# Implementering af programmet

I dette afsnit vil programmet blive beskrevet, både en overordnet programbeskrivelse og en dybdegående forklaring af programmets funktioner. Derudover vil der blive beskrevet hvordan brugeren interagere med programmet.

# Programbeskrivelse

Programmet begynder med at indlæse alle de forudbestemte tilgængelige attraktioner fra en tekstfil. Alle disse attraktioner bliver vist på en liste for brugeren i kommandopromten med numre ud for hver attraktion. Brugeren vælger så hvilke attraktioner vedkommende har lyst til at besøge, ved at indtaste attraktionens nummer, hvor attraktionen så vil blive tilføjet til en liste. Denne liste kører gennem programmet, og den korteste rute bestemmes. Programmet tjekker derefter, for nærliggende attraktioner der kan tilføjes til ruten, på samme måde som i begyndelsen da de valgte attraktioner til deres rute. Hvis brugeren vælger ekstra attraktioner til deres rute, kører programmet igen, og den nye rute beregnes. Når ruten er beregnet, vil ruten være tilgængelig i tekstfilen ”KortesteRute.txt”, hvor brugeren kan se hvilken rækkefølge attraktionerne skal besøges i, og distancen mellem dem.

Attraktionerne til programmet bliver indlæst fra en .txt fil når prorammet køres. I programmet bliver filen indlæst i en funktion som hedder initialiserAttraktioner. Hvis filen ikke er tom vil elementerne, vha. funktionen fscanf, blive indlæst i grupper af 4, hvor de bliver indlæst som ”attraktion” hvilket er defineret som et struct. Hvis filen er tom, vil en advarsel blive vist i prompten og lukke programmet ned. Når alt information er indlæst fra filen, og den ikke længere er nødvendig, lukkes filen.

**Indsæt billede af ”initialiserAttraktioner”.**

Brugeren bliver præsenteret for en liste over alle tilgængelige attraktioner fra inputparameteren, og attraktionernes tilhørende nummer i funktionen ” valgafAttraktioner”. Brugeren bliver bedt om at indtaste attraktionernes matchende numre, hvilket vil blive tilføjet til en liste. Taster brugeren 0, bliver brugeren præsenteret for sine valg, og beregningen af den korteste rute igangsættes. Programmet returnere listen af valgte attraktioner.

**Indsæt billede af ”** **valgafAttraktioner”.**

Funktionen ”findNaboRute” benytter NNA (Nearest Neighbour Algorithm, eller Nærmeste Nabo Algoritme), til at finde den korteste rute mellem en række attraktioner for et bestemt startsted, der er opgivet i inputparametrene. Efter at have fundet den korteste rute ud fra NNA som en beskrevet i teoriafsnittet **X**, returnere den et array med ruten og distancen for denne rute.

**Indsæt billede af ”findNaboRute”.**

”findKortesteNaboRute” benytter funktionen ”findNaboRute”, til at finde ud af hvilket startsted der giver den korteste rute. Dette gøres ved at sætte en variable til en stor værdi, som rutedistancen ikke vil gå over, og opdatere den hvis ”findNaboRute” returnere en distance for de givne attraktioner med en given start attraktion, der er lavere end de forgående ruter. ”findKortesteNaboRute” returnere et array der er en sorteret liste over attraktionerne, og den samlede længde mellem disse.

**Indsæt billede af ”findKortesteNaboRute”.**

Funktionen ”beregn\_dist” er en implementering af Harasin formlen, der er beskrevet i teoriafsnittet **X**. Funktionen tager længde- og breddegrader for to attraktioner som inputparametre. Disse længde- og breddegrader bliver brugt i formlen til at udregne distancen mellem de to punkter. Funktionen returnere til sidst den beregnede distance mellem de to attraktioner.

**Indsæt billede af ”beregn\_dist”.**

Funktionen ”udregn\_kanter” bruges til at udregne distancen mellem punkterne. Beregningen af distancerne sker vha. ”beregn\_dist”, ved at sende længde- og breddegrader for attraktionerne, og opretter en kant. Dette gøres i en for løkke i en anden for løkke, hvor der for hvert punkt blive udregnet distancen til de punkter der ikke allerede er blevet oprettet en kant for i en tidligere itteration af løkkerne. Kanterne bliver tilføjet til kantarrayet, gennem den pointer der er inputparameter til funktionen, så kanterne kan blive tilgået fra resten af programmet.

**Indsæt billede af ”udregn\_kanter”.**

Brugeren af programmet bliver præsenteret for ruten gennem funktionen ”output\_liste”. Funktionen udskriver en liste med attraktionerne i den rækkefølge de skal besøges, hvis den korteste rute ønskes. Funktionen sørger også for at skrive distancen mellem attraktionerne.

**Indsæt billede af ”output\_liste”.**

Funktionen ”findDist” gennemgår alle kanter der allerede er oprettet, for at returnere distancen mellem to attraktioner, uden at skulle beregne den igen.

**Indsæt billede af ”findDist”.**

Funktionen ”attraktionErTilfoejet” bruges til at finde ud af, om en attraktioner allerede er tilføjet til ens liste over ekstra attraktioner, og returnere enten true eller false.

**Indsæt billede af ”attraktionErTilfoejet”.**

Funktionen ”prikProdukt” bruges til at finde prikproduktet mellem to vektorer, hvilket også er det funktionen returnere.

**Indsæt billede af ”prikProdukt”.**

..

**Indsæt billede af ”findEkstraAttraktionerFirkant”.**

For at foreslå ekstra attraktioner til den valgte rute, bruges funktionen ”findEkstraAttraktioner”, hvor dette vil udgøre den interessante rute. Dette gøres den ved at først at bruge funktionen ”findDist”, til at finde ikke valgte attraktioner inden for en bestemt distance af de valgte attraktioner. Er en attraktion ikke inden for denne radius, bruges funktionen ”findEkstraAttraktionerFirkant” for at finde ud af, om attraktionen ligger tæt på ruten mellem to attraktioner. De attraktioner der enten er inden for den bestemte distance af enten attraktionerne eller ruterne derimellem, tilføjes til et array der returneres fra funktionen.

**Indsæt billede af ”findEkstraAttraktioner”.**

”valgAfEkstraAttraktioner” præsenterer brugeren for listen af ekstra attraktioner, som ligger tæt på enten de tidligere valgte attraktioner eller på ruten mellem attraktionerne. Brugeren har så mulighed for at vælge mellem disse attraktioner, på samme måde som i ”valgafAttraktioner” funktionen. Attraktionerne bliver tilføjet til det oprindelige ”valgteAttraktioner” array.

**Indsæt billede af ”valgAfEkstraAttraktioner”.**

Funktionen ”andre\_startsted”, tager en rute som inputparameter, sammen med det startsted der ønskes. Funktionen laver derefter et nyt array, der har det nye startsted som første og sidste element, så ruten starter i startstedet, og vender tilbage dertil. Dette array er hvad funktionen returnere.

**Indsæt billede af ”aendre\_startsted”.**

# Opsummering

Her er en liste med de overstående funktioner, med en kort forklaring om hvad hver funktions opgave i programmet er.

* initialiserAttraktioner – Initialisere attraktioner fra en tekstfil
* valgafAttraktioner – Lader brugeren vælge attraktioner
* findNaboRute – Bestemmer den korteste rute mellem en række attraktioner, for en bestemt attraktion
* findKortesteNaboRute – Finder det optimale startsted
* beregn\_dist – Beregner distancen mellem to attraktioner
* udregn\_kanter – Bruger ”beregn\_dist” til at oprette kanter mellem attraktioner
* output\_liste – Udskriver den korteste rute til brugeren
* findDist – Returnere en allerede beregnet distance mellem to attraktioner
* attraktionErTilfoejet – Finder ud af om attraktionen allerede er tilføjet til ens liste over attraktioner
* prikProdukt – Udregner prikproduktet mellem to vektorer
* findEkstraAttraktionerFirkant – Finder ekstra attraktioner tæt på ruten mellem attraktionerne
* findEkstraAttraktioner – Forslår ekstra attraktioner til ruten
* valgAfEkstraAttraktioner – Lader brugeren vælge mellem hvilke ekstra attraktioner der ønskes besøgt
* aendre\_startsted – Ændre rutens startsted