Geometria ja ohjelmointi

Joustavaan matematiikkaan -täydennyskoulutushanke

Materiaalin kuvaus: Tässä Geometria ja ohjelmointi -itseopiskelumateriaalissa paneudutaan matematiikan ja ohjelmoinnin välisiin yhteyksiin syvemmällä tasolla koulun geometrian yhteydessä. Materiaali soveltuu opettajille, jotka jo hallitsevat ohjelmoinnin; kurssilla käytetään Scratch-kieltä, mutta sitä ei välttämättä tarvitse osata ennestään. Teknisen puolen sijaan kurssilla keskitytään matemaattisen ongelmaratkaisuprosessin ja uuden ohjelman luomisen analogiaan prosesseina sekä matemaattisen perustelun kehittämiseen ohjelmoinnissa opittujen seikkojen perustuen. Materiaali sopii parhaiten peruskoulun 3. luokasta lukioon.

Materiaalia on käytetty ja kehitetty aiemmin Opetushallituksen rahoittamissa täydennyskoulutuksissa. Materiaalin vertailutehtävät on tuotettu osana Joustavaan matematiikkaan -hanketta (JoMa). JoMa on vuosina 2018–2023 toiminut valtakunnallinen matematiikan opetuksen täydennyskoulutushanke varhaiskasvatukseen, esiopetukseen, alakouluun, yläkouluun ja lukioon. Hankkeessa tuotettiin 19 verkkokurssia. Kursseilla kehitettyä materiaalia löytyy täältä Avointen oppimateriaalien kirjastosta. Opetushallituksen rahoittaman hankkeen toteuttamiseen osallistuivat Turun yliopisto, Åbo Akademi, Jyväskylän yliopisto ja Oulun yliopisto.

Materiaalin tekijät: Peter Hästö & Riku Klén sekä Vuokko Kangas, Päivi Portaankorva-Koivisto & Johanna Saari

Lisenssi: Tämä teos on lisensoitu Creative Commons Nimeä 4.0 Kansainvälinen -käyttöluvalla. Tarkastele käyttölupaa osoitteessa https://creativecommons.org/licenses/by/4.0/deed.fi.

Geometria ja ohjelmointi

Peter Hästö & Riku Klén

sekä

Vuokko Kangas, Päivi Portaankorva-Koivisto & Johanna Saari

20. syyskuuta 2019

Sisältö

Luku 1. Tausta-ajatuksia 3
1. Matematiikka ja ohjelmointi 3
2. Ongelmanratkaisu 5
3. Heuristiikkoja matematiikkaan ja ohjelmointiin 7
4. Mitä oppimateriaali sisältää? 9

Luku 2. Opetusmateriaali 12
1. Kulmat 12
2. Säännöllinen monikulmio 15
3. Mystinen luku π 20

Luku 3. Vertailutehtävät 24
1. Kulmat 25
2. Säänöllinen monikulmio 28
3. Mystinen luku π 35

Liite A. Korkeampi ulottuvuus ja fraktaalit 39
4. Pinta-ala, eli 2-alotteinen tilavuus 39
5. π:n paluu 41
6. Rekursio 43
7. Fraktaali 45
8. Dimensio 47

LUKU 1

Tausta-ajatuksia

1. Matematiikka ja ohjelmointi

Ohjelmoinnin oppimisen haasteita ovat Bentonin, Hoylesin, Kalasin ja Nossin[1] mukaan olleet kielen syntaksi ja siitä johtuvat virheilmoitukset, niiden ymmärtäminen ja korjaaminen, sekä luokan rajoittunut teknologia. Uudet ohjelmointiympäristöt ja -kielet ovat huomattavasti korjanneet tilannetta varsinkin, kun oppilaiden kanssa on voitu käyttää graafista ohjelmointia. Teknologiarajoitteetkin ovat vähentyneet, kun useita ohjelmointiympäristöjä voidaan käyttää verkkopohjaisina. Ohjelmointiosaamisen voi jakaa esimerkiksi seuraaviin osa-alueisiin:

(1) Ohjelmallinen (algoritminen) ajattelu.
(2) Peruskäskyt ja -elementit: muuttujat, funktiot, `for/while` ja `if` -käskyt, aritmeettiset ja loogiset operaattorit.
(3) Tietorakenteet.
(4) Ohjelmointikielen syntaksi.
(5) Erikoiskäskyt (usein erillisenä pakettina toteutetut).
(6) Ohjelmointiympäristön toiminta.

Lye ja Koh[2] jakavat ohjelmallisen ajattelun kolmeen dimensioon: ohjelmalliset käsitteet, ohjelmalliset käytänteet ja ohjelmalliset näkökulmat. *Ohjelmallisia käsitteitä* ovat ohjelmoijan käyttämät käsitteet kuten muuttujat ja silmukat. *Ohjelmallisia käytänteitä* ovat ongelmanratkaisuun liittyvät menetelmät, joita tarvitaan ohjelmointiprosessin aikana kuten iterointi, testaus, uudelleenkäyttö ja yhdistely, abstrahointi ja modularisointi. *Ohjelmallisella näkökulmalla* Lye ja Koh tarkoittavat oppilaiden käsityksiä itsestään osana teknologisoituvaa yhteiskuntaa ja mahdollisuuksia tarkastella sitä kriittisesti. Lyen ja Kohin mielestä visuaaliset ohjelmointikielet helpottavat näiden kolmen dimension omaksumista, sillä ne eivät sisällä tarpeettomia syntakseja, kuten puolipisteitä ja sulkeita, vaan vähentävät oppilaiden kognitiivista kuormaa: näin oppilaiden on helpompi omaksua ohjelmallisia käsitteitä. Oppilaat myös oppivat ohjelmallisia käytänteitä, koska visuaalisessa ohjelmointiympäristössä ohjelman toiminta näkyy heti animoituina objekteina ja testaus ei edellytä niin paljon kognitiivista työskentelyä. Silloin opitaan helpommin myös ongelmanratkaisua. Kun oppilailla on mahdollisuus rakentaa multi-mediaa sisältäviä digitaalisia tuotteita, he voivat myös ilmaista luovia ajatuksiaan, kehittää digitaalista lukutaitoaan ja toimia aktiivisina teknologian käyttäjinä.

Kun tarkastellaan ohjelmoinnin historiaa, havaitaan, että ohjelmallinen ajattelu ja ohjelmoinnin peruskäskyt ja -elementit ovat pääosin säilyneet ennallaan sitten 1970-luvun. Jos otetaan huomioon, että tämän päivän koululaisten pitää pärjätä työelämässä 2060 tai 2070-luvulle asti, on mielestämme perusteltua ohjelmoinnin opetuksessa panostaa nimenomaan näihin kestäviin perustuksiin. Voi sanoa, että kohdat (1) ja (2) muodostavat perustan, josta jokainen innovatiivinen ohjelma muodostuu. Jos haluaa kulkea polkua, joka on jo valmiiksi tallattu, niin näyttävämpiä tuloksia saa erikoiskäskyillä. Oppilaiden motivointi vaatii koulussa ehkä näiden kahden elementin yhdistämistä. Näistä syistä emme pidä erityisen tärkeänä myöskään kysymystä siitä, mitä ohjelmointikieltä opetuksessa käytetään.

[1] Bridging primary programming and mathematics: Some findings of design research in England, 2017
[2] Review on teaching and learning of computational thinking through programming: What is next for K-12? 2014

Kuva 1. Tietokoneita 1970 ja 2010-luvuilla.

Tässä monisteessa olemme käyttäneet graafista Scratch -ohjelmointiympäristöä. Mielestämme se sopii hyvin myös yläkouluun, varsinkin oppilaille, joilla ei ole paljon ohjelmointikokemusta. Graafinen ohjelmointiympäristö mahdollistaa tutkivamman lähestymistavan, eikä aikaa tuhlaannu syntaksivirheiden tekemiseen ja etsimiseen. Scratchin etu on myös, että kaikki käskyt ovat valmiina saatavilla valikosta, joten erillisiä oppaita ei tarvitse opiskella. Tästä huolimatta Scratch on yllättävän monipuolinen, ja siinä pystyy ratkomaan monenlaisia ongelmia. On toki sanottava, että ongelmien monimutkaisutessa rupeaa Scratchista jossain vaiheessa ollemaan enemmän haittaa kuin hyötyä; tämä on se vaihe, jossa on luonnollista siirtyyä tekstipohjaiseen kieleen. Erityisesti lausekkeisen tekeminen ja muuttujien sekä tietorakenteiden hallinnointi on Scratchissa hankalampaa. Valitettavaa on myös debuggerin puuttuminen Scratch -ympäristöstä; luvussa 4 on yksi esimerkki, miten tämän puutteen voi kiertää.

Misfeldt ja Ejsing-Duun[3] ovat havainneet, että matematiikan ja ohjelmoinnin yhdistämisellä voidaan saavuttaa kolmenlaisia oppimistuloksia:

(1) oppilas asemoituu uudella tavalla tiedon tuottajaksi tai tuotteen laatijaksi,
(2) yhdistäminen tukee abstraktia ajattelua ja
(3) yhdistäminen kehittää algoritmista ajattelua.

Schanzer, Fisler, Krishnamurthi ja Felleisen[4] toteavat kuitenkin, että tietotekniset lähestymistavat (ml. ohjelmointi) eivät ole juuri tuottaneet haluttua matemaattista osaamista. Ongelmana on puutteellinen siirtovaikutus yhdestä kontekstista toiseen. Tutkimusten mukaan siirtovaikutus edellyttää syvällisiä rakenteellisia yhteyksiä eri kontekstien välillä sekä nimenomaista opetusta siitä, miten käsitteet toisesta kontekstista soveltuvat toiseen kontekstiin. Opettajan rooli on siis keskeinen.

A. Stylianides[5] on esittänyt, miten päättely voi olla osa matematiikan opetusta alusta alkaen, kunhan määritelmät on sovitettu kohdejoukon tasolle. Tämän monisteen tehtävissä lähdetään liikkeelle koulusta tutun käsitteen tarkemmalla määritelmällä. Määritelmien tarkoituksena on mahdollistaa tutun kaavan perusteleminen. Perusteleminen on tarkoitettu haastavaksi tehtäväksi. Tässä kohtaan mukaan astuu ohjelmointi. Perustelussa vaadittavia elementtejä tuodaan esiin ohjelmointitehtävissä. On tärkeää huomata, että ohjelman laskemat esimerkit, kuten muutkaan empiiriset havainnot, **eivät muodosta matemaattisesti pätevää perustelua**. Sen sijaan ohjelmoinnin avulla oppilaat hahmottavat (opettajan avustuksella), niitä määritelmien ominaisuuksia, joita perusteluun voidaan käyttää. On opettajan vastuulla varmistaa, että kaikki oppilaat ymmärtävät, että matemaattisesti pätevä perustelu

[3] Learning mathematics through programming: An instrumental approach to potentials and pitfalls, 2015
[4] Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap, 2015
[5] The notion of proof in the context of elementary school mathematics, 2007

etenee määritelmistä deduktiivisin askelin haluttuun lopputulokseen. Kun formalismin taso on sopiva, on tutkimuksissa osoitettu, että jo pienetkin lapset kykenevät tällaiseen deduktiiviseen päättelyyn[5]. Tässä monisteessa sekä matemaattista että ohjelmallista tehtävää on lähestytty ongelmanratkaisun keinoin. Vaikka monisteessa on vain muutama tehtävä, kulunee niiden tekemiseen useampi oppitunti. Eriyttäminen perustuu Swanin[6] ”Differentiating by level of support” -lähestymiseen (s. 47): eri oppilasryhmille annetaan eri tasoista tukea vihjeiden, ohjeistuksen ja vertailutehtävien muodossa. Vihje-heuristiikat on tärkeää antaa sellaisessa yleisessä muodossa, että oppilaat saavat itse harjoitella strategista suunnittelua (eli heuristiikan soveltamista yksittäiseen tilanteeseen).

2. Ongelmanratkaisu

Leppäahon[7] määrittelyn mukaan ongelmatehtävä on sellainen tehtävä jota oppilas ei kykene välittömästi ratkaisemaan, mutta hänellä on kuitenkin valmiudet ratkaisun saavuttamiseen ajattelun ja opiskelun avulla. Ongelmatehtävä eroaa tavallisesta suljetusta tehtävästä monin tavoin. Yeo[8] esittää, että suljetussa tehtävässä on yksiselitteinen ratkaisu, jonka saamiseen on vain yksi mahdollinen menetelmä, se on riittävän helppo kaikille oppilaille ja sitä ei voida laajentaa. Tehtävän avoimuus sen sijaan voi näyttäytyä monin eri tavoin. Avoimen ongelmatehtävän tavoite voi olla hyvin määritelty ”etsi niin monta kuin pystyt” tai epätarkasti määritelty ”tutki”. Avoin ongelmatehtävä voi myös olla tutkiva tehtävä, jossa lähtökohta on määritelty, mutta tuloksena voi olla erilaisia ratkaisuja. Nämä ratkaisut voivat olla objektiivisia ja odotettavia (esimerkiksi piin määrittäminen erilaisia ympyrälieriöitä tarkastelemalla) tai subjektiivisia kuten useat todellisen elämän ongelmatehtävät. Edellisten lisäksi avoimet ongelmatehtävät voivat olla menetelmällisesti avoimia tai niitä voidaan syventää tai laajentaa ja näin lisätä niiden kompleksisuutta tai mahdollisuutta etsiä yleistä ratkaisua.

Tiivistetysti ongelmanratkaisussa on kyse siitä, miten toimit, kun kohtaat ennalta tuntemattoman tilanteen, johon aiemmin harjoittelemasi menetelmät eivät suoraan toimi. Ongelmanratkaisutilanteesta selviäminen vaatii perustaitojen lisäksi sinnikkyyttä, strategista toiminnan suunnittelua, sekä metakognitiota. Nämä ovat taitoja, jotka ovat tulevaisuuden työtehtävissä ehdottoman tarpeellisia. Nämä ovat taitoja, joita on haasteellista omaksua, sillä niiden kehittyminen vaatii pitkäaikaista työskentelyä ja ponnistelua sekä oppilaalta että opettajalta. Nämä ovat taitoja, joita voi oppia ohjelmoinnin ja matematiikan avulla.

Ongelmanratkaisun ensimmäinen vaihe on valmisteluvaihe. Tätä vaihetta oppilas ei yleensä näe, vaan tässä vaiheessa opettaja valitsee ongelmatehtävän. Swan[6] ehdottaa, että valitessaan ongelmatehtävää opettajan tulisi käyttää rikastuttavia tehtäviä. Näitä kuvastaa se, että niissä pääsee heti alkuun, mutta ne ovat laajennettavissa. Ne ovat hauskoja ja yllätyksellisiä ja oppilaat saavat niitä tehdessään itse tehdä päätöksiä. Rikastuttavat tehtävät haastavat oppilaat testaamaan, todistamaan, selittämään, pohtimaan ja tulkitsemaan. Ne aktivoivat keskustelua ja rohkaisevat yksilöllisiin ratkaisuihin ja keksintöihin. Rikastuttavissa tehtävissä yhdistellään eri matematiikan osa-alueita toisiinsa ja usein niissä käytetään myös sopivasti teknologiaa. Valmisteluvaiheessa opettaja tutustuu valitsemansa ongelmatehtävän taustalla olevaan matemaattiseen sisältöön, käsitteisiin ja sääntöihin, ja miettii, miten esittäisi ongelmatehtävän oppilailleen.

Ongelmanratkaisun toinen vaihe on ongelman esittäminen. Tässä vaiheessa opettajan on tärkeintä varmistaa, että oppilaat ymmärtävät, mitä tehtävässä on tarkoitus tuottaa. Usein tehtävänanto sisältää myös motivoivia elementtejä. Nämä ovat tärkeitä, sillä ongelmanratkaisuun liittyy paljon tunnetiloja. Työskentely vaatii pitkäjänteisyyttä ja saattaa tuntua joskus

[6]Collaborative Learning in Mathematics, A Challenge to our Beliefs, 2006

[7]Matemaattisen ongelmanratkaisutaidon opettaminen peruskoulussa: ongelmanratkaisukurssin kehittäminen ja arviointi, 2007

[8]Development of a framework to characterise the openness of mathematical tasks, 2017

jopa turhauttavalta. Ongelman esittämisvaiheessa opettaja yleensä myös organisoi työskentelyn, ajoittaa sen ja jakaa tarvittavat materiaalit. Swan[6] ehdottaa, että opettaja käyttäisi ongelmanratkaisutehtävissä yhteisöllistä pienryhmätyöskentelyä.

