisen etäisyyden.
- findRegionParents = etsii kyseisen alueen parentit rekursiivisesti.

- findRegionParentsSecond = samanlainen kuin findRegionParents, mutta pienellä twistillä.
- findRegionChilds = etsii alueen lapset rekursiivisesti.
- findCommonRegion = etsii kahden eri alueen yhteisen alueen.
- sortStopsByCoord = järjestää vektorin koordinaattien mukaan.
- sortStopsByName = järjestää vektorin aakkosjärjestykseen.