a) Algoritmit (Algorithms)

- Sovellutusta ohjelmoitaessa on hyvä jakaa ohjelmointitehtävä pienempiin ongelmiin. Tämän jälkeen niiden ratkaiseminen onnistuu helpommin yksitellen. Tehtävät kannattaa olla selkeitä, joihin voi paneutua tehtävä tai ongelma ongelmalta ja ratkaista ne. Kun näin tekee myös sovellutuksen tekeminen tavallaan etenee palikka palikalta eteenpäin. Palikat on tietysti oltava sellaisia, että ne sopivat yhteen kun kaikki kootaan yhteen ohjelmistossa.

- Tällä tunnilla ryhdymme tarkastelemaan, miten ongelmia ratkotaan ohjelmoinnissa. Tämä on ohjelmoinnin kannalta tärkeä vaihe sillä, jollei tehtävää ole ratkaisu, ei siitä varmaan pysty kovin kummoista ohjelmaa ohjelmoimaan. Varsinkaan jos on kyseessä vaikeampi ongelma.

- Esimerkkinä voisi alkuun käyttää matematiikasta tuttua keskiarvon laskentaa. Tehtävänä olisi siis ratkaista, miten keskiarvo lasketaan, ja sen jälkeen ohjelmoida tähän tarkoitukseen ohjelma. Jos et tiedä, miten keskiarvo lasketaan, joudut ensin selvittämään, miten se lasketaan. Matematiikan kirjojen avulla selviäisi, että ensin täytyy saada jostain numerot, joista keskiarvo on tarkoitus laskea. Näistä lähteistä selviäisi myös, että keskiarvossa lasketaan numeroiden keskinäinen summa ensimmäiseksi, jonka jälkeen tarvitaan vielä tietää, kuinka monta numeroa laskettiin yhteen. Tämän jälkeen vasta pääsee jakamaan summan lukumäärällä. Tästä voisi esittää kaavankin:

$$\text{keskiarvo} = \frac{\text{numero1+numero2+…}}{\text{numeroiden\_lukumäärä}}$$

  Kun tämä on laskettu, on saatu keskiarvo selville.

  Olemme tehneet ensimmäisen algoritmimme. Matematiikasta löytää paljon vastaavanlaisia esimerkkejä, sillä sieltä algoritmi käsite on peräisin. Ratkaisumme keskiarvon lasketaan, on erittäin yksityiskohtainen. Myös työvaiheet ovat tarkkaan esitetty. Niiden järjestystä ei voi kovinkaan paljoa muuttaa. Lukumäärän ja summan voi laskea missä järjestyksessä vain, mutta muut työvaiheet on tehtävä tiukasti tietyssä järjestyksessä. Tämä kaikki tekee ratkaisustamme sellaisen, että siitä voi kutsua algortmiksi.

  Tietokone ohjelmissa haluamme kuitenkin tehdä ohjelmia myös muunlaisista tehtävistä. Niihinkin on ensin hyvä miettiä ratkaisumalli ennen kuin alkaa ohjelmoimaan. Näin olisi vaikkapa tilanne uuden navigointi ohjelmankin kohdalla. Tässä navigointi ohjelmassa tarvitaan tietään kahden paikkakunnan välinen matka. Huomaa, että koko navigaattorin sisältämiä piirteitä ei ratkaista yhdellä kertaa. Niille tehdään kullekin oma algoritminsa. Ongelmamme kahden paikkakunnan välisestä matkasta on selkeä, joten siitä varmaankin saisimme aikaan hyvän yleiskäyttöisen algoritmin keskiarvon tapaan. Tätäkin ongelmaa ryhdymme ratkomaan selvittämällä, miten tämän laskennan voisi tehdä. Ensin täytyy tietysti tietää paikkakunta, josta lähdetään. Toisena tietona tarvitaan tietää kohdepaikka. Nämä ovat tavallaan tulevan ratkaisumme lähtötietoja aivan kuten keskiarvon laskennassa numerot. Nämä lähtötiedot eivät tässä tehtävässä riitä, vaan tarvitsemme myös kartan, jolta kyseiset paikkakunnat löytyvät. Tietokoneelle kaikki tiedot pitää kuitenkin olla datana (numeroina tai tekstinä). Tämän vuoksi joudumme tekemään ”kartastamme” numeromuotoisen, jossa eri paikkakuntien väliset etäisyydet ovat tallennettuna.