Kolmas ongelmanratkaisun vaiheista on ongelman parissa työskentely. Kun oppilaat työskentelevät ongelmatehtävän kimpussa joko pareina tai pienryhminä, opettajan rooli muuttuu ohjaajan rooliksi. Hän tukee työskentelyä sekä emotionaalisesti että tiedollisesti. Emotionaalisen tuen tarkoituksena on kannustaa, rohkaista, kohottaa itseluottamusta ja lisätä tehtävään sitoutumista. Opettajan tulee havainnoida työskentelyä niin, että tunnistaa erilaiset tuen tarpeet. Pehkonen ja Ahtee[9] kutsuvat tätä empaattiseksi kuuntelemiseksi ja korostavat, että tärkeämpää kuin antaa ohjeita tai jakaa ideoita, on saada oppilaat kertomaan omista ideoistaan ja jäsentää niitä. Swan[6] suosittelee, että opettaja rakentaa työskentelyä ja ohjaamista oppilaiden tietojen ja taitojen varaan. Opettajan on hyvä nostaa esiin ja keskustella oppilaiden uskomuksista. Onko matemaattinen osaaminen sitä, että ratkaisee tuossa tuokiossa jonkin tehtävän vai sitä, että jaksaa työskennellä ongelman kimpussa luovuttamatta. Tässä vaiheessa opettajan kannattaa panostaa kysymysten laatuun ja antaa oppilaille riittävästi aikaa vastata. Hyviä ohjaavia keskustelunavauksia ovat: Kerro esimerkki. Mikä virhe tässä on, miten korjaat sen? Mitä eroa ja mitä yhtäläisyyttä ratkaisuilla on? Tämä on erikoistapaus, mutta mistä? Pitääkö tämä aina paikkansa? Perustele. Voitko olla varma, että ...? Voitko vakuuttaa minut? Työskentelyvaiheessa opettajan kannattaa arvostaa pikemminkin menetelmiä kuin vastauksia.

Lye ja Koh[2] esittävät opettajan ohjaukselle kuusi roolia:

(1) Rekrytointi – opettaja ylläpitää oppilaiden kiinnostusta tehtävään
(2) Vapausasteiden rajaaminen – opettaja muokkaa tehtävän hallittavaksi
(3) Suora tuki – opettaja pitää oppilaat kiinni työssä
(4) Kriittisten ominaisuuksien merkitseminen – opettaja korostaa ominaisuutta, jolla on merkitystä tehtävän ratkaisemisessa
(5) Turhautumisen kontrolli – opettaja antaa oikea-aikaista ohjausta, jotta oppilaat eivät turhaudu ja luovuta
(6) Havainnollistaminen – opettaja toimii roolimallina ongelmanratkaisussa

Viimeinen ongelmanratkaisun vaiheista on tärkein, mutta useimmin kiireen vuoksi sivuutettu. Tässä vaiheessa koko luokka osallistuu ongelman purkamiseen. On hyvä, jos opettaja on työskentelyn aikana tehnyt havaintoja siitä, miten kukin ryhmä on edennyt ja osaa poimia erilaisia oppilaitten ratkaisuja esille. Yleensä kannattaa aloittaa esittelemällä niiden ryhmien ratkaisuja, jotka ovat edenneet tehtävässä vähiten ja lisätä niihin uusia oivalluksia pidemmälle edenneiltä ryhmiltä. Yhteenvedon tarkoituksena ei ole kuitenkaan vain esitellä eri ratkaisuja, vaan myös punnita niiden matemaattista laadukkuutta, kuulla oppilaiden selityksiä ja perusteluja. Opettajan tärkein tehtävä on luoda myönteinen ilmapiiri, rakentaa kiinnostava keskustelu sekä huolehtia kaikkien osallistumisesta keskusteluun.

Miten opettaja tukee oppilaan ongelmanratkaisua sulkematta tehtävää? Jos tehtävä on avoin ongelma eli siihen ei välittömästi ole olemassa yhtä oikeaa ratkaisumallia ja ratkaisua, opettajan on varottava, ettei vahingossa esitä ongelmaa liian suljetussa muodossa tai sulje tehtävää työskentelyn aikana.

Ongelmatehtävää valitessa näin voi käydä, jos opettaja kokee, että tehtävä ei vastaa oppitunnin tavoitteita tai hän ajattelee, että oppilaat tarvitsevat joitain lisätietoja voidakseen ratkaista tehtävän. Saattaa myös olla, että opettajat lykkäävät ongelmatehtävää siihen vaiheeseen, jolloin se jo lakkaa olemasta ongelmatehtävä, vaan muuttuu pikemminkin rutiinitehtäväksi. Yleensä hyvät ongelmatehtävät ovat sellaisia, joihin oppilailla on valmiuksia, mutta ei vielä opittuja menetelmiä. Opettaja voi tahtomattaan sulkea avoimen tehtävän myös esittäessään

[9] Levels of teachers' listening in working with open problem, 2006

ongelmatehtävää luokalle. Hän on saattanut opettaa tehtävän edellyttämiä tekniikoita juuri edellisellä oppitunnilla ja yrittää nyt aktiivisesti palauttaa niitä oppilaiden mieleen juuri ennen tehtävänantoa. Yleensä avoin ongelmatehtävä kannattaa antaa oppilaille valmiiksi muotoiltuna, ilman erityistä pohjustusta tai selittämistä ja huolehtia työskentelyn alkaessa, että kaikki ymmärtävät tehtävänannon ja pääsevät alkuun.

Avoimet ongelmatehtävät muuttuvat joskus suljetuiksi myös kesken työskentelyn. Näin voi tapahtua, kun opettaja huomaa, ettei tehtävä edisty hänen toivomaansa tahtia, luokassa ilmenee levottomuutta tai oppilaiden motivaatio laskee. Silloin opettaja saattaa lopettaa työskentelyn ja antaa lisäohjeita tai vinkkejä ohjatakseen oppilaiden ajattelua, tai jopa opettaa jonkin asian yhteisesti koko luokalle helpottaakseen työskentelyä. Hän voi myös päättää lyhentää työskentelyaikaa palauttaakseen työrauhan. On hyvä, jos opettaja tässä vaiheessa pitää mielessään ongelmanratkaisun tunteisiin liittyvät näkökohdat ja tukee oppilaita emotionaalisesti kannustaen ja rohkaisten, mieluummin kuin kognitiivisesti ohjaten.

Ongelmatilanteen luominen on haaste, josta opettaja on pitkälti vastuussa. Tilanne, joka toiselle oppilaalle on juuri sopivan haasteellinen, onkin toiselle oppilaalle niin helppo, ettei se edistä sinnikkyyttä mitenkään. Malcolm Swan on kirjassaan *Improving learning in mathematics: challenges and strategies* listannut useita strategioita, joita opettajat voivat käyttää haastavien tilanteiden luomiseen.

3. Heuristiikkoja matematiikkaan ja ohjelmointiin

Ongelmanratkaisuun kuuluu myös yleisesti käytettyjä strategioita eli heuristiikkoja. Heuristiikka on nyrkkisääntö, joka on osoittautunut olevan useassa ongelmatilanteessa hyödyllinen. Useissa tutkimuksissa on havaittu, ettei niiden suora opettaminen tuo toivottuja etuja, vaan tehokkaampi tapa on saattaa oppilaita tietoisiksi käyttämistään heuristiikoista ja valikoida ongelmatehtäviä niin, että eri tekniikoille tulee käyttöä. Schoenfeldin[10] mukaan tieto heuristiikasta ei auta ongelmanratkaisussa, vaan heuristiikan käyttö on taito, jota pitää harjoitella. Luokkahuone- tai ryhmäkeskustelussa opettaja voi nostaa esille yksittäisen ratkaisun taustalla olevan heuristiikan, jolloin oppilaat näkevät, miten sitä on sovellettu yksittäistapauksessa.

Käytetyin ongelmanratkaisustrategia on ”arvaa ja tarkista”. Se ei edellytä mitään valmistelua. Tavoitteena on vain tehdä valistunut arvaus ja tarkistaa, pitikö se paikkansa. Hyödylliseksi tämä heuristiikka muuttuu vasta, kun saatua tietoa käytetään tehtäessä uutta, parempaa arvausta. Toinen suosittu menetelmä on asettua tehtävän hahmon rooliin tai tilanteeseen ja esittää tai kokeilla, mitä tehtävässä tapahtuu.

Ongelmatehtävän jäsentelyyn liittyviä strategioita ovat ”piirrä kuva tai kaavio” ja ”laadi järjestetty lista tai taulukko”. Nämä auttavat hahmottamaan tilannetta ja käsittelemään sitä. Samalla huomataan, mitä tietoa puuttuu ja mitä vielä tarvitaan. Lisäksi esimerkiksi listasta havaitsee helpommin yhtenevyyksiä ja säännönmukaisuuksia. Tähän liittyy myös ”mitä tietoja tarvitaan?” -heuristiikka: mitä tietoja tallennetaan muuttujiksi (valitaan vapaaksi muuttujaksi) ja mitä tietoja johdetaan näistä erilaisilla kaavoilla (sidottuja muuttujia). Esimerkiksi suorakaiteesta voi tallentaa alavasen kulma sekä leveys ja korkeus, tai vaihtoehtoisesti alavasen sekä yläoikea kulma. Toisessa tilanteessa toinen voi olla hyödyllistempi ja toisessa toinen.

Jos ongelmatehtävässä on sarja vaiheita tai laskutoimituksia voidaan heuristiikaksi valita lopusta alkuun tai alusta loppuun työskentely. Tällainen peruuttamistekniikka sopii esimerkiksi ratkaistaessa tuttua Hanoin torni -ongelmaa.

[10] Mathematical Problem Solving, 1985

Joskus ongelmatehtävään pääsee käsiksi, kun sen muotoilee omin sanoin. Se saattaa auttaa ymmärtämään ongelman paremmin ja tunnistamaan oleelliset tekijät. Samantapainen heuristiikka on yksinkertaistaa ongelmaa esimerkiksi muuttamalla käytetyt luvut pienemmiksi tai helpommiksi käsitellä tai vähentämällä tehtävän asioiden lukumäärää. Yleensä yksinkertaisemman tehtävän ratkaiseminen helpottaa myös monimutkaisemman tehtävän ratkaisua. Lisäksi voi pilkkoa monimutkaisen ongelman pienemmiksi osaongelmiksi ja ratkaista ensin niistä yksinkertaisin.

Ohjelmoinnissa ja ongelman ratkaisussa voi usein hyödyntää 'hajoita ja hallitse' -strategiaa, jossa ongelma (tai ohjelma) jaetaan pienempiin osaongelmiin (tai osaohjelmiin) ja lopuksi osat yhdistetään yhdeksi kokonaisuudeksi. Ennen osiin jakamista ongelma tai ohjelmointitehtävä tulee analysoida ja samalla hahmottaa pienemmät osat. Kuten ongelman ratkaisussa myös ohjelmoinnissa on useita eri ratkaisuvaihtoehtoja.

Tarkastellaan esimerkkinä seuraavaa ongelmaa: arpalipukkeessa on kokonaisluku väliltä 1, . . . , 200. Järjestä 70 satunnaista arpalipuketta suuruusjärjestykseen.

> **Tapa 1.** Etsi suurin arpalippu ja siirrä se toiseen pinoon. Jäljelle jää 69 arpalipuketta. Etsi jäljellä olevista suurin ja aseta se toisen pinon päälle. Jatka näin, kunnes jäljellä on enää yksi arpalipuke, joka on pienin. Siirrä se toisen pinon päälle ja lipukkeet ovat suuruusjärjestyksessä.
> **Tapa 2.** Siirrä ensimmäinen arpalipuke toiseen pinoon. Ota seuraava arpalipuke ja laita se toisen toiseen pinoon suuruusjärjestyksen mukaan oikeaan kohtaan. Jatka näin, kunnes kaikki arpalipukkeet ovat toisessa pinossa.

Tavassa 1 tarvitaan osaohjelma, joka osaa etsiä arpalipukkeista suurimman. Tavassa 2 tarvitaan osaohjelma, joka osaa sijoittaa arpalipukkeen suuruusjärjestyksen mukaan oikeaan kohtaan.

Viimeisenä heuristiikkana tässä esitetään "vertaa ja aseta vastakkain". Tässä heuristiikassa ongelmatehtävän rinnalle otetaan jokin toinen vastaava jo tuttu tehtävä ja tutkitaan ongelmaa siihen verraten. Kun oppilaat ovat ratkaisseet useita ongelmatehtäviä, heille karttuu varastoa, johon verrata uutta ongelmaa ja katsoa, ratkeaisiko se samoja menetelmiä käyttäen.

Ainakin seuraavat heuristiikat voivat tulla käyttöön tämän monisteen tehtäviä tehdessä.

(1) Piirrä kuva/aja ohjelma
(2) Mitä tietoja tarvitaan?
(3) Ratkaise ensin osaongelma
(4) Ratkaise ensin helpompi ongelma
(5) Ratkaise ensin se ongelma jonka osaat

Tehtävien yhteydessä on tarkempia ohjeita siitä, miten mikäkin heuristiikka voisi soveltua kyseiseen tehtävään.

Ongelmanratkaisussa ehkä keskeisin neuvo löytyy kuitenkin Linnunradan käsikirja liftareille takakannelta: Don't Panic.

Tarkastellaan seuraavaksi erästä yksinkertaista ja erästä monimutkaista esimerkkiä. Aloitetaan yksinkertaisesta.

Ohjelmointitehtävä 3.1. Kirjoita ohjelma, joka laskee niiden välillä 1 ja n olevien kokonaislukujen summan, jotka eivät ole jaollisia kolmella.

Tarkastellaan miten Ohjelman 3.1 ratkaisua voisi lähestyä

(1) **Piirrä kuva/aja ohjelma**: luonnollinen tapa on kirjata ylös summia esim. arvoilla $n = 1$, $n = 2$, $n = 3$, $n = 4$ ja $n = 5$. Näitä tuloksia voi myös hyödyntää ohjelman oikeellisuuden tarkistamisessa.
(2) **Mitä tietoja tarvitaan?** Tarvitaan tieto, onko luvut $1, 2, \ldots, n$ jaollisia kolmella. Ratkaisutavasta riippuen voidaan tarvita myös summakaava luvuille $1, 2, \ldots, n$.
(3) **Ratkaise ensin osaongelma**: osaongelma voi olla esim. testaaminen onko luku jaollinen kolmella.
(4) **Ratkaise ensin helpompi ongelma**: voidaan esim. laskea kolmella jallisten summa $3 + 6 + \cdots + 3\lfloor \frac{n}{3} \rfloor$.
(5) **Ratkaise ensin se ongelma jonka osaat**: voidaan esim. laskea summa $1 + 2 + \cdots + n$.

Siirrytään lopuksi monimutkaisempaan ohjelmointiongelmaan. Ongelma on otettu kansainvälisestä ohjelmoinnin joukkuekilpailusta (International Collegiate Programming Contest, IPCP). Tarkoitus on näyttää, että kun etsitään huippuohjelmoijia, niin ratkaisevaa on nimenomaan matemaattistyyppinen päättely ja sen konkretisointi sopivin algoritmein ja tietorakenteineen, ei niinkään eri paketteihin liittyvien yksityiskohtien tietämys.

Ohjelmointitehtävä 3.2. Tasapainoinen ruokavalio (ICPC 2016, Problem A)

Pekka ostaa joka päivä yhden karkin ja syö sen. Kaupassa on m erilaista karkkia, jotka on numeroitu $1, \ldots, m$. Pekka tietää, että tasapainoinen ruokavalio on tärkeä ja noudattaa sitä karkkeja ostaessaan. Jokaiselle karkille i hän on asettanut tavoiteluvun, joka on murtoluku f_i ($0 < f_i < 1$). Hän haluaa, että karkkia i on syöty noin f_i-osa kaikista hänen syömistään karkeista. Hänen syömänsä karkit ovat *tasapainossa*, jos jokaiselle i

$$nf_i - 1 < s_i < nf_i + 1,$$

missä s_i kertoo kuinka monta i-karkkia Pekka on syönyt ja $n = \sum_{i=1}^{m} s_i$ on syötyjen karkkien kokonaismäärä.

Pekka on ostanut ja syönyt karkkeja jo jonkin aikaa ja koko ajan hänen ruokavalio on ollut tasapainossa. Kuinka kauan hän voi vielä syödä karkkeja horjuttamatta tasapainoa? Jos tavoiteluvut f_i ja tähän asti syötyjen karkkien määrät s_i on annettu, niin määritä kuinka monta karkkia Pekka voi vielä ostaa ja syödä niin että tasapaino säilyy.

Ohjelman syöte koostuu: luvuista m ja k ($1 \leq m, k \leq 10^5$), luvuista $a_1, \ldots, a_m$, missä $f_i = \frac{a_i}{\sum_{j=1}^{m} a_j}$, ja kokonaisluvuista $b_1, \ldots, b_k$, missä $b_i \in \{1, \ldots, m\}$ on i:ntenä päivänä syöty karkki. Syötteen mukaiset arvot täyttävät karkkien tasapainoehdon.

Ohjelman tuloste on kuinka monta karkkia Pekka voi enintään syödä rikkomatta tasapainoa. Jos Pekka voi syödä karkkeja loputtomasti, niin tuloste on "loputtomasti".

Esimerkkisyöte 1:
6 5
2 1 6 3 5 3
1 2 5 3 5
Ohjelman tuloste: 1

Esimerkkisyöte 2:
6 4
2 1 6 3 5 3
1 2 5 3
Ohjelman tuloste: loputtomasti

4. Mitä oppimateriaali sisältää?

Matemaattiselta näkökulmasta tässä monisteessa kerrotaan tarina dimensosta eli ulottuvuudesta. Tavoitteena on esittää matemaattisesti johdonmukainen ja oppilaiden käsitettävissä oleva, mutta haastava kokonaisuus. Perusosan (Luku 2) tarkoitus on syventää oppilaiden ymmärrystä geometrisista suureista pituus ja pinta-ala. Laajennusosa (Luku A), joka voisi toimia lukion paikallisena syventävänä kurssina, näytettään, miten tämä johtaa fraktaaleihin, joissa dimensio ei ole kokonaisluku.

Karkeasti ottaen dimensio d tarkoittaa, että kappallen K mitta on verrannollinen sen läpimitaan potenssiin d, $|K| \approx (\operatorname{diam} K)^d$. Ensimmäinen askel on oppia ymmärtämään verrannollisuutta kun $d \neq 1$. De Bock, Verschaffel ja Janssens[11] ovat usean tutkimuksen sarjassaan osoittaneet, että oppilaat (ja myös aikuiset) on erittäin huonoja käyttämään geometrisissa ongelmissa muuta kuin suoraa verrannollisuutta. Em. tutkimuksessa he esittivät oppilaille mm. seuraavat kysymykset (vapaasti suomennettuna):

- Tarvitset noin 6 tuntia purjehtiaksesi pyöreän saaren, jonka halkaisija on 70 km, ympäri. Kuinka monta tuntia tarvitset purjehtiaksesi pyöreän saaren, jonka halkaisija on 140 km, ympäri?
- Tarvitset noin 400 grammaa siemeniä pyöreää kukkaistutusta varten, jonka halkaisija on 10 metriä. Kuinka monta grammaa siemeniä tarvitset pyöreää kukkaistutusta varten, jonka halkaisija on 20 metriä?

Heidän testissään noin 90 % 12–13 vuotiaista osasi ensimmäisen, mutta vain 5 % jälkimmäisen tehtävän. Valtaosa vastasi myös jälkimmäiseen suoraan verrannollisuuden perusteella 800 g, siinäkin tapauksessa, että käytössä oli tilannetta esittävä kuva.

Toisessa ryhmässä 15–16 vuotiaista edelleen vain 20 % vastasi oikein jälkimmäiseen tehtävään. Toisessa kysymyksessä, jossa ei voinut käyttää kaavaa laskuun (kyse oli epäsäännöllisestä geometrisesta muodosta) vain 7 % tästä ryhmästä osasi soveltaa verrannollisuutta oikein. On ilmeistä, että kaavojen käyttö mallitehtävien tyyppisissä tapauksissa ei tuo sellaista geometristä ymmärrystä, joka mahdollistaisi tarkoituksen mukaisen käytön vaihtelevissa tilanteissa.

Eräs tavoite tässä materiaalissa omaksutulla lähestymistavalla on, että parannettaisiin oppilaiden edellytyksiä onnistua edellä mainitun tapaisissa verrannollisuuden käyttöä mahdollistavissa tilanteissa. Kun kulkee tätä polkua, pääsee kuin varkain myös fraktaalien ihmeelisen maailman portille.

Oppimateriaali-luvun teksti on osoitettu ensi sijassa opettajalle. Kuten tässä luvussa on esitetty, on tutkimuksissa todettu, että jo alkuopetuksen oppilaat pystyvät suorittamaan matemaattisia päättelyjä ja puhumaan määritelmistä ja konjekturista kun heitä oikealla tavalla ohjaa. Päättelyn mahdollistamiseksi on avainkäsitteille esitetty tässä monisteessa määritelmät. Opettajan harkintaan jää, millä tavoin hän niiden sisällön oppilaille esittää, ml. käyttääkö sanaa määritelmä vai ei.

Yksi toimintatapa on kirjoittaa määritelmät taululle sopivine esimerkkipiiroksineen. Tehtävät ovat myös lyhyitä, eli ne voi esittää taululle kirjoittamalla, jos käyttää tutkivaa lähestymistapaa yllä olevien ideoiden mukaisesti. Opettaja voi myös lisätä annetun kehyksen ympärille motivoivia elementtejä, sen mukaan, minkä arvioi ryhmälle sopivaksi.

Monisteessa sekä vertailutehtävissä on esitetty Scratchillä tehtyjä ohjelmia ja ohjelmapätkiä. Jos opettaja haluaa esittää joitain niistä luokalle suosittelemme, että hän rakentaa ohjelman raahaamalla palikat paikoileen jakaen näyttönsä koko luokan katsottavaksi. Tällöin hän mallintaa oppilaille ohjelman syntyprosessia eikä vain lopputulosta. Mahdollisesti näin tulee tehtyä joku virhe joka pitää debuggauksen kautta paikallistaa ja korjata. Oppilaat tarvitsevat myös aikaa omaksua ohjelma, joka tulee luonnollisesti kun he näkevät sen syntyvän käsky kerrallaan. Kun ohjelman kirjoittaa on myös mahdollista lisätä sopiviin paikkoihin taukoja ja muuttujien arvojen tulostamisen jotta ohjelman toiminta on helpompi hahmottaa. Esimerkkinä on seuraavalla sivulla esitetty ”animaatio” viisikulmion piirtämisestä, jota voi käyttää Ohjelmointitehtävän 2.10 yhteydessä.

Pedagogiselta kannalta on mainittava erityisesti vertailutehtävät, jotka on koottu käyttöohjeineen lukuun 3. Perusmateriaalissa on esitetty lyhyesti tilanteeseen liittyvä tehtävänanto

[11] The predominance of the linear model in secondary school students' solutions of word problems involving length and area of similar plane figures, 1998

ohjelmonititehtävä -laatikossa. Laatikossa on myös linkki vastaavaan vertailutehtävään, jossa on esitetty, miten Kalle ja Leena ovat ratkaiseet kyseisen ongelman.

Vertailutehtäviä voi käyttää ainakin kahdella tavalla. Ne voi antaa suoraan oppilaille ja pyytää heitä käymään läpi vertailutehtävän sekä vastaamaan siinä oleviin kysymyksiin. Vertailutehtävän voi periaattessa tehdä ilman ohjelmointia, mutta luonnollinen toimintatapa on usein kopioida vertailutehtävän koodi tietokoneelle sekä selvittää sen eri osien toiminnallisuus systemaattisella muokkaamisella.

Vaihtoehtoisesti, opettaja voi itse lukea vertailutehtävän ja sitä kautta saada idea siitä, mitä kullakin tehtävällä haetaan. Oppilaille voi antaa vain monisteen ytimekkään tehtävänannon, ja opettaja voi yrittää löytää luokan eri oppilaiden/ryhmien tuotoksista sellaisia, joita vertailemalla saa tuotua esille tehtävän keskeisen idean kuten valmiissa vertailutehtävässä.

Ensimmäinen lähestymistapa lienee oppilaille helpompi, joten sitä voi käyttää, jos tehtävät tuottavat oppilaille liian suuria vaikeuksia. Mahdollista on myös aloittaa ytimekkäällä tehtävänannolla, ja jonkun ajan kuluttua antaa vertailutehtävä niille oppilaille, jotka eivät pääse tehtävän kanssa eteenpäin.

Tehtävien ajankäyttö riippuu sekä käyttötavasta että luokka-asteesta. Tukimuksissa on todettu, että tutkiva lähestymistapa johtaa syvällisempään ymmärrykseen mutta vie myös enemmän aikaa. Yhden ohjelmointitehtävän käsittely vie 10 minuutista tuntiin.

LUKU 2

Opetusmateriaali

1. Kulmat

Ennen kulmiin liittyvää ohjelmointia oppilaita voi haastaa miettimään kulman käsitettä.

Pohdintatehtävä 1.1. Jos huoneen ovi on raollaan, niin miten voit selittää sanallisesti kuinka paljon se on raollaan niin, että toinen henkilö pystyy kuvauksesi perusteella asettamaan oven samaan asentoon?

Vihjeet ja neuvot (Pohdinta 1.1): Tehtävän voi ratkaista usealla tavalla. Esimerkiksi voi

- piirtää siitä yksinkertainen kuva kahden samasta pisteestä lähtevän janan avulla asettamalla paperi oven saranoiden kohdalle;
- mitata viivottimella oven kulman etäisyyden ovenkarmista;
- sopia omasta mittayksiköstä, esim. jos ovi/ikkuna on kiinni, niin se saa arvon 0. Jos se taas on täysin auki, niin se saa arvon 4.

Tehtävää voi lähestyä myös toiminnallisesti. Voitte pelata peliä, jossa on kysyjä ja arvaaja. Kysyjä piirtää raollaan olevan oven ja antaa toiselle sitä vastaavan numeron. Arvaaja piirtää numeroa vastaavan raollaan olevan oven. Lopuksi verrataan piirrustuksia. Vuorotelkaa arvaajaa.

Harjoitellaan vielä kulmaan liittyvää intuitiota sekä terminologiaa. Tavoitteena olisi käyttää luokittelussa vain kulman suuruutta, ei muita seikkoja kuten sivujanojen pituuksia. Lisäksi pitäisi havaita tietyt erikoistapaukset (*nollakulma*, *suorakulma*, *oikokulma*, *täysi kulma*). Näiden neljän erikoistapauksen väliin jääviä kulmia kutsutaan *teräviksi*, *tylpiksi* ja *kuperiksi*. Huomaa, että oppilaat (sopivalla johdattelulla) voivat keksiä e.m. luokat tai osan niistä, mutta nimiä ei voi tutkivasti selvittää, sillä ne perustuvat mielivaltaiseen sopimukseen, toisin kuin luokat itse.

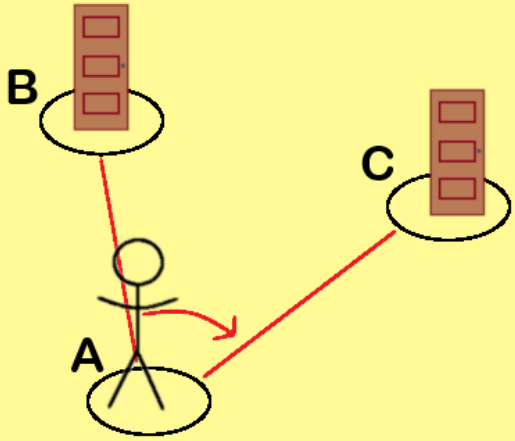
Pohdintatehtävä 1.2. Kulma - luokittelu, katso sivu 25.

Vihjeet ja neuvot (Pohdinta 1.2): Suorakulmassa on neliön muotoisen kulmamerkinnän sijasta poikkeuksellisesti kaareva kulmamerkintä tehtävän mielekkyyden säilyttämiseksi.

Tehtävän vastaukset:
Koveria kulmia ovat kulmat 1, 2, 5, 6, (8), 9 & 10 ja kuperia kulmia ovat kulmat 4 & 7.
Teräviä kulmia ovat kulmat 2, 6, 9 & 10 ja tylppiä kulmia ovat kulmat 1 & 5.
Kulma 3 on oikokulma ja kulma 8 on suorakulma.
Keskenään yhtäsuuria kulmia ovat kulmat 2, 6 & 9, kulmat 1 & 5 ja kulmat 4 & 7.
Kulmien erisuuruuteen vaikuttavat ainoastaan niiden asteluvut. Sivujen pituudet tai kulman suuntautuminen eivät vaikuta kulmien erisuuruuteen.

Määritellään seuraavaksi, mitä tarkasti ottaen tarkoitamme kulmalla. Tässä on mahdollisuus erilaisille valinnolle. Määritelläänkö kulma kahden samasta pisteestä lähtevän janan avulla? Vai olisiko parempi määritellä kulma kolmen pisteen avulla? Entä määritetäänkö kulmalle suunta?

Määritelmä 1.3. Henkilö seisoo pisteessä A ja katsoo kohti pistettä B. *Kulman* $\angle BAC$ *suuruus* kertoo kuinka paljon henkilön on on käännyttävä, jotta hän katsoo kohti pistettä C.



Kun tiedetään mitä tarkoitetaan kulmalla, voidaan siirtyä kysymykseen kulman mittaamisesta. Olet aiemmin tutustunut pituuden mittaamiseen. Tietyn esineen pituutta voi mitata viivottimella tai mittanauhalla. Mittaamiseen voi käyttää erilaisia yksiköitä, kuten senttimetrejä, metrejä, tuumia, jne. Eri yksiköt liittyvät usein toisiinsa; esimerkiksi senttimetri saadaan jakamalla metrin mitan sataan (100) yhtä suureen osaan. Kulman suuruuden määrittämisessä käytetään yleisimmin astetta.

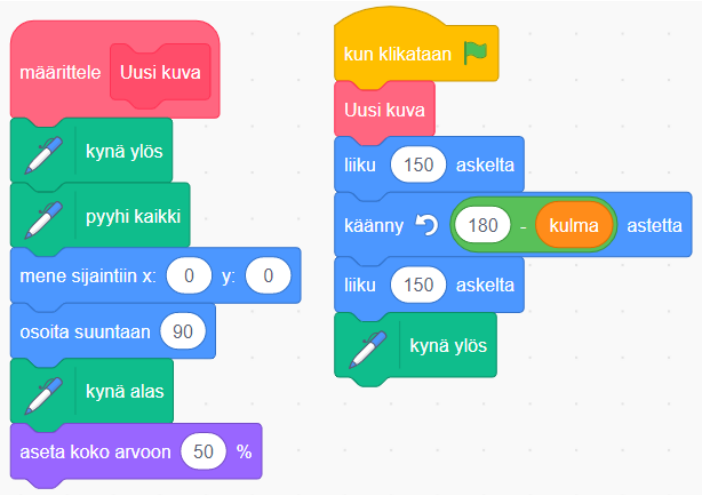
Määritelmä 1.4. Henkilö seisoo pisteessä A ja katsoo kohti pistettä B. Hän kääntyy ympäri yhden kokonaisen kierroksen jolloin hän katsoo taas pistettä B. Tämän kulman suuruus määritellään kolmeksisadaksi kuudeksikymmeneksi asteeksi (360°). Yksi *aste* on siis koko kulman kolmassadas kuudeskymmenesosa ($\frac{1}{360}$).

Esimerkki 1.5. Suoran kulman suuruus on 90°, sillä se on neljäsosa täydestä kulmasta, ja 90 on neljäsosa luvusta 360.

Pohdintatehtävä 1.6. Piirrä paperilla kulma, jonka suuruus on 15° jakamalla täysi kulma (tai suora kulma) sopivaan moneen yhtä suureen osaan. Tarkista tuloksesi vertaamalla sitä kilpikonnagrafiikalla piirrettyyn samansuuruiseen kulmaan.

Vihjeet ja neuvot (Pohdinta 1.6): Tehtävää voi helpottaa tekemällä ensin 45° kulma. Tehtävän voi tehdä piirtämisen sijaan myös taittamalla paperia.

Seuraavaksi harjoitellaan kulman piirtämistä kilpikonnagrafiikan avulla.[1] Scratchissä[2] (versio 3.0) tähän tarvitaan kynä, jonka saa lisättyä painamalla ensin vasemman alareunan Lisää laajennus -nappia ja sitten valitsemalla ilmestyvästä valikosta Kynä-laajennuksen.[3]



Pohdintatehtävä 1.7. Kopioi edelliset koodinpätkät Scratchiin. Mitä kukin käsky tekee ja miten kokonaisuus toimii?

Vihjeet ja neuvot (Pohdinta 1.7): Ohjelmoinnissa voi hyödyntää systemaattista kokeilua: tee kjo

Tässä ohjelmassa aloitetaan määrittelemällä Uusi kuva -lohko, jota käytetään sen jälkeen apuna seuraavassa koodinpätkässä. Kyseistä lohkoa tullaan käyttämään apuna myös monessa tämän monisteen vertailutehtävässä.

Myös ohjelmoinnilla voi harjoitella kulmien luokittelua.

Ohjelmointitehtävä 1.8. Piirrä kilpikonnagrafiikalla erilaisia ja eri kokoisia kulmia.
Vertailutehtävä: Kulma - erisuuruiset kulmat, katso sivu 26.

Huomaa ero tämän kulman määritelmän ja kilpikonnan näkökulman välillä: Kilpikonna kulki pisteestä B pisteeseen A, jolloin se katsoi lähtötilanteessa päinvastaiseen suuntaan kuin kulman määritelmän henkilö. Tästä syystä kilpikonnalle kerrottava kulman suuruus on halutun kulman suplementti.

Ohjelmointitehtävä 1.9. Piirrä kilpikonnagrafiikalla 45 asteen kulma
Vertailutehtävä: Kulma - suplementti, katso sivu 27.

[1] Pythonilla kilpikonnagrafiikkaa voi harjoitella esim. osoitteissa http://trinket.io/turtle ja http://turtleacademy.com.
[2] http://scratch.mit.edu/projects/editor/
[3] Englanninkielinen ohjevideo löytyy osoitteesta http://www.youtube.com/watch?v=eyTfQwSHWEw.

