Asiaa harjoituksista ja tavoitteista eri luokka-asteille

Varhaiskasvatus ja alakoulu

Perinteisesti algoritmisen ajattelun harjoittelu voidaan aloittaa ilman tietokonetta käyttämällä käskyjen vastaanottajana toista lasta, opettajaa tai robottia. Tällaisten harjoitusten avulla syntyy käsitys siitä, että tietokoneen tuntemien käskyjen määrä on pieni, käskyt vaihtuvat ohjelmointikielen mukaan ja käskyt ovat varsin rajattuja merkityksiltään.

Tämän jälkeen päästään helposti leikillisin menetelmin erilaisten algoritmien harjoittelemiseen, kun jokaiselle lapselle annetaan oma toimintalogiikkansa. Useimmiten jokaisella ”ohjelman osalla” on sama määräys. Koska lapset yleensä joutuvat dataelementin rooliin, voi kuitenkin syntyä väärinkäsityksiä ohjelman käskyjen ja datan välillä.

CS Unplugged -sivustolta löytyy paljon harjoituksia ja ohjeita erilaisten leikkien toteuttamiseen. Valitettavasti nämä materiaalit ovat englanninkielisiä.

Nuorimmille algoritmisen ajattelun opettelijoille on tarjolla valikoima robottimaisia leluja (Piagetin mukaan esine, jonka kautta voi ajatella ja johon voi samaistua näkökulman luomiseksi). Tällainen on esimerkiksi Beebot, jota voi ohjata komentosarjoilla painamalla lelun komentonappeja: liiku eteen, liiku taakse, käännös vasemmalle tai käännös oikealle. Harjoittelun voi aloittaa yksittäisistä komennoista eli painalluksista ja lelun reaktioiden seuraamisesta. Myöhemmin voi antaa samalla kertaa pidemmän jonon komentoja ja seurata, päätyykö lelu aiottuun pisteeseen. Yksi lapsien kohtaamista haasteista on tarkastella komentoja lelun näkökulmasta, ei omastaan. Lelun kasvopuoli päätyy helposti osoittamaan lapsen kasvoja, mikä pakottaakin lapsen oivaltamaan ohjelmoinnin kannalta olennaisen hahmotustavan.

Jos tällaisia leluja ei ole käytettävissä, lapset nauttivat myös opettaja robottina -leikistä. Komennot annetaan sanallisesti ja opettajan on pidettävä huoli siitä, ettei korjaa annettua komentosarjaa ns. loogiseksi, vaan nimenomaan tuo esiin mahdottomat tilanteet, kuten käskysarjassa molempien jalkojen nostaminen ilmaan laskematta toista välillä maahan.

Tekniikan tarinamatto on hyvä paikka rauhalliselle keskustelulle ja arjessa kohdattuun teknologiaan tarkemmin perehtymiselle.

Myöhempää tietokoneella tapahtuvaa ohjelmointia ajatellen hyvää harjoittelua ovat muun muassa riimittely ja toistuvien hahmojen tai hahmosarjojen tunnistaminen sekä tekstinä, piirroksina että numerosarjoina. Esimerkiksi Britanniassa lapset riimittelevät paljon ja piirtävät toistuvan yksinkertaisen kuvion muodostamia kuvia. Nämä harjoitukset auttavat myöhemmin esimerkiksi löytämään tekstipohjaisen ohjelman toisistaan kaukana olevia alku- ja loppusulkuja toimintosarjojen rajoilta ja hahmottamaan eri parametriarvoilla kutsuttujen aliohjelmien yhteistoimintaa. Samoin esineiden tai asioiden luokittelu kategorioihin auttaa myöhemmin ongelmanratkaisussa, kun on jo totuttu käsitteiden täsmällisiin määrittelyihin kategoriatehtävien kautta.

Olennaisin tavoite alkuopetuksessa on ohjelmoinnista ja tietojenkäsittelystä innostuminen. Arvioinnissa korostuukin siten harjoitteisiin ja leikkeihin halukkaasti osallistuminen, uteliaisuus ja kyseleminen sekä omien havaintojen jakaminen.

Alakoulujen osalta ohjelmoinnin opetus keskittyy ennen kaikkea siihen, että TVT-taitoja kerrytetään ja tietokoneita lakataan aristelemasta arjessa. Niin oppilaat kuin opettajatkin.