Esimerkiksi:

Helsinki, Lahti, 110
Lahti, Heinola, 40
Helsinki, Turku, 175
Oulu, Kemi, 110
….
Ainakin tämän tehtävän ratkaisuun tämä tietorakenne riittäisi. Koko navigaattorin tarpeisiin päätyisimme vektorimaiseen tietorakenteeseen, mutta emmehän ole vielä edes niin pitkällä. Nyt ainakin tämän tehtävän tarpeisiin nämä tiedot riittävät. Jos haluaisimme tietää matkan Helsingistä Heinolaan, pystymme selvittämään sen haun avulla yllä olevasta tietorakenteesta. Helsingin ja Heinolan välistä matkaa ei ole suoraan, mutta Lahti on tietorakenteen perusteella näiden välillä, joten laskemalla 110 yhteen 40, saamme matkan selville. Jälleen kerran ratkaisumme ongelmaamme on hyvin yksityiskohtainen. Jälleen kerran jotkut työvaiheet on tehtävä tiukasti tietyssä järjestyksessä. Joitain toki voi tehdä vapaamminkin, muttei ihan missä järjestyksessä vaan. Olemme saaneet tehtyä matka ongelmaamme algoritmin, jonka perusteella voi ryhtyä ohjelmoimaan varsinaista ohjelmaa.

- Tämän viikon viikkotehtävissä harjoitellaan algoritmista ajattelua, eli ensin ratkaistaan ongelma kyllin yksityiskohtaisesti vaihe vaiheelta eteneväksi ratkaisuksi. Ja kun tämä on tehty, tehdään ohjelma. Tämän viikon kahdessa ensimmäisessä tehtävä sarjassa ratkaistaan Blocklies:iin tehtyjä peli ja labyrinttitehtäviä. Niissä navigaattoritehtävämme tietorakenne näkyisi grafiikkana, niin kuin näkyisi meidän lopullisessa navigaattorissamme. Näyttäisimme siellä käyttäjälle tietysti kartan emmekä suunnittelemaamme tietorakennetta. Blocklies:in Flappy Bird ja labyrintti tehtävissä ohjelma tietysti vastaavanlaisesti käyttää tehtävään sopivaa tietorakennetta.

- Tämän viikon kolmantena tehtävänä on Blocklies:issa painoindeksin laskentatehtävä. Se on vähän samankaltainen tehtävä kuin keskiarvon laskenta edellä. Ensin on selvitettävä, miten painoindeksi (bmi, body mass index) lasketaan. Tämän saa varmasti nopeasti selville Google haulla. Tämän jälkeen pääsee tehtävässä eteenpäin.

- Algoritmeja on nykyään tehty lähes tulkoon minkä tahansa tehtävän ratkaisemiseen. On osoittautunut, että melkein kaiken voi esittää data muodossa (siis merkkinä ja numeroina vähän samaan tapaan kuin teimme edellä matkan laskentatehtävässä). Käyttäjälle tietysti useimmiten näytämme jotain sopivaa grafiikkaa, mutta itse tietokone ohjelma tekee kaiken dataan perustuen.

b) Algoritmien kuvaustavat

- Edellä esitimme algoritmimme aina tekstimuodossa ilman sen kummempaa kuvaustapaa. Olisimme voineet käyttää vaikkapa vuokaaviota (Flowchart) tai pseudokoodia (Pseudocode). Ne ovat tavallisimmat vaihtoehdot, joita näkee käytettävän. Vuokaavio on ehkä havainnollisempi, mutta moni mutkaisten algoritmien esittämisessä loppuu helposti tila. Pseudokoodi puolestaan sopii myös moni mutkaisten ja pitkien algoritmien esittämiseen. Tällä opintojaksolla emme perehdy näihin tarkemmin, mutta niistä on hyvä tietää jo tässä vaiheessa, että erilaisia algortimien kuvaustapoja on olemassa.

- Vuokaavio (Flowchart, ks. https://en.wikipedia.org/wiki/Flowchart)
  Yllä esitetyssä Wikipedian linkissä on esitetty vuokaavio melko kattavasti. Koska kyseessä on kaavio tai piirustus, on se usein havainnollinen. Huonona puolena on se, että moni mutkaisen algoritmien kuvaamisessa voi tulla puute tilasta.

- Pseudokoodi (Pseudocode, ks. https://en.wikipedia.org/wiki/Pseudocode)
  Pseudokoodia näkee usein käytettävän moni mutkaisten algoritmien kuvaamisessa. Pseudokoodi usein on puoleksi englantia ja puoleksi ohjelmointikieltä sisältävä esitystapa. Sen vuoksi nimikin on pseudokoodi. Wikipediassa on kattava esitys pseudokoodista yllä olevassa linkissä.