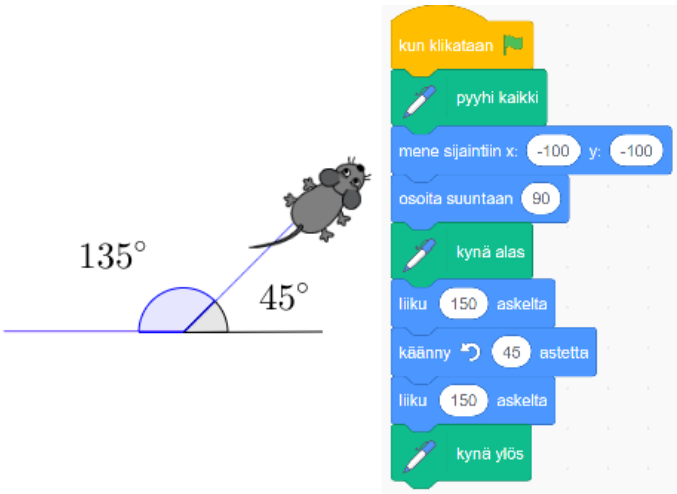
Vihjeet ja neuvot (Ohjelma 1.9): Tässä tehtävässä on tärkeää, että oppilas itse oivaltaa suplementin käytön ja merkityksen. Oppilasta voi johdatella seuraavilla neuvoilla, paljastamatta oikeaa tulosta.

- Minkä kokoisen kulman Leenan ohjelma oikeasti piirtää?
- Miten tilanne muuttuisi, jos Kallen ja Leenan tehtävänä olisikin piirtää 135 asteen kulma?

Lopuksi oppilaalta voi kysyä, että osaisiko tämä nyt tehdä ohjelman, joka piirtää 30 asteen kulman.

2. Säännöllinen monikulmio

Piirretään seuraavaksi monikulmioita kilpikonnagrafiikalla. Kilpikonnagrafiikalla monikulmion sivu piirretään liikuttamalla kilpikonnaa `liiku` x `askelta` -komennolla suoraan eteenpäin (`kynä alas` tilassa). Kulma muodostuu, kun kilpikonna piirtää yhden sivun, kääntyy (`käänny` x `astetta` -komennolla), ja piirtää seuraavan sivun, ks. Kuva 1. Piirtämistehtävissä pitää valita sivujen pituudet ja käännösten suuruudet sopivasti, jotta muodostuu haluttu muoto.



Kuva 1. Kilpikonnagrafiikassa 135° kulmaan tarvitaan $180° - 135° = 45°$ käännös.

Ohjelmointitehtävä 2.1. Piirrä kilpikonnagrafiikalla kolmio, jonka ensimmäinen sivu on 80 ja ensimmäinen käännös 90.

Kokeile toisen sivun pituudeksi 60 ja toiseksi käännökseksi 143.

- Kuinka pitkä kolmannen sivun tulee olla?
- Kuinka suuria ovat kolmion kulmat?

Kokeile muita sivun pituuksia ja kulmia.

Vihjeet ja neuvot (Ohjelma 2.1): Tässä tehtävässä kokeillaan geometrisen kuvion piirtämistä kilpikonnagrafiikalla. Tavoite on ymmärtää eri käskyjen ja tuotetun geometrisen

kuvion välinen yhteys. Kolmannen sivun pituuta ei tarvitse osata laskea tarkasti (se vaatii sini- tai kosiinilauseen käyttöä).

Monessa tapauksessa kuvion piirtäminen onnistuu useilla eri käskyillä. Joissakin tapauksissa toinen käsky saattaa olla toista kätevämpi – tämän arvioiminen on strateegisen joustavuuden ydintä.

Ohjelmointitehtävä 2.2. Piirrä kilpikonnagrafiikalla käyrä, joka yhdistää origon pisteeseen $(100, 100)$.
Vertailutehtävä: Monikulmio - kahden pisteen välinen käyrä, katso sivu 28.

Vihjeet ja neuvot (Ohjelma 2.3): Jos oppilas tekee havainnon, että hiiret katsovat eri suuntiin, kannattaaa tältä kysyä mistä tämä johtuu.

Näiden tapojen lisäksi on vielä esimerkiksi olemassa tapa, missä osoitetaan suuntaan ja liikutaan. Tätä tapaa voidaan käyttää sekä Kallen tavalla liikkua suoraan haluttuun pisteeseen että Leenan tavalla liikkua haluttuun pisteeseen kahden janan avulla.

Neliö koostuu yhtä pitkistä sivuista ja sen jokainen kulma on 90 astetta. Huomioi, että kun piirrät monikulmioita, on kilpikonnan käännösmäärä 180 – haluttu kulma, ks. kuva 1. Kaikki tämän luvun ohjelmointitehtävät voi ratkaista kuvan käskyillä. Joissain tehtävissä on kuitenkin hyödyksi osata käyttää myös toisto- eli looppirakennetta.

Ohjelmointitehtävä 2.3. Piirrä kilpikonnagrafiikalla neliö, jonka sivun pituus on 100.
Vertailutehtävä: Monikulmio - neliö, katso sivu 29.

Vihjeet ja neuvot (Ohjelma 2.3): Jos oppilailla on vähän ohjelmointikokemusta, tehtävää voi lähestyä kopioimalla ensin kuvan esimerkkiohjelma ja muokkaamalla sitä. Oppilaat voivat selvittää kokeilemalla, mitä kukin käsky tekee.

- Jos liikkuu liian vähän, ja hahmo on liian iso, voi jana jäädä hahmon taakse piiloon. (varoitus)
- Piirrä paperille käsin, vaiheittain. (vihje)
- Osaatko tehdä ohjelman, joka piirtää yhden janan? Kaksi janaa? (vihje)
- Montako kertaa pitää kääntyä? (vihje)
- Paljonko pitää kääntyä? (vihje)
- Kumpaan suuntaan pitää kääntyä? (syventävä)

Määritelmä 2.4. Kaksi geometrista kuviota ovat *yhteneviä* jos toisen saa muutettua toiseksi siirtämällä, kiertämällä ja peilaamalla.

Ohjelmointitehtävä 2.5. Piirrä kilpikonnagrafiikalla kaksi neliötä, jotka ovat yhteneviä sekä kaksi kolmiota, jotka ovat yhteneviä.

Vertailutehtävät:
Monikulmio - yhtenevät neliöt, katso sivu 30.
Monikulmio - yhtenevät kolmiot, katso sivu 31.

Vihjeet ja neuvot (Ohjelma 2.5): Tässä, kuten monessa muussakin tehtävässä, voi lähteä liikkeelle muokkamalla aiemmassa kohdassa tehtyä ohjelmaa.

Leena kumittaa ensin piirretyn neliön ylimääräisellä Uusi kuva-lohkolla.

Kolmio-vertailutehtävässä Kallen koodi tuottaa käännetyn kolmion ja Leenan peilatun kolmion.

Vihjeet ja neuvot (Yhtenevyydestä): Edellinen määritelmä-ohjelmointitehtävä pari muodostaa kokonaisuuden, jossa ohjelmaa käytetään määritelmän havainnollistamiseen ja siten parempaan ymmärtämiseen.

- Mikä esimerkkiohjelmassa/neliötehtävässä määrää, mihin kulma/neliö piirtyy? (vihje)
- Mikä esimerkkiohjelmassa/neliötehtävässä määrää, mihin suuntaan kulma/neliö piirtyy? (vihje)
- Miten peilaamisen voi toteuttaa ohjelmassa? (syventävä)
- Voitko ohjelmalla piirtää neliön joka ei ole alkuperäisen kanssa yhtenevä? (syventävä)

Tasasivuisessa kolmiossa kulman suuruus on $\frac{180}{3} = 60$ astetta. Kuinka paljon kilpikonnan tulee kääntyä, jotta sisäkulmaksi tulee 60 astetta?

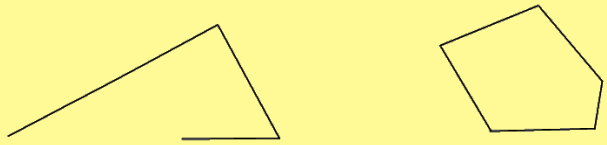
Ohjelmointitehtävä 2.6. Piirrä kilpikonnagrafiikalla tasasivuinen kolmio, jonka sivu on 100.

Vihjeet ja neuvot (Ohjelma 2.6): Ohjeista oppilaita tarkistamaan (kirjasta/verkosta) mitä tasasivuinen kolmio tarkoittaa, jos eivät sitä muista.

- Tee ensin ohjelma, joka piirtää kolmen janan käyrä, ja murehdi kulmista myöhemmin. (vihje)
- Paljonko pitää kääntyä, että tulee kolmio? (vihje)
- Kumpaan suuntaan pitää kääntyä? (vihje/syventävä)
- Kuinka paljon kääntymistä tuli yhteensä, jos lasket kaikki käännöksissä olevat kulman suuruudet yhteen? (syventävä)
- Myös kolmion osalta voi tarkastella yhteyttä yhtenevyyden kanssa. (syventävä)

Seuraavaksi jatketaan yleisempiin monikulmioihin.

Määritelmä 2.7. *Murtoviiva* on käyrä, joka koostuu äärellisestä määrästä janoja (vasen kuva). *Monikulmio* on sulkeutuva murtoviiva eli se alkaa ja päättyy samaan pisteeseen (oikea kuva).



Ohjelmointitehtävä 2.8. Piirrä kilpikonnagrafiikalla kaksi murtoviivaa, joista toinen on sulkeutuva ja toinen ei ole.

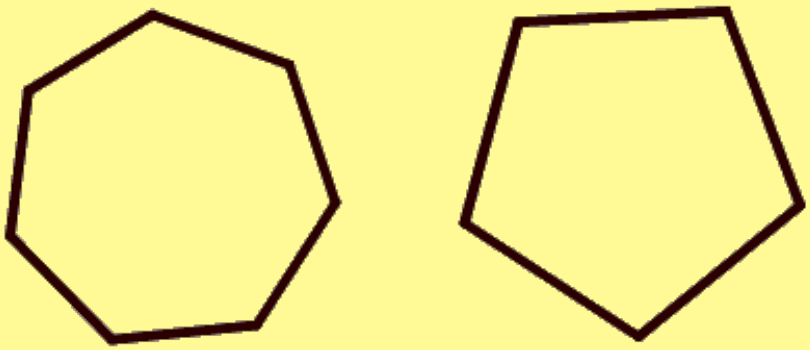
Vertailutehtävät:
Monikulmio - sulkeutuva murtoviiva (1), katso sivu 32.
Monikulmio - sulkeutuva murtoviiva (2), katso sivu 33.

Vihjeet ja neuvot (Ohjelma 2.8): Oppilaalta voi kysyä, miten kolmion kulmien summa liittyy Kallen ratkaisuun. Leenan ratkaisussa viimeinen kääntyminen on turha.

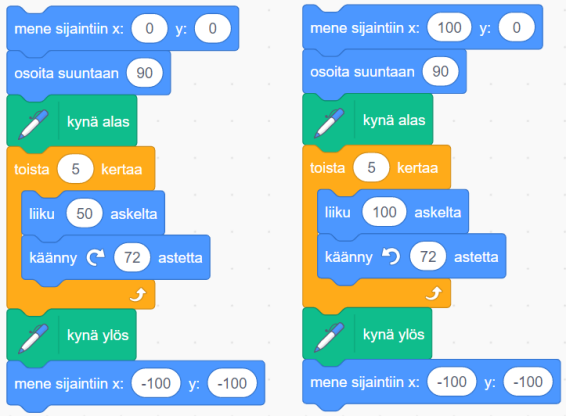
Vihjeet ja neuvot (Murtoviivasta): Edellinen määritelmä-ohjelmointitehtävä pari muodostaa kokonaisuuden, jossa ohjelmaa käytetään määritelmän havainnollistamiseen ja siten parempaan ymmärtämiseen.

- Tehtävässä on paljolti kyse siitä, että osaa yhdistää uuteen käsitteeseen jo tehtyjä ohjelmia. Miten voit käyttää hyväksi aiemmin tekemiäsi ohjelmia? (vihje)
- Kun ymmärtää yhteydet, voi tuntua siltä, että tehtävässä ei ole mitään tekemistä. Tämä olisi hyvä kokemus matematiikan syvällisemmän oppimisen kannalta. (syventävä)
- Sulkeutuva murtoviiva voi olla vaikeampi piirtää. Yhtenevyystehtävässä kysymys oli miten neliön aloittaa tietystä pisteestä, nyt pitäisi osata lopettaa tiettyyn pisteeseen. Analogia voi auttaa oppilaita erään ratkaisun keksimiseen. (vihje)

Määritelmä 2.9. *Säännöllinen monikulmio* on monikulmio, jonka kaikki kulmat ovat yhtä suuria ja kaikki sivut yhtä pitkiä.



Säännöllisessä viisikulmiossa jokainen sivu on yhtä pitkä ja jokainen kulma yhtä suuri. Koska yhteensä kilpikonnan tulee kääntyä 360 astetta, niin yhden käännöksen suuruus on $\frac{360}{5} = 72$ astetta.[4] Seuraavalla sivulla on kaksi ohjelmaa, jotka piirtävät erikokoisia viisikulmioita.



Ohjelmointitehtävä 2.10. Tarkastellaan säännöllisten monikulmioiden piirtämistä. Valitse ensin sopiva sivun pituus ja määritä sitten käännöksen suuruus. Miten saisit aikaan

- säännöllisen kuusikulmion?
- säännöllisen 8-kulmion?
- säännöllisen 13-kulmio?

Vertailutehtävä: Monikulmio - epäsäännöllinen monikulmio, katso sivu 34.

Vihjeet ja neuvot (Ohjelma 2.10):

- Miten voit hyödyntää aikaisempia ohjelmia, neliötä ja kolmiota? (vihje)
- Mihin sivun pituus vaikuttaa? (vihje)
- Mikä on oikea kulma? Kuinka paljon kilpikonnan tulee kääntyä? (vihje)
- Ei säännöllinen monikulmio voi olla vaikeampi piirtää. Tässä toimii sama analogia kuin sulkeutuvan murtoviivan vihjeessä. (vihje)

Voit hahmottaa seuraavan tehtävän ratkaisua paperilla tai kilpikonnagrafiikalla.

Pohdintatehtävä 2.11. Anna esimerkki monikulmiosta, joka ei ole säännöllinen, mutta joko

(1) sen kaikki kulmat ovat yhtä suuria, tai
(2) sen kaikki sivut ovat yhtä suuria.

Vihjeet ja neuvot (Pohdinta 2.11): Tehtävän ratkaisua voi helpottaa, jos käyttää aiemmin tehtyjä ohjelmia apuna. Tällöin on hyvä, jos ohjelmassa silmukat kirjoitetaan auki, esimerkiksi kuusikulmiossa toistetaan samat käskyt (liiku+käänny) kuusi kertaa, ja sitten lähtee jotakin niistä muokkaamaan. Tämä toimii hyvin yhdessä seuraavien vihjeiden kanssa:

[4] Johdannon luvussa 4 on versio ohjelmasta jossa taukojen avulla selkeämmin havainnollistetaan ulkokulmien summa.

- Valitse jokin sivun pituus ja piirrä erilaisia monikulmioita käyttäen vain valittua sivun pituutta. Miltä tilanne näyttää? (vihje)
- Valitse jokin kulma ja piirrä erilaisia monikulmioita käyttäen vain valittua kulmaa. Miltä tilanne näyttää? (vihje)
- Jos oppilaat tarjoavat pelkkää ohjelmaa tai piirrettyä kuvaa, haasta heidät perustelemaan asia myös matemaattisesti: mistä tiedät, että kulmat/sivut ovat täsmälleen, eikä vaan melkein, yhtä suuria? (syventävä)

Ohjelmointitehtävässä 2.10 huomasimme, että säännöllinen monikulmio lähestyy ympyrää, kun sivujen lukumäärä kasvaa.

Pohdintatehtävä 2.12. (Syventävä tehtävä) Miten voi perustella, ettei monikulmio koskaan ole ympyrä, vaikka sivujen määrä olisi kuinka suuri?

Vihjeet ja neuvot (Pohdinta 2.12): Kun sivumäärä kasvaa, näyttää säännöllinen monikulmio ympyrältä. Tämä tehtävä korostaa täsmällisen päättelyn merkitystä matematiikassa, ja luo yhteyden seuraavaan kokonaisuuteen (Mystinen luku π).

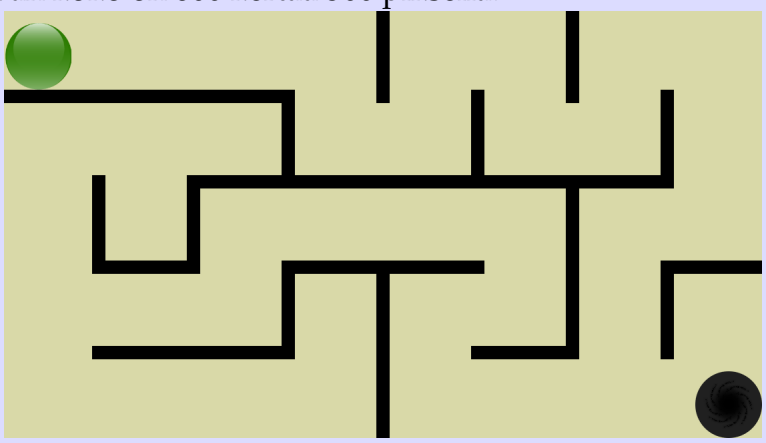
- Jos oppilaat esittävät epämääräisen perustelun, kysy heiltä, ovatko he varmoja. On tärkeä esittää kysymys riippumatta siitä, onko heidän perustelu oikein vai ei. (syventävä)
- Miten ympyrä määritellään? (vihje)

3. Mystinen luku π

Koulussa meille on opetettu, että r säteisen ympyrän ympärysmitta on $2\pi r$ ja sen pinta-ala on πr^2. Mutta mitä tämä tarkemmin ottaen tarkoittaa ja miten nämä väittämät voi perustella?

Jotta voitaisiin perustella väite tarkasti, pitää meidän tietää, mitä tarkoitetaan käyrän pituudella.

Pohdintatehtävä 3.1. Kuinka kaukana vihreä ja musta piste ovat toisistaan oheisessa labyrintissa? Kuvan koko on 800 kertaa 500 pikseliä.



Vertailutehtävä: Mystinen luku π - kahden pisteen väinen etäisyys, katso sivu 35.

Vihjeet ja neuvot (Pohdinta 3.1): Matkan voi mitata monella tavalla. Linnuntietä pitkin matka selviää Pythagoraan lauseella. Labyrintissa pitää liikkua 3 askelta vasemmalle, askel alas, askel vasemmalle, askel ylös, jne. Jokaisen askeleen pituus on 100 pikseliä. Jos merkitään näin muodostuva polku, saadaan murtoviiva, jonka pituus on 1700 piskeliä.

Määritelmä 3.2. *Murtoviiva* on käyrä, joka koostuu äärellisestä määrästä janoja. Murtoviivan *pituus* on kyseisten janojen pituuksien summa.

Määritelmää kannattaa selventää esimerkeillä, kuten seuraavassa ohjelmointitehtävässä tehdään.

Ohjelmointitehtävä 3.3. Seuraava ohjelma piirtää erään murtoviivan, jonka pituus on 150 (yksikköä):

Kirjoita ohjelma, joka piirtää murtoviivan ja ilmoittaa Määritelmän 3.2 mukaisesen pituuden.

Pohdintatehtävä 3.4. Mystinen luku π - murtoviivan pituus, katso sivu 36.

Vihjeet ja neuvot (Pohdinta 3.4): Ensimmäisen murtoviivan pituus on 230 (yksikköä) ja jälkimmäisen murtoviivan pituus 280 (yksikköä).

Muun käyrän pituus saadaan määriteltyä käyttämällä murtoviivaa joka kulkee lähellä käyrää. Tarkemmin ottaen:

Määritelmä 3.5. Käyrä voidaan *approksimoida* murtoviivalla valitsemalla käyrältä äärellinen määrä pisteitä järjestyksessä ja yhdistämällä peräkkäiset pisteet janoilla.

Käyrän *pituus* saadaan käyrää approksimoivien murtoviivojen pituuksista kun murtoviivan janojen pituus pienenee kohti nollaa.

Edellinen määritelmä perustuu siihen ajatukseen, että tiedämme mikä on janan pituus, ja muun käyrän pituus määritetään sitä käyttämällä. Kyseessä on approksimaatioon perustuva määritelmä, missä monimutkaista objektia (käyrä) arvioidaan käyttämällä suurta määrää yksinkertaisia objekteja (jana). Tämä on tyyppillinen lähestymistapa myös tietotekniikassa, sillä tietokoneelle on ominaista käsitellä juuri suuria määriä yksinkertaisia objekteja.

Matematiikassa määritelmä on lähtökohta, joka kertoo, mitä jokin käsite tarkoittaa. Siksi määritelmiä ei voi todistaa tai perustella matemaattisesti. Määritelmien järkevyyttä voi kuitenkin arvioida tarkastelemalla, antaako se tutuissa tapauksissa sen tuloksen, jota odottamme.

Määritelmässä 3.5 annetaan käyrän pituudelle määritelmä approksimoivan murtoviivan avulla. Määritelmä on matemaatisesti pätevä, mutta siihen liittyy useita hienovaraisuuksia, joita ei tässä ruodita. Näitä ovat ainakin:

- Mitä tarkoitetaan käyrällä?
- Antaako approksimaatioprosessi aina saman lukuarvon, riippumatta siitä, miten pisteet valitaan käyrältä?
- Määritelmä on prosessiluonteinen (mikä sopii hyvin yhteen tietokoneimplementaation kanssa), joten siinä vältetään raja-arvon käsitettä.

Ohjelmointitehtävä 3.6. Kirjoita ohjelma, joka piirtää säännöllisen

- kuusikulmion
- kahdeksankulmion
- kymmenenkulmion

ja ilmoittaa Määritelmän 3.2 mukaisesen pituuden.

Vertailutehtävä: Mystinen luku π - piirtoalue, katso sivu 37.

Vihjeet ja neuvot (Ohjelma 3.6): Jos ryhmällä on vaikeuksia, niin tehtävä kannattaa jakaa piirtämiseen ja laskemiseen. Piirtämisessä voi hyödyntää aikaisemmassa luvussa tehtyjä ohjelmia.

Vertailutehtävässä Leenan monikulmiossa Scratchin piirtotila loppuu kesken. Hiiri luulee etenevänsä, mutta todellisuudessa se kulkee vain rajaa päin. Kuviosta tulee tämän johdosta epämuodostunut. Ongelma poistuu, kun sivun pituutta pienennetään.

Edellisen ohjelmointitehtävän monikulmiot lähestyvät ympyrää, joka voidaan täsmällisesti määritellä seuraavasti.

Määritelmä 3.7. *Ympyrä* on tasojoukko jonka pisteet ovat yhtä etäällä (eli säteen päässä) annetusta pisteestä. Ympyrän *ympärysmitta* on sen käyrän pituus, joka kiertää ympyrää yhden kierroksen.

Seuraavaksi laaditaan ohjelma, joka approksimoi ympyrän piiriä eli ympärysmittaa. Tässä pitää tukeutua edellisiin määritelmiin.

Ohjelmointitehtävä 3.8. Kirjoita ohjelma, joka approksimoi Määritelmän 3.5 mukaisesti ympyrän ympärysmittaa. Ohjelman tulee tulostaa approksimaatiolla ympärysmitan ja ympyrän halkaisijan suhde.

Voit lähteä liikkeelle ohjelmointitehtävästä 3.6 ja käyttää monikulmiota ympyrää approksimoivana murtoviivana.

Vertailutehtävä: Mystinen luku π - monikulmion halkaisija, katso sivu 38.

Vihjeet ja neuvot (Ohjelma 3.8):

(1) Hahmota ensin paperille, mitä haluat tehdä.
(2) Jos haluat ohjelmassa käyttää jotain tietoa, pitää se laittaa talteen tallentamalla se muuttujaksi.

Edellisen ohjelmointitehtävän tuloksen voi havaita, että ympyrän ympärysmitan ja säteen suhde ei vaikuta riippuvan säteestä. Tämä empiirinen havainto ei kuitenkaan ole matemaattisesti pätevä peruste. Määritelmiä käyttäen voidaan nyt osoittaa, että tuttu kaava $2\pi r$ pätee. Siihen tarvitaan vielä yksi määritelmä.

Määritelmä 3.9. Luku π määritellään sellaisen ympyrän ympärysmittana, jonka säde on $\frac{1}{2}$, eli läpimitta on 1.

Pohdintatehtävä 3.10. (Syventävä tehtävä) Ohjelmointitehtävässä 3.8 nähtiin, miten ympyrän ympärysmittaa voidaan laskea monikulmioiden avulla.

(1) Piirrä sama monikulmio kahden erisäteisen ympyrän sisälle.
(2) Etsi kahdesta monikulmiosta yhdenmuotoisia kolmioita.
(3) Osoita, että kahden ympyrän ympärysmitan ja säteen suhde on sama.
(4) Osoita, että ympyrän, jonka säde on $r > 0$, ympärysmitta on $2\pi r$.

LUKU 3

Vertailutehtävät

Vertailukeskustelun käyminen

Miksi keskustella matematiikasta?

- Syventää oppijoiden ymmärrystä matemaattisesta sisällöstä
- Vahvistaa oppijoiden osallisuutta ja kiinnostusta matematiikkaa kohtaan

Mitä opettaja voi tehdä mahdollistaakseen hyvän matemaattisen keskustelun?

Ennen keskustelua:

- Ratkaise ajatuksella tehtävät, joista keskustellaan
- Ennakoi oppijoiden vastauksia, virheitä ja haasteita
- Suunnittele millaisia kysymyksiä kysyt ja millaisia jatkotehtäviä voit antaa

Keskustelun aikana:

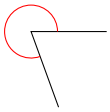
- **Kysy** avoimia kysymyksiä, esimerkiksi seuraavanlaisia kysymyksiä käyttämällä voit yrittää herättää ja jatkaa keskustelua:
  - Oletko samaa mieltä Leenan kanssa? Miksi?
  - Voitko tiivistää, mitä Kalle sanoi?
  - Voisitko antaa toisen esimerkin?
  - Kuvailisitko vielä vähän tarkemmin ajatustasi?
  - Mitä tarkoitat sillä, että ... ?
  - Miten sinä teit sen?
  - Mikä saattaisi olla hämmentävää tässä esimerkissä?
- **Toista** ja **tiivistä** oppijoiden ideoita (tai pyydä toista oppijaa tekemään se), keskustelun vauhdittamiseksi ja selkeyttämiseksi
  - Kuulin sinun sanovan, että . . . . Ymmärsinkö oikein? / Sitäkö tarkoitit?
  - Tarkoitatko, että ...
  - Haluaisin ymmärtää vielä vähän paremmin mitä tarkoitat. Voitko selittää vielä uudestaan?
- **Ohjaile** keskustelun kulkua siten, että useat ihmiset ja näkökulmat ovat läsnä
- **Osallista** mahdollisimman monia oppijoita keskusteluun
  - Houkuttele keskusteluun mukaan oppijoita, joiden on vaikeaa osallistua (esim. arpomalla seuraavan vuoro tai heittelemällä puheenvuoro-pehmolelua, jonka saa heittää kenelle haluaa)
- **Vastuuta** oppijoita toisten ymmärtävään kuuntelemiseen.
  - Timo, voitko selittää Emilian ajatuksen vielä omin sanoin?
- **Anna tunnustusta** toivotusta käytöksestä: kuuntelemisesta, oman ajattelun kuvailusta, perustelemisesta, omaperäisyydestä ym.

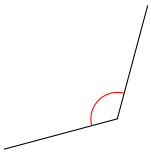
Kulmien tyyppien luokittelu

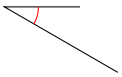
1.

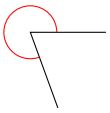
2.

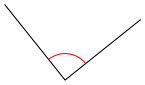
3.

4. 

5. 

6. 

7. 

8. 

9.

10.

1. Mitkä kulmista ovat keskenään yhtäsuuria?
2. Mitkä edellisistä kulmista ovat koveria? Entä kuperia?
3. Mitkä koverista kulmista ovat teräviä kulmia? Entä tylppiä?
4. Millaisia kulmia jäljelle jääneet kulmat ovat?
5. Mitkä asiat vaikuttavat kulmien erisuuruuteen? Mitkä asiat eivät vaikuta?
6. Mitä opit näistä vertailuista?

Kalle ja Leena ovat piirtäneet erisuuruiset kulmat eri tavoilla.

Kallen tapa	Leenan tapa

Piirrän kulman liikkumalla, kääntymällä ja liikkumalla uudestaan.

Uusi kuva -aliohjelma vaikuttaa kätevältä, joten aloitan sillä. Sitten piirrän kulman, joka koostuu kahdesta suorasta ja joiden väliin tulee käännös.

Piirrän toisen kulman vähentämällä liikuttavia askeleita viiteenkymmeneen.

Muodostan toisen kulman jatkamalla edellistä koodia vastaavalla tavalla, mutta muokkaamalla käännöksen suuruutta.

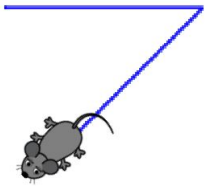
1. Millaiset tulosteet Kallen koodit tuottavat? Entä Leenan?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Kumman tapa on oikein? Miksi?
4. Mitä opit tästä vertailusta?

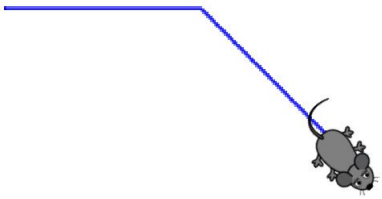
Kalle ja Leena ovat piirtäneet 45° kulman eri tavoilla kilpikonnagrafiikalla.

Kallen tapa	Leenan tapa

Kokeilemalla huomasin, että laittamalla käännösmääräksi 135 saadaan 45 asteen kulma.

Tässä on tuloste.





Koska pitää piirtää 45 asteen kulma, niin laitan käännösmääräksi 45.

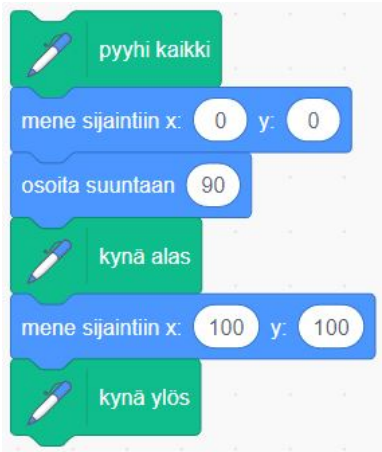
Minusta tämä ei näytä 45 asteen kulmalta.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Kumman tapa on oikein? Miksi?
3. Mitä etuja Kallen tavassa on? Entä Leenan?
4. Mitä opit tästä vertailusta?

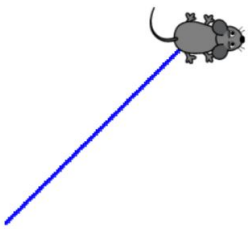
Kalle ja Leena ovat piirtäneet käyrän origosta pisteeseen $(100, 100)$ kilpikonnagrafiikalla eri tavoilla.

Kallen tapa	Leenan tapa

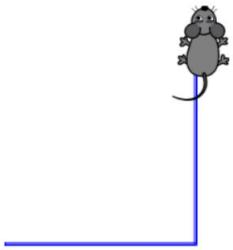
Löysin käskyn, joka liikuttaa hiiren suoraan haluttuun kohtaan.



Jostakin syystä hiiri ei katso piirtämäni janan suuntaisesti.



Liikun ensin sata askelta oikealle ja sitten sata askelta ylöspäin.



Tuloste on tässä. Yhdistin vaaditut pisteet kahden janan avulla.

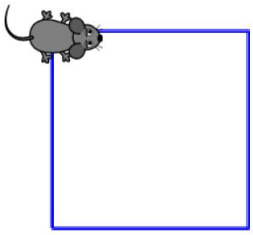
1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Kumman tapa on parempi? Miksi?
3. Mitä etuja Leenan tavassa on? Entä Kallen?
4. Olisiko vielä jotain muuta tapaa?
5. Mitä opit tästä vertailusta?

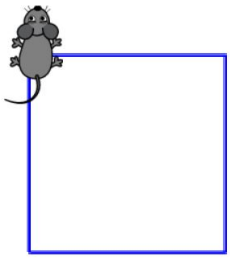
Kalle ja Leena ovat piirtäneet neliön kilpikonnagrafiikassa eri tavoilla.

Kallen tapa	Leenan tapa

Huomasin, että kaksi käskyä toistuvat, kun neliön piirtää kilpikonnagrafiikalla. Siksi teen nämä kaksi käskyä sisältävän toistorakenteeen.

Tässä on valmis neliö.





Ajattelin neliön piirtämistä sivu kerrallaan. Ensin liikun oikealle, sitten alaspäin, sitten vasemmalle ja viimeiseksi ylöspäin.

Lopputulos on tällainen.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Kumman tapa on parempi? Miksi?
4. Mitä etuja Leenan tavassa on? Entä Kallen?
5. Mitä opit tästä vertailusta?

Kalle ja Leena ovat piirtäneet yhtenevät neliöt eri tavoilla kilpikonnagrafiikalla.

Kallen tapa	Leenan tapa

Kopioin edellisessä vertailutehtävässä tekemäni koodin.

Sitten kirjoitan koodin toiselle neliölle eri lähtöpisteellä ja lopuksi liitän nämä kaksi koodinpätkää yhteen.

Kalle oli löytänyt edellisessä tehtävässä helpon tavan piirtää neliöitä.

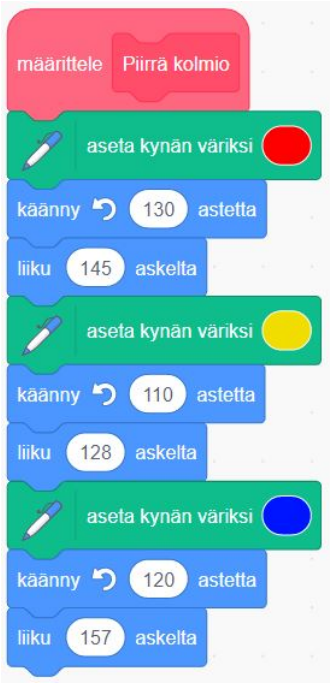
Käytän toiselle neliölle samaa neliönpiirtämisrakennetta eri lähtöpisteellä ja lähtösuunnalla.

1. Pystytkö päättelemään millaiset tulosteet Kallen koodit tuottavat? Entä Leenan?
2. Miten Kalle ratkaisi tehtävän? Entä Leena?
3. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
4. Kumman tapa on oikein? Miksi?
5. Mitä opit tästä vertailusta?

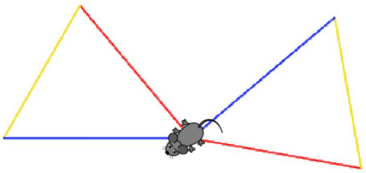
Kalle ja Leena ovat saaneet valmiin koodin kolmiolle ja piirtäneet sille yhtenevät kolmiot eri tavoilla kilpikonnagrafiikalla.

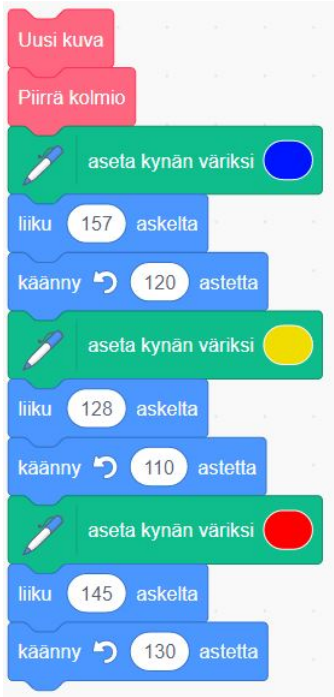
Kallen tapa	Leenan tapa

Teen saamastani koodista uuden lohkon, jonka nimeksi annan Piirrä kolmio. Sitten käytän sitä kahdesti, mutta muutan välissä hiiren lähtösuuntaa.

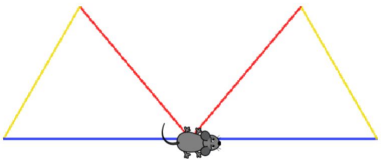


Eri väriset sivut helpottavat koodin hahmottamista.





Määrittelen saamastani koodista lohkon samalla tavalla kuin Kalle. Sitten piirrän sille peilikuvan piirtämällä sen sivut päinvastaisessa järjestyksessä.



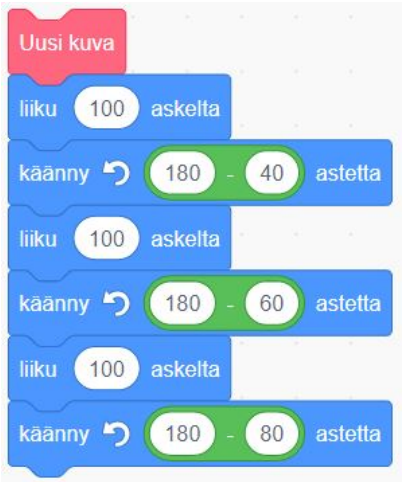
Lopputulos on peilikuva alkuperäisestä kolmiosta.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Miksi nämä kummatkin tavat toimivat?
3. Mitä opit tästä vertailusta?

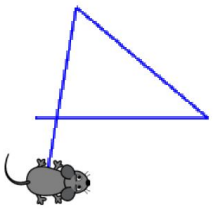
Kalle ja Leena ovat piirtäneet sulkeutuvat murtoviivat eri tavoilla kilpikonnagrafiikalla.

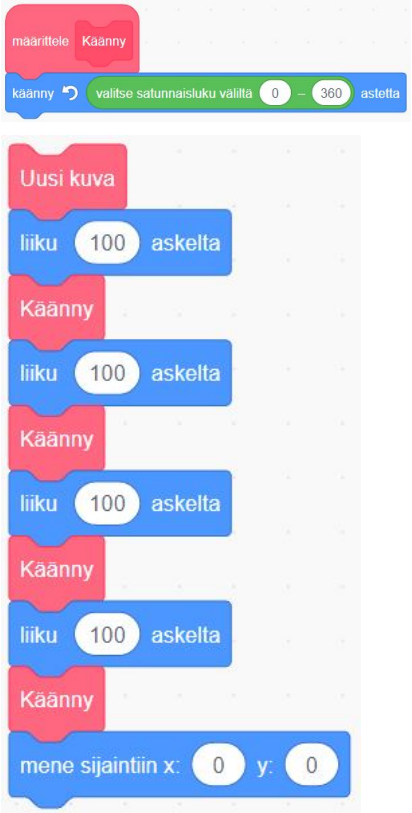
Kallen tapa	Leenan tapa

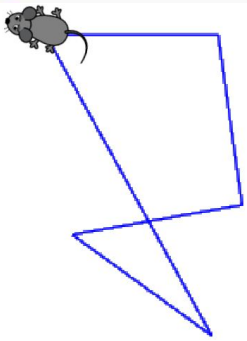
Käytän murtoviivan piirtämisen apuna kolmion kulmien summaa $40 + 60 + 80 = 180$. Kirjoitan käännöksien suplementit tällaisien miinuskäskyjen avulla, jotta koodin ideasta tulee selkeämpi.



Mutta miten saan kolmion sivut sopivan pituisiksi?







Aloitan liikkumalla satunnaisesti neljä janaa eteenpäin. Sen jälkeen palaan alkupisteeseen mene sijaintiin -komennon avulla, sillä en tiedä millä muulla tavalla pääsisin takaisin aloituspisteeseen.

Lopputuloksessa kuvion sivut leikkaavat toisiaan, mutta se ei ole ongelma.

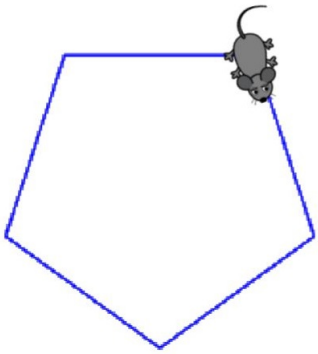
1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Kumman tapa on oikein? Miksi?
3. Mitä opit tästä vertailusta?

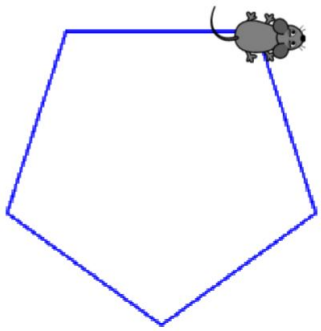
Kalle ja Leena ovat piirtäneet sulkeutuvat murtoviivat eri tavoilla kilpikonnagrafiikalla.

Kallen tapa	Leenan tapa

Lisäsin tuttuun neliön piirtämiskoodiin yhden sivun ja kokeilemalla löysin viisikulmiolle sopivan kulman suuruuden.

Tässä on tuloste.





Mietin, että miten neliönkoodista saisi muokattua koodin viisikulmiolle. Kokeilemalla löysin sopivan kulman.

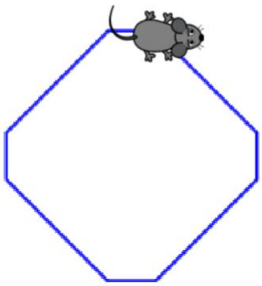
Kuva näyttää oikealta.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Kumman tapa on oikein? Miksi?
4. Mitä opit tästä vertailusta?

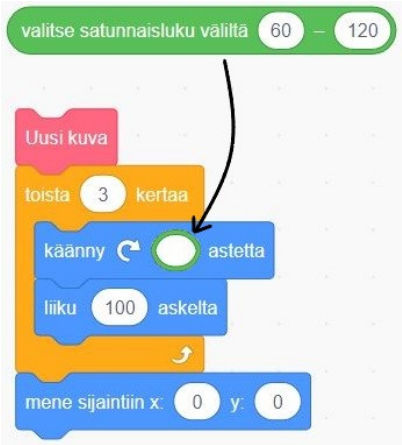
Kalle ja Leena ovat ovat piirtäneet epäsäännölliset monikulmiot eri tavoilla kilpikonnagrafiikalla.

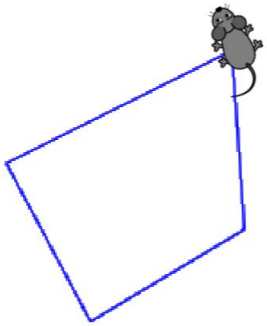
Kallen tapa	Leenan tapa

Piirrän kahdeksankulmion, jossa on kahden pituisia sivuja.



Tulos on "puolisäännöllinen" monikulmio.



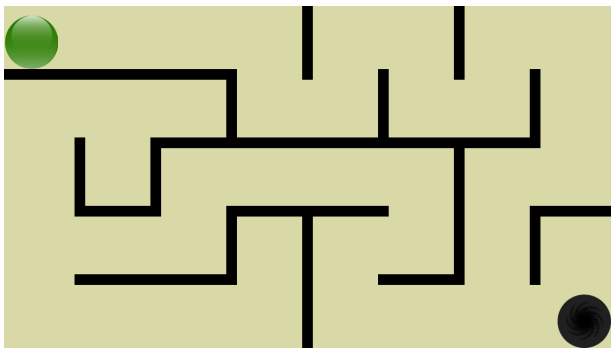


Koska tavoitteena on piirtää epäsäännöllinen monikulmio, niin käytän apuna käskyä, joka valitsee satunnaisluvun halutulta väliltä.

Lopputulos vaihtelee. Tällä suorituskerralla lopputulos on tällainen.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Miksi nämä kummatkin tavat toimivat?
4. Mitä opit tästä vertailusta?

Kalle ja Leena ovat laskeneet labyrintissä olevien vihreän ja mustan pisteen välisen etäisyyden eri tavoilla.

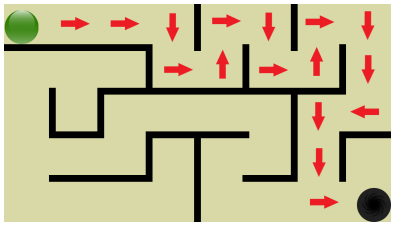


Kuvan koko on 800 kertaa 500 pikseliä.

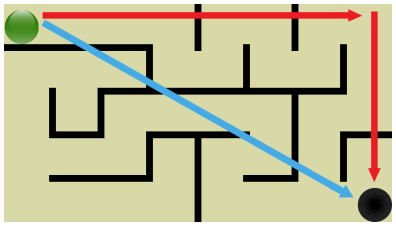
Kallen tapa	Leenan tapa

Liikun vihreältä pisteeltä mustalle pisteelle sadan pisteen mittaisissa askeleissa.

Askeleita tuli yhteensä seitsemäntoista. Vastaus on tässä.



1700



$\sqrt{800^2 + 500^2} = z^2$

943

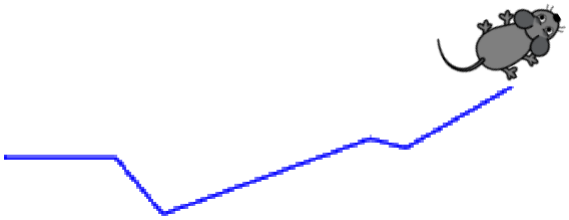
Lasken pisteiden välisen etäisyyden käyttämällä Pythagoraan lausetta $x^2 + y^2 = z^2$.

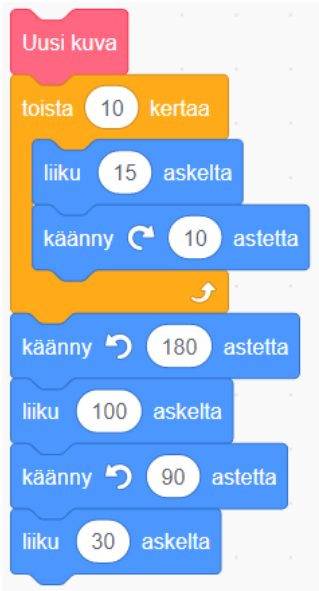
Sijoitan kateettien x ja y paikalle labyrintin mitat ja ratkaisen hypotenuusan z.

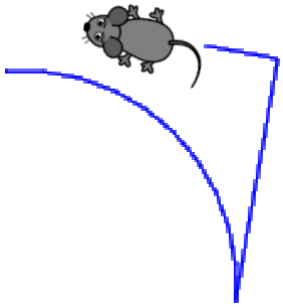
Saan vastauksen.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Kumman tapa on järkevämpi? Miksi?
3. Mitä opit tästä vertailusta?

Murtoviivan pituus





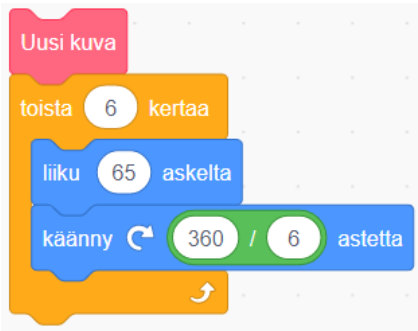


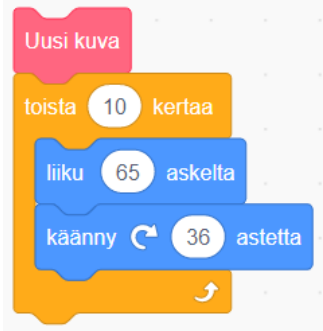
1. Kuinka pitkiä edelliset murtoviivat ovat?

Kalle ja Leena ovat ovat piirtäneet erikokoiset monikulmiot.

Kallen tapa	Leenan tapa

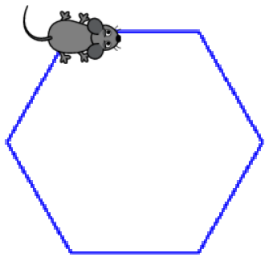
Piirrän monikulmion käyttämällä loop-rakennetta.

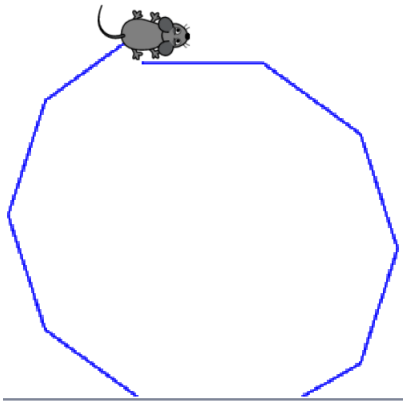




Käytän vanhaa tuttua loop-rakennetta monikulmion piirtämiseen.

Tuloste on tässä.





Jostakin syystä hiiri ei palaa takaisin aloituspisteeseen.

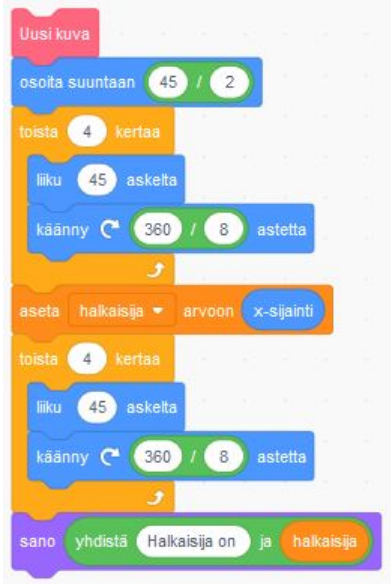
1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Miksi Kallen ratkaisu toimii mutta Leenan ei?
4. Mitä opit tästä vertailusta?

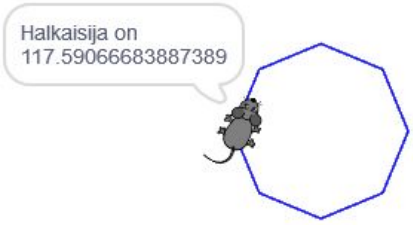
Kalle ja Leena ovat määrittäneet kahdeksankulmion vastakkaisten pisteiden välisen etäisyyden kahdella eri tavalla.

Kallen tapa	Leenan tapa

Tein ohjelman joka piirtää puolikkaan säännöllisen kahdeksankulmion, koska silloin hahmo on päätynyt lähtöpisteen vastakkaiseen pisteeseen.

Hahmoni ilmoitti lopuksi säännöllisen kahdeksankulmion vastakkaisten pisteiden välisen etäisyyden.





Tein ohjelman, joka piirtää säännöllisen kahdeksankulmion ja tallentaa matkan puolivälissä hahmon x-koordinaatin muuttujan 'halkaisija' arvoksi.

Hahmoni ilmoitti lopuksi säännöllisen kahdeksankulmion vastakkaisten pisteiden välisen etäisyyden.

1. Miten Kalle ratkaisi tehtävän? Entä Leena?
2. Mitä samanlaista ja mitä erilaista heidän tavoissaan on?
3. Kumman tapa on oikein? Miksi oikea tapaa toimii? Miksi väärä ei toimi?
4. Mitä hyviä ja mitä huonoja puolia ohjelmissa on?
5. Mitä opit tästä vertailusta?

LIITE A

Korkeampi ulottuvuus ja fraktaalit

4. Pinta-ala, eli 2-alotteinen tilavuus

Seuraavaksi käsittelemme pinta-alaa. Edellisessä kappaleessa lähtökohtana oli janan pituus, joka oletettiin tunnetuksi. Nyt puolestaan lähdetään neliön pinta-alasta, ts. neliön, jonka sivun pituus on a, pinta-ala on a^2. Esimerkiksi, jos neliön sivun pituus on 1 cm, niin sen pinta-ala on 1 cm^2.

Määritelmä 4.1. Asetetaan tasoon säännöllinen ruudukko, jossa ruudun koko on $a \times a$. Tasoalueen E, *a-pinta-ala* määritellään laskemalla yhteen niiden ruutujen pinta-alat, jotka eivät ole kokonaan tasoalueeseen E ulkopuolella.

Tasoalueen E *pinta-ala* määritellään a-pinta-alojen arvona kun a pienenee kohti nollaa.

Tehdään seuraavaksi ohjelma, jota voidaan hyödyntää myöhemmmin pinta-alaa laskettaessa.

Ohjelmointitehtävä 4.2. Tee ohjelma, joka piirtää ruudukon annetulla ruudun sivulla.

Vihjeet ja neuvot (Ohjelma 4.2):

(1) Hahmotelkaa ruudukosta kuva paperille.
(2) Mitä kuviota toistetaan?
(3) Mitkä muuttujat ohjelmalle tulee antaa?

Harjoituksena voit tarkistaa määritelmän mielekkyyden laskemalla sen avulla suorakaiteen pinta-alan.

Ohjelmointitehtävä 4.3. Lataa valmis ohjelma `suorakaide.sb2` ja aja se. Sinun pitäisi nähdä ruudukko ja suorakaide S. Laske (ruudusta katsoen) kuinka monta ruutua eivät ole kokonaan S:n ulkopuolella ja määritä S:n a-pinta-ala. Toista tämä muutamalla muuttujan a eri arvoilla. Miten a-pinta-alat käyttäytyvät, kun a pienenee?

Vihjeet ja neuvot (Pohdinta 4.3): Tee taulukko, johon merkitään

(1) käytetyn neliön sivu b,
(2) pinta-ala b^2,
(3) suorakaiteen b-pinta-ala ja
(4) suorakaiteen a-pinta-ala.

Kasvata ja pienennä lukua b esim. $b = 2a$ ja $a/2$. Miten a-pinta-ala muuttuu?

Ohjelmassa 4.3 ruudukko oli asetttu suorakaiteen sivujen suuntaisesti. Tehtävää voi muokata kiertämällä suorakaidetta. Muuttuuko tehdyt havainnot?

Tarkastellaan seuraavaksi ympyrän sisään rajaaman alueen (eli ympyräkiekon) pinta-alaa.

Määritelmä 4.4. *Ympyräkiekko* on tasojoukko, jonka pisteet ovat yhtä etäällä tai lähempänä annetua pistettä.

Huomaa, että ympyräkiekon reuna on ympyrä. Aloitetaan ympyräkiekon tutkiminen tekemällä alustavia havaintoja.

Ohjelmointitehtävä 4.5. Lataa valmis ohjelma `ruudukko.sb2` ja aja se. Sinun pitäisi nähdä ruudukko ja ympyräkiekko Y. Laske (ruudusta katsoen) kuinka monta ruutua eivät ole kokonaan Y:n ulkopuolella ja määritä Y:n a-pinta-ala. Toista tämä muutamalla muuttujan a eri arvoalla. Miten a-pinta-alat käyttäytyvät, kun a pienenee?

Vihjeet ja neuvot (Ohjelma 4.5): Tee taulukko, johon merkitään

(1) käytetyn neliön sivu b,
(2) pinta-ala b^2,
(3) suorakaiteen b-pinta-ala ja
(4) suorakaiteen a-pinta-ala.

Kasvata ja pienennä lukua b esim. $b = 2a$ ja $a/2$. Miten a-pinta-ala muuttuu?

Vaikuttaako ruudukon kiertäminen tulokseen kuten suorakaiteen tapauksessa?

Tarkastele, miten edellisessä ohjelmassa ruudukko oli rakennettu. Ihmisen on helppo havaita geometrisia säännönmukaisuuksia globaalilla tasolla, sillä ihmisen näköaisti on erikoistunut tähän. Sen sijaan tietokone toimii taas kerran paremmin, kun sillä on käsiteltävänään suuri määrä yksinkertaisia objekteja, tässä tapauksessa ruutuja. Tämä on eräs huomioitava seikka seuraavaa ohjelmointihaastetta ratkottaessa. Kuten aiemmin, ohjelman suunnitteleminen kannattaa jakaa pieniin osiin. Opettaja antaa tarvittaessa vihjeitä, minkälaisilla periaatteilla osat voisi muodostaa, sen jälkeen kun olette itse niitä jonkin aikaa pohtineet.

Lasketaan seuraavaksi ympyräkiekon pinta-ala.

Ohjelmointitehtävä 4.6. Kirjoita ohjelma, joka laskee Määritelmän 4.1 mukaisesti ympyräkiekon pinta-alan.

Yleistä ja arvioi:

- Kuinka tarkka on ohjelman laskema arvo?
- Antaako ohjelma sekä ylä- että alarajan arvolle?
- Toimiiko ohjelma mielivaltaiselle keskuspisteelle ja säteelle?
- Vertaa erilaisten ympyräkiekkojen säteitä ja pinta-aloja. Mitä havaitset?
- Miten lähestyin ongelmaa, mitä vaikeuksia kohtasin, ja miten ne hoidin?

Vihjeet ja neuvot (Ohjelma 4.6):

(1) Piirtäkää tilanteesta kuva paperille. Miten ruudukko asetetaan suhteessa ympyrään?
(2) Mitä parametreja ohjelmalle tulee antaa?
(3) Kun ohjelma on valmis, niin taulukoikaa pinta-aloja erilaisille ympyräkiekoille.

Tässä haastavassa tehtävässä voi antaa ainakin seuraavat vihjeet:

(1) Ratkaise ensin osaongelma: yksi kiekko ja yksi neliö, leikkaako vai ei?
(2) Ratkaise ensin helpompi ongelma: milloin neliön keskipiste on kiekon sisällä?
(3) Ratkaise ensin se ongelma jonka osaat: neliön ylänurkka (jos ryhmä on keksinyt tämän tapauksen, mutta empii lähteä toteuttamaan sitä).
(4) Mitä tietoja tarvitaan? 100 neliötä vai 10*10 ruudukko: ero tiedon määrässä.

Ohjelmista 4.5 ja 4.6 huomataan, että a-pinta-ala antaa aina ylärajan tasoalueen pinta-alalle. Miten a-pinta-alan määritelmää tulisi muuttaa, että sen avulla saisi laskettua alarajan tasoalueen pinta-alalle?

Pohdintatehtävä 4.7. Muokkaa a-pinta-alan määritelmää (Määritelmä 4.1) sellaiseksi, että se antaa tasokuvion pinta-alalle alarajan.

Vihjeet ja neuvot (Pohdinta 4.7):

(1) Piirtäkää renkaasta kuva paperille.
(2) Mitä tarkoittaa, että "a on pieni verratuuna lukuun r"?

Kuten edellisessä luvussa, empiirinen aineisto tukee otaksumaa, että ympyräkiekon pinta-ala on suoraan verrannollinen säteen neliöön. Seuraavaksi tehtävä on osoittaa tämä, määritelmiin perustuen.

Pohdintatehtävä 4.8. Osoita, että ympyräkiekon, jonka säde on on $r > 0$, pinta-ala on cr^2, missä $c > 0$ on vakio.

Vihjeet ja neuvot (Pohdinta 4.8):

(1) Piirtäkää tilanteesta Määritelmien 4.1 ja 4.4 mukainen kuva paperille.
(2) Miten ympyräkiekon a-pinta-alaa voi arvioida?
(3) Miten määritelmän avulla voi arvioida ympyräkiekon pinta-alaa?

5. π:n paluu

Edellisessä kappaleessa on osoitettu, että ympyräkiekon pinta-ala on suoraan verrannollinen säteen neliöön. Tässä kappaleessa osoitetaan, että verrannollisuuden vakio on sama luku π, joka esiintyy ympyrän pituuden yhteydessä.

Määritelmä 5.1. *Rengas* on tasojoukko, jonka pisteiden etäisyys annetusta pisteestä ovat kahden annetun positiivisen luvun välilä. Lukujen erotus on renkaan *paksuus*.

Tarkastellaan renkaan pinta-alaa.

Ohjelmointitehtävä 5.2. Kirjoita ohjelma, joka laskee Määritelmän 4.1 mukaisesti renkaan pinta-alan.

Yleistä ja arvioi:

- Kuinka tarkka on ohjelman laskema arvo?
- Antaako ohjelma sekä ylä- että alarajan arvolle?
- Toimiiko ohjelma mielivaltaiselle keskuspisteelle ja säteelle?
- Vertaa erilaisten renkaiden säteitä ja pinta-aloja. Mitä havaitset?
- Miten lähestyin ongelmaa, mitä vaikeuksia kohtasin, ja miten ne hoidin?

Vihjeet ja neuvot (Ohjelma 5.2):

(1) Piirtäkää renkaasta ja a-pinta-alan laskemisesta hahmotelma paperille.
(2) Voiko ohjelmassa hyödyntää ympyräkiekon pinta-alan laskevaa ohjelmaa (Ohjelm 4.6)?
(3) Mitkä muuttujat ohjelmalle tluee antaa?
(4) Taulukoi ohjelman laskemia pinta-aloja eri säteillä ja keskipisteillä.

Pohdintatehtävä 5.3. Osoita, että rengas, jonka säde on on $r > 0$ ja paksuus b, pinta-ala on noin πrb kun b on pieni verrattuna lukuun r.

Vihjeet ja neuvot (Pohdinta 5.3):

(1) Piirtäkää renkaasta kuva paperille.
(2) Mitä tarkoittaa, että "b on pieni verratuuna lukuun r"?

Ympyräkiekko voidaan ajatella myös koostuvan äärellisestä määrästä renkaita. Siten kiekon pinta-ala voidaan laskea renkaiden pinta-alojen avulla.

Pohdintatehtävä 5.4. Osoita, että ympyräkiekko, jonka säde on on $r > 0$, pinta-ala on πr^2, missä $\pi > 0$ on määritelty yllä.

Vihjeet ja neuvot (Pohdinta 5.4):

(1) Piirtäkää tilanteesta hahmotelma paperille.
(2) Miten renkaiden avulla voi "täyttää"ympyräkiekon?
(3) Voiko Pohdintaa 5.3 hyödyntää?

Ympyräkiekon pinta-alaa voi lähestyä myös seuraavan tehtävän avulla:

Ohjelmointitehtävä 5.5. Tee neliö $2r$ ja sen sisään ympyrä säde r. Sitten arpomalla merkitään pisteitä neliöön. Lopuksi lasketaan ympyrän sisällä olevien pisteiden suhteellinen osuus. Miten tämä liittyy lukuun π?

Vihjeet ja neuvot (Ohjelma 5.5):

(1) Piirtäkää tilanteesta hahmotelma paperille.
(2) Valitse ohjelmaa varten sopiva r (esim. 100)
(3) Piirrä pisteet kuvaan eri väreillä ympyrän sisä- ja ulkopuolelle

6. Rekursio

Rekursiivinen määritelmä esiintyy matematiikassa säännöllisesti. Tavoitteemme on tarkastella fraktaaleita, joista yksinkertaisimmat määritellään rekursion avulla. Ensimmäinen rekursioon liittyvä esimerkki on kertoman laskeminen positiiviselle kokonaisluvulle. Vaikka käsite on tuttu, niin rekursiivisen ohelman tekeminen voi olla haastavaa. Tavoitteena on

- ymmärtää miten rekursio toimii
- milloin rekursio lopetetaan (jos kyseessä on äärellinen määrä iteratioita) ?
- miten rekursiota voi hyödyntää (ongelman ratkaisussa ja ohjelmoinnissa) ?

Rekursiossa ohjelma kutsuu itseään uudestaan yksinkertaisemmalla syötteellä. Esimerkiksi rekursiivinen lukujonon järjestäminen (Tapa 1 sivulla 3). Rekursiivinen ohjelma on usein helpompi toteuttaa ja se tukee rinnakkaislaskentaa.

Tarkastellaan postiivisen kokonaisluvun n kertoman $n! = 1 \cdot 2 \cdot \ldots \cdot n$ laskemista. Silmukan avulla toteutettu ohjelma voisi olla toteutettu seuraavasti

kertoma1(n):
  alusta tulos arvolla 1
  toista luvuilla $k = 1, 2, \ldots, n$ lasku tulos=tulos $\cdot k$
  palauta tulos

Rekusrsiivisessa ohjelmassa kutsutaan itse ohjelmaa uudelleen

kertoma2(n):
  jos n on 1, niin palauta 1
  jos n on suurempi kuin 1, niin palauta $n\cdot$ kertoma2($n-1$)

Ohjelmointitehtävä 6.1. Kirjoita silmukan avulla ohjelma, joka laskee positiivisen kokonaisluvun kertoman.

Kirjoita avulla ohjelma, joka laskee rekursiivisesti positiivisen kokonaisluvun kertoman.

Yleistä ja arvioi:

- Mitä ero on ohjelmilla on?
- Mitkä ovat kummankin ohjelman hyvät ja huonot puolet?
- Miten ohjelmat toimivat, jos syöte ei ole kokonaisluku?
- Miten ohjelmat toimivat, jos syöte on negatiivinen?

Ohjelman 6.1 tekemisessä oppilaille voi olla hyödyllistä laskea paperilla konkreettinen esimerkki vaikka 7!.

Vihjeet ja neuvot (Ohjelma 6.1): Jos ryhmällä on vaikeuksia hahmottaa rekursiivista ratkaisua, voit käyttää seuraavia neuvoja:

(1) Laske kertomat 1!, 2!, 3!, 4! ja 5!.

(2) Miten luvun kertoma n saadaan palautettua luvun $n-1$ kertomaan?
(3) Mitä ohjelma palauttaa kun $n = 1$?

Vastauksia kysymksiin:

- Toinen suorittaa laskun yhdellä funktiokutsulla ja toinen useammalla.
- For–silmukan hyvä puoli on sen selkeys. On helppo ymmärtää, että ohjelma toimii oikein. Silmukkaratkaisun huono puoli on sen hitaus.
  Rekursiivisen ohjelman hyvä puoli on sen lyhyt toteutus. Lisäksi suurella luvulla n laskennan voisi jakaa useammalle prosessorille ja näin nopeuttaa laskemista. Rekursiivisen ohjelman oikeellisuus on vaikeampi tarkistaa ja ikuisen silmukan mahdollisuus on suurempi.
- Jos n ei ole kokonaisluku, niin molemmat ohjelmat toimivat samalla tavalla väärin.
- Jos n on negatiivinen, niin molemmat ohjelmat toimivat väärin ja rekursiivinen ohjelma joutuu mahdollisesti ikuiseen silmukkaan.

Seuraavaksi tarkastellaan Cantorin joukkoa. Kyseessä on yksikköjanan osajoukko, joka saadaan muodostettua seuraavasti:

Määritelmä 6.2. Määritellään joukot C_n.
(1) joukko C_1: jana pisteestä $(0,0)$ pisteeseen $(1,0)$
(2) joukko C_2: poista edellisen kohdan janasta keskimmäinen kolmannes, jolloin jäljelle jää janat pisteestä $(0,0)$ pisteeseen $(1/3,0)$ ja pisteestä $(2/3,0)$ pisteeseen $(1,0)$
(3) joukko C_n: poista edellisen kohdan janoista keskimmäinen kolmannes

Cantorin joukko C saadaan raja-arvona $C = \lim_{n\to\infty} C_n$.

Huomataan, että Cantorin joukon määritelmä on rekursiivinen. Joukko C_1 muodostuu yhdestä janasta. Sen jälkeen seuraava joukko C_n saadaan edellisestä korvaamalla jokainen jana kahdella lyhyemmällä janalla.

Kuva 1. Joukot C_1, C_2, C_3 ja C_4. Jokaisessa joukossa pisteet $(0,0)$ ja $(1,0)$ on merkitty.

Cantorin joukon C määritelmässä ehto (2) on esitetty ymmärtämisen helpottamiseksi. Se on ylimääräinen, koska se sisältyy ehtoon (3). Ideana rekursiivisessa ohjelmassa on kutsua ohjelmaa uudestaan kahdesti: kerran vasemman puoleiselle ja kerran oikean puoleiselle janalle. Kun haluttu iteraatiosyvyys n on saavutettu piirretään yksi jana. Iteraatio syvyydelle kannattaa luoda muuttuja, jota kasvatetaan jokaisella iteratiivisella kutsulla.

Ohjelmointitehtävä 6.3. Tee ohjelma, joka piirtää edellä esitetyn määritelmän mukaan joukon C_n.

Vihjeet ja neuvot (Ohjelma 6.3): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Piirtäkää hahmotelma joukoista C_1, C_2 ja C_3 paperille.

(2) Mikä määritelmän kohdista on lopetusehto? Ja minkä avulla muodostetaan iteraatio?
(3) Mitä ohjelman tulee tehdään, kun lopetusehto täyttyy?
(4) Kuinka monta kertaa ohjelmaa kutsutaan uudestaan yhdellä iteraatiolla?

Funktiolle kannattaa antaa muuttujaksi janan päätepisteet ja kokonaisluku, joka vastaa joukon C_k arvoa k.

Ohjelma 6.3 piirtää joukon C_n. Indeksin n kasvaessa piirretty C_n muistuttaa enemmän Cantorin joukkoa C. Joukkoa C ei voi piirtää, sillä se koostuu yksittäisistä pisteistä ja jokainen C_n koostuu lyhyistä janoista.

Joukko C_1 on jana, jonka pituus on 1, ja joukkojen C_n janojen yhteispituus on enintään 1. Seuraavaksi tarkastelemme janojen yhteispituutta.

Pohdintatehtävä 6.4. Kuinka monta janaa on joukossa C_n? Mikä on joukon C_n janojen yhteispituus?

Vihjeet ja neuvot (Pohdinta 6.4):

(1) Piirtäkää hahmotelma joukoista C_1, C_2, C_3 ja C_4 paperille.
(2) Laskekaa janojen lukumäärä piirretyissä joukoissa. Kuinka monta janaa on joukossa C_n?
(3) Minkä pituisia janoja on joukossa C_2, C_3 ja C_4? Kuinka pitkiä ne ovat?
(4) Hyödynnä janan pituuksia ja janojen lukumäärää janojen yhteispituuden määrittämiseen.
(5) Voitko hyödyntää ohjelmointitehtävän ratkaisua (mahdollisesti hieman muokattuna) ratkaisusi tarkastamiseen?

Muistuta tarvittaessa ryhmiä geometrisen sarjan summakaavasta.

7. Fraktaali

Fraktaalin määritteleminen matemaattisesti on melko monimutkaista ja emme tee sitä tässä materiaalissa. Tarkoitamme fraktaalilla joukkoa, jonka dimensio (katso seuraava luku) ei ole kokonaisluku. Esimerkiksi Cantorin joukko on tässä mielessä fraktaali. Dimensio vaatimuksesta seuraa, että fraktaalilla on tietynlainen rakenne riippumatta siitä kuinka paljon suurennamme sitä. Seuraavaksi tarkastelemme kahta eri tason fraktaalia. Usein fraktaali määritellään kuten Cantorin joukko, raja-arvona iteratiivisesti muodostetuista joukoista.

Ensimmäinen tason fraktaali on Sierpińskin kolmio. Se muodostetaan tasakylkisten kolmioiden avulla.

Määritelmä 7.1. Määritellään joukot S_n.
(1) joukko S_1: tasakylkinen kolmio
(2) joukko S_{n+1}: muunna jokainen kuvion S_n tasakylkinen kolmio kuvan 2 mukaisesti kolmeksi uudeksi pienemmäksi taskylkiseksi kolmioksi

Sierpińskin kolmio S saadaan raja-arvona $S = \lim_{n\to\infty} S_n$.

KUVA 2. Joukot S_1, S_2, S_3, S_4 ja S_5.

Sierpińskin kolmio on määritelty yksinkertaisesti tasakylkisten kolmioiden avulla. Ennen joukkojen S_n piirtämistä anna oppilaiden tutkia määritelmää määrittämällä kolmioiden lukumääriä ja sijainteja. Niiden tutkiminen auttaa myös myöhemmin dimension ymmärtämisessä.

Huomataan, että joukkojen S_n määritelmä on rekursiivinen.

Pohdintatehtävä 7.2. Kuinka monta kolmiota on joukossa S_n?

Vihjeet ja neuvot (Pohdinta 7.2): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Piirtäkää hahmotelma joukoista S_1, S_2 ja S_3 paperille.
(2) Mitä tapahtuu kolmiolle S_1, kun muodostetaan S_2?
(3) Mitä tapahtuu yksittäiselle kolmiolle joukossa S_2, kun muodostetaan S_3?
(4) Kuinka monta kolmiota yhdestä kolmiosta tulee?

Ennen Ohjelman 7.3 tekemistä kannattaa tehdä ohjelma, joka piirtää tietyn kokoisen tasakylkisen kolmion tiettyyn paikkaan. Tämä voidaan antaa oppilaille valmiina tai se voi olla ohjelma, jonka oppilaat ovat tehneet aiemmin.

Ohjelmointitehtävä 7.3. Tee iteratiivinen ohjelma, joka piirtää edellä esitetyn määritelmän mukaan joukon S_n.

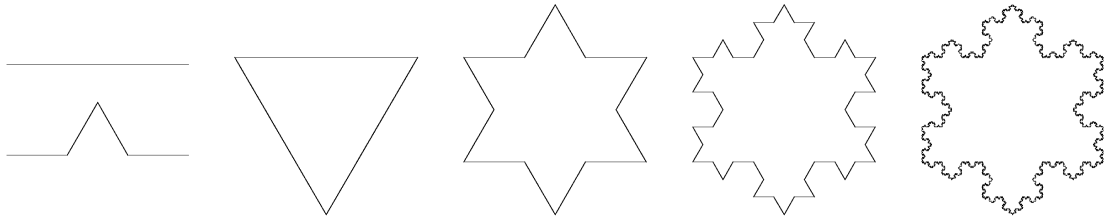
Vihjeet ja neuvot (Ohjelma 7.3): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Piirtäkää hahmotelma joukoista S_1, S_2 ja S_3 paperille.
(2) Mitä tapahtuu yhdelle kolmiolle, kun siirrytään joukosta S_1 joukkoon S_2? Kuinka monta kolmiota yhdestä tulee? Missä uudet kolmiot sijaitsevat? Minkä kokoisia uudet kolmiot ovat?
(3) Milloin rekursio lopetetaan? Mitä silloin piirretään?

Seuraavaksi esittelemme fraktaalin, jota kutsutaan nimellä Kochin lumihiutale.

Määritelmä 7.4. Määritellään joukot K_n.
(1) joukko K_1: tasakylkinen kolmio, joka muodostuu kolmesta janasta
(2) joukko K_{n+1}: tee jokaiselle kuvion K_n janalle kuvan 3 mukainen kulmamuunnos

Kochin lumihiutale K saadaan raja-arvona $K = \lim_{n\to\infty} K_n$.



KUVA 3. Kulmamuunnos, joukot K_1, K_2, K_3 ja K_4.

Kochin lumihiutaleessa keskeisenä asiana on kolmiomuunnos. Ennen Ohjelmaa 7.5 on hyvä tehdä ohjelma, joka piirtää annetulle janalle kolmiomuunnoksen. Tämä ohjelma voidaan tatrvittessa antaa oppilaille valmiina.

Ohjelmointitehtävä 7.5. Tee iteratiivinen ohjelma, joka piirtää edellä esitetyn määritelmän mukaan joukon K_n.

Vihjeet ja neuvot (Ohjelma 7.5): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Piirtäkää hahmotelma joukoista K_1, K_2 ja K_3 paperille.
(2) Mikä on rekursion lopetusehto? Mitä silloin piirretään?

Tarkastellaan lopuksi joukkojen K_n janojen lukumäärää. Kuten Cantorin joukon tapauksessa, tämä tarkastelu auttaa määritelmän ymmärtämistä ja helpottaa myöhemmin dimension laskemista. Huomataan, että kulmamuunnoksessa kolmion poistaminen (ja uloimien janojen säilyttäminen) tuottaa Cantorin joukon.

Pohdintatehtävä 7.6. Kuinka monta janaa on joukossa K_n? Mitä yhteistä on Kochin lumihiutaleella ja Cantorin joukolla?

Vihjeet ja neuvot (Pohdinta 7.6): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Kuinka monta janaa yhdestä janasta tulee kulmamuunnoksessa?
(2) Piirtäkää hahmotelma joukoista K_1, K_2 ja K_3 paperille. Kuinka monta janaa joukoissa on?
(3) Kochin lumihiutaleessa käytettiin kulmamuunnosta. Minkälainen muunnos on Cantorin joukon konstruktiossa?

Kochin lumihiutaleesta voidaan tehdä helposti muunnoksia muuttamalla kolmiomuunnosta. Muunnoksessa keskimmäinen kolmannes voidaan korvata esim. neliöllä. Kolmimuunnoksen kolmion kokoa voidaan myös muuttaa: kolmanneksen sijaan valitaan neljännes. Kolmiomuunnosta muokatessa tulee varmistua, ettei fraktaalin käyrä leikkaa itseään.

8. Dimensio

Tarkastelemme dimensiota. Esimerkiksi pisteen dimensio on 0, suoran tai janan dimensio on 1 ja tason tai ympyräkiekon dimensio on 2. Edellä esitetyt fraktaalit edustavat yksinkertaisia

konstruktioita, joiden dimensio ei ole kokonaisluku. Cantorin joukon dimensio on "enemmän" kuin pisteen, mutta "vähemmän" kuin suoran. Sierpińskin kolmion ja Kochin lumihiutaleen dimensio on "enemmän" kuin suoran, mutta "vähemmän" kuin tason. Dimensio voidaan määritellä usealla eri tavalla, joista eräs on Minkowskin dimensio. Se perustuu eri kuvion peittävien r-säteisten ympyräkiekkojen lukumäärään, kun r lähestyy nollaa.

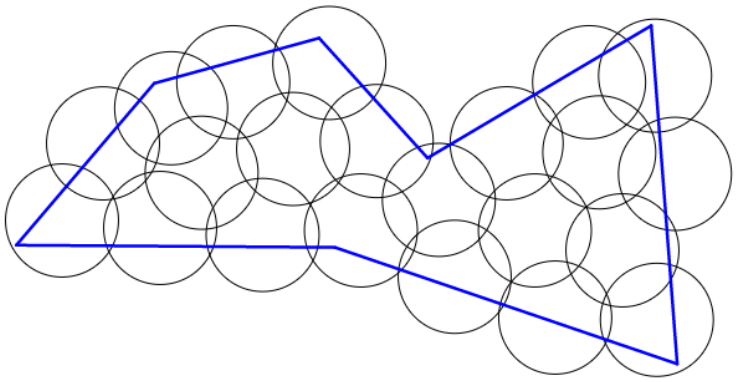
Määritelmä 8.1. Olkoon E tason joukko. Sanomme, että joukko E voidaan peittää joukoilla $A_1, A_2, \ldots, A_m$, jos

$$E \subset \bigcup_{i=1}^{k} A_i.$$

Merkitään luvulla $N_r(E)$ pienintä määrä r-säteisiä ympyräkiekkoja, joiden avulla voidaan peittää joukko E.

Joukon E dimensio on

$$\dim(E) = \lim_{r \to 0} \frac{\log N_r(E)}{-\log r}.$$



Kuva 4. Sininen monikulmio on peitetty saman säteisillä ympyröillä.

Tarkastellaan Määritelmässä 6.2 esitetyn joukon C_n peittämistä r-säteisillä ympyräkiekoilla.

Joukossa C_n on 2^{n-1} kappaletta 3^{1-n} pituista erillistä janaa (tämä laskettiin Pohdinnassa 6.4). Merkitään $r_i = 3^{1-i}$. Joukko C_1 voidaan peittää 3^{i-1} kiekolla, joiden säde on r_i. Joukko C_2 voidaan peittää 2, r_i-säteisellä ympyräkiekolla, jos $i \leq 2$, ja $2^{n-1} \cdot 3^{i-2}$, jos $i > 2$. Joukko C_k voidaan peittää 2^{n-1}, r_i-säteisellä ympyräkiekolla, jos $i \leq n$, ja $2^{n-1} \cdot 3^{i-k}$, jos $i > n$.

Ohjelmointitehtävä 8.2. Tee ohjelma, joka laskee kuinka monella r-säteisellä ympyräkiekolla voidaan peittää C_n.

Vihjeet ja neuvot (Ohjelma 8.2): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Joukot C_n koostuvat janoista. Miten jana voidaan peittää ympyräkiekoilla?
(2) Hahmottele kuva joukoista C_n ja niiden peittävistä ympyräkiekoista arvoilla $n = 1, 2, 3$.
(3) Jos peittäminen tuottaa ongelmia, niin kertaa Luku 4.
(4) Laske ympyräkiekkojen määrä ensin paperilla.
(5) Miten säteen r arvo vaikuttaa joukon C_n peittämiseen?

Pohdintatehtävä 8.3. Osoita, että Cantorin joukon dimensio on $\log 2/\log 3$.

Pohdinnassa 8.3 lasketaan Cantorin joukon dimensio. Sen laskeminen perustuu Ohjelman 8.2 yhteydessä laskettuun arvoon $N(C_n)$.

Pohdintatehtävä 8.4. Voiko määritelmässä käyttää ympyräkiekkojen sijaan neliöitä tai tasakylkisiä kolmiota?

Tarkastellaan tilannetta $k = i$. Silloin

$$\dim(C_n) = \lim_{i\to\infty} \frac{\log 2^{i-1}}{-\log 3^{1-i}} = \lim_{i\to\infty} \frac{i-1}{1-i}\frac{\log 2}{-\log 3} = \lim_{i\to\infty} \frac{\log 2}{\log 3} = \frac{\log 2}{\log 3}.$$

Vihjeet ja neuvot (Pohdinta 8.4): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Jos ympyräkiekon koko määriteltiin säteen r avulla, niin miten neliön ja kolmion koko kannattaa määritellä?
(2) Miten yksittäinen ympyräkiekko avulla voidaan peittää yhdellä neliöllä tai kolmiolla? Hahmottele tilannetta kuvan avulla.
(3) Miten yksittäisestä tapauksesta päästään koko joukon peittämseen? Hahmottele tilannetta kuvan avulla.

Koska Cantorin joukko on suoran osajoukko, niin sen dimension on enintään 1. Pohdinnan 8.4 mukaan dimensio on $\log 2/\log 3 \approx 0{,}63$. Sierpińskin kolmio ja Kochin lumihiutale ovat tason osajoukkoja, joten niiden dimensio on enintään 2.

Lasketaan seuraavaksi Sierpińskin kolmion dimensio.

Ohjelmointitehtävä 8.5. Tee ohjelma, joka laskee kuinka monella r-säteisellä ympyräkiekolla voidaan peittää S_n.

Vihjeet ja neuvot (Ohjelma 8.5): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Joukot S_n koostuvat tasakylkisistä kolmioista. Miten tasakylkinen kolmio voidaan peittää ympyräkiekoilla?
(2) Piirrä kuva joukoista S_n, $n = 1, 2, 3$, ja niiden peittämistä ympyräkiekoista.
(3) Mikä säteen r arvo vaikuttaa joukon S_n peittämiseen?

Pohdinnassa 8.6 lasketaan Sierpinksin kolmion dimensio. Sen laskeminen perustuu Ohjelman 8.5 yhteydessä laskettuun arvoon $N(S_n)$.

Pohdintatehtävä 8.6. Osoita, että Sierpińskin kolmion dimensio on $\log 3/\log 2$.

Vihjeet ja neuvot (Pohdinta 8.6):

(1) Hahmottele tilanteesta kuva paperille.
(2) Miten $N(S_n)$ lasketaan? (Ohjelma 8.5)
(3) Miten dimensio lasketaan? (Määritelmä 8.1)

Lopuksi lasketaan Kochin lumihiutaleen dimensio.

Ohjelmointitehtävä 8.7. Tee ohjelma, joka laskee kuinka monella r-säteisellä ympyräkiekolla voidaan peittää K_n.

Vihjeet ja neuvot (Ohjelma 8.7): Jos ryhmällä on vaikeuksia hahmottaa tilanne, voit käyttää seuraavia neuvoja:

(1) Joukot K_n koostuvat janoista. Miten jana voidaan peittää ympyräkiekoilla?
(2) Piirrä kuva joukoista K_n, $n = 1, 2, 3$, ja niiden peittämistä ympyräkiekoista.
(3) Mikä säteen r arvo vaikuttaa joukon K_n peittämiseen?

Pohdinnassa 8.8 lasketaan Kochin lumihiutaleen dimensio. Sen laskeminen perustuu Ohjelman 8.7 yhteydessä laskettuun arvoon $N(K_n)$.

Pohdintatehtävä 8.8. Osoita, että Kochin lumihiutaleen dimensio on $\log 4/\log 3$.

Vihjeet ja neuvot (Pohdinta 8.8):

(1) Hahmottele tilanteesta kuva paperille.
(2) Miten $N(S_n)$ lasketaan? (Ohjelma 8.7)
(3) Miten dimensio lasketaan? (Määritelmä 8.1)