Nuorimpien oppilaiden kohdalla ohjelmointiin liittyvää ajattelua voi harjaannuttaa antamalla päättelytehtäviä, joissa sallittujen toimenpiteiden joukko on rajoitettu. Hauska leikki on myös robottina toimivan luokkatoverin – tai jopa opettajan – ohjeistaminen tietyn toimenpidesarjan lävitse. Alkuun seuraava ohje annetaan robotin liikuttua, myöhemmin harjaannutaan pohtimaan koko toimintosarja kerralla ennen ohjeiden antamista robotille.

Algoritminen ajattelu kasvaa jo näin pienin toimin. Kun havaitaan vielä, että erilaiset tavat tuottaa haluttu lopputulos ovat tehokkuudeltaan erilaisia,

päästään kiinni algoritmisen ajattelun yhteen keskeiseen teemaan: onko tapa tehokas? Tarvitaanko tarpeettoman paljon viestintää tai muistitilaa?

Esimerkki algoritmi- ja protokollaleikistä

- Kukin lapsi edustaa käskyä tai muuttujaa.
- Käskyjonon ensimmäiselle annetaan syötetieto ja hän noudattaa hallitsemaansa käskyä.
- Käsky voi sisältää pyynnön muuttuja-lapselta tiedon kysymisestä tai määräyksen tiedon kirjoittamisesta muistiin.
- Kun käskyjono on käyty läpi, viimeinen käsky tuottaa jotain hauskaa – esimerkiksi muuttuja-lapset joutuvat luettelemaan hassun sanan kirjaimet vuorollaan.
- Testaus-variaationa tässä leikissä takerrutaan epäselviin käskyihin tai virheisiin: lapset huomaavat, että käsky on väärä, mutta eivät saa leikin sääntöjen mukaan korjata sitä, vaan “kaatavat ohjelman” sovitulla tavalla.

Alakoululaisille OPS suosittaa graafiseen ohjelmointiympäristöön tutustumista ja siinä ohjelmien tekemistä sekä positiivisten kokemusten hankkimista. Esimerkiksi Linkki-resurssikeskuksen kerho- ja leiritoiminta kattaa tällaisia harjoitteita ja opettajan ohjeet näiden Scratch-ohjelmien teettämiseen ovat vapaasti saatavissa. Pulmaario-ohjaajan oppaasta on myös apua.

Scratch-ohjelmointiympäristö on yhdysvaltalaisen MIT-korkeakoulun tuotos ja sekin vapaasti käytettävissä. Opettajat voivat hyvin käydä kokeilemassa ohjelmointia itsekseen, katsoa Linkin opettajille suunnattuja webinaareja tai käydä paikan päällä Linkki-koordinaattoreiden opissa. Kokeneita ohjaajia voi myös pyytää koululle pitämään ensimmäisiä tunteja, jolloin opettajakin voi tutustua ympäristöön samalla.

Scratch-yhteisössä tunnukset ovat pseudonyymejä, mikä tarkoittaa että aitoja nimiä ei tarvitse käyttää, vaan kaikilla on erillinen näkyvä nimimerkki. Tehtäviä voi toki palvelussa tehdä kirjautumattakin, mutta projekteja ei silloin voi jakaa toisten nähtäväksi eikä esittää keskusteluryhmissä kysymyksiä. Kaikki Scratch-järjestelmän keskustelupalstat ovat valvottuja. Periaatteena on

nostaa nuoria edistyneitä käyttäjiä palstavahdeiksi ja antaa heille valtaa esimerkiksi nostaa esiin omia suosikkipelejään jaettujen projektien joukosta. Alle 13-vuotiaan tunnuksen luonnissa tarvitaan vanhemman tai huoltajan sähköpostiosoite.

Muunlaista graafista ohjelmointia sekä tietokoneella tehtäviä harjoituksia löytyy muun muassa osoitteesta Code.org. Vain osa tehtävistä on käännetty suomeksi.

- https://code.org/student/elementary
- https://code.org/student/middle-high
- https://code.org/student/university

Edistyneempiä oppilaita voi ohjata tekstipohjaisiin ohjelmointikieliin. Mahdollisia vaihtoehtoja ovat esimerkiksi Tie Koodariksi -sivuston Python-oppimismateriaali, Linkin Python-oppimismateriaali sekä Helsingin yliopiston Ohjelmoinnin MOOC -kurssi. Jotkut ovat myös pitäneet CodeCombat-materiaalista (joka on kaupallinen tuote), jossa ohjelmointia voi harjoitella Pythonilla tai JavaScriptillä.

Kynnyskäsitteiden osalta esimerkiksi silmukka ja silmukkamuuttuja vaativat sekä abstraktia ajattelua että harjaantumista esimerkkien kautta. Opettajan on rohjettava puuttua logiikkavirheisiin. Graafisissa ohjelmointiympäristöissä on toisinaan tuki silmukoille toistokertojen konkreettisen lukumäärän mukaan, vaikka normaalimpaa on silmukkamuuttujan ja mutkikkaampien ehtolausekkeiden käyttö silmukan kiertämisen päättämiseksi. Perinteisiä erehdyksiä silmukan hallinnassa on kahdenlaisia:

- silmukan kierrosmäärä on valittu väärin (yleensä yksi liikaa tai liian vähän)
- silmukassa tehtävän työn ennakkovalmistelut tai silmukan jälkeen tehtävät asiat eksyvät silmukan sisään.

Teknisemmällä puolella tietotekniikka käsittelee muun muassa antureita, jotka mittaavat jatkuvasti jotakin suuretta – esimerkiksi lämpötilaa, valoisuutta, kosteutta tai kosketusta – ja antavat kuuntelevalle laitteelle aika ajoin mitatun lukuarvon. Antureista, johtimista ja pienistä servomoottoreista voidaan rakentaa erilaisia monitoimilaitteita, jotka käyvät paristoilla. Tämä laventaa

esimerkiksi teknisen työn tuntien valikoimaa. Tällaisten harjoitteiden laatimista on harjoiteltu yliopistollakin, joten koeteltuja tehtävämalleja on tältäkin alueelta saatavilla.

Ohjelmointiin ja tietojenkäsittelyyn liittyvää harrastustoimintaa on hyvä tuoda esiin monissa eri asiayhteyksissä.

Ohjelmointitaidon alkeiden lisäksi alakoululaisten oppimistavoitteita edistää ymmärrys verkossa olevien palveluiden toiminnasta, joista esimerkkeinä yksityisyyteen ja datan keräämiseen liittyvät asiat. Lähtökohtaisesti palveluntarjoajilla on pääsy heidän lähettämiinsä viesteihin sekä palveluihin lataamiinsa kuviin/tiedostoihin. Palveluntarjoajat myös tallentavat käyttäjistä monenlaista aktiivisuus- ja sijaintidataa. Oma palveluiden käyttö voi vaikuttaa negatiivisesti myös muiden ihmisten yksityisyyteen.

Tietoturvan osalta oppilaita on tarpeen ohjata käyttämään kriittisissä palveluissa (kuten rahaan, maineeseen tai opintoasioihin liittyvissä palveluissa) vahvoja salasanoja yhdessä kaksivaiheisen tunnistautumisen kanssa. Samaa salasanaa ei pidä käyttää useassa eri kriittisessä palvelussa. Salasanaholvisovellusten käyttö voi joissakin tapauksissa olla harkitsemisen arvoista.

Yksityisyydestä huolehtiminen edellyttää myös, että oppilaat ymmärtävät, ettei kenenkään kuvaa saa julkaista ilman lupaa ja kavereiden turvallisuudesta huolehditaan myös. Turvallisen verkon käytön kannalta oppilaiden kanssa puhutaan myös siitä, että tapaamisia tuntemattomien kanssa ei pidä sopia. Joitakin näkökulmia sosiaalisen median käytöstä voi löytää KenGuru-materiaalista. Suomen Kyberturvallisuuskeskukselta voi myös löytyä hyödynnettävää materiaalia. Heiltä voi esimerkiksi tilata ilmaiseksi Turvallisesti netissä -opasvihkosia. Samoin Mediataitokoululta voi löytää erilaisia materiaaleja.

Yläkoulu ja lukio

Vasta yläkoululaisille suositellaan tekstipohjaisen ohjelmointikielen käyttöä, mutta myös edistyneempiä alakoululaisia voi hyvin ohjata tekstipohjaisten ohjelmointikielten pariin. Algoritminen ajattelu yhdistyy tässä vaiheessa eri oppiaineista löytyviin laskentakaavoihin: mekaniikan lakeihin fysiikassa,

yhtälöryhmien ratkaisemiseen graafisesti matematiikassa tai vaikkapa sanakoekilvan toteuttamiseen kielten tunneille.

Tekstipohjaisen ohjelmointikielen opettelu vaatii enemmän harjaantumista kuin graafinen ohjelmointiympäristö. Käskyjen luonne ja täsmällinen ilmaisutapa täytyy oppia kirjoittamaan oikein, vaikka monet ohjelmoijaa tukevat tekstinkäsittelyohjelmat ja ohjelmointiympäristöt asiassa auttavatkin.

Jos opettajan oma ohjelmointitaito tai käytettävissä oleva valmisteluaika ei riitä, käytettävissä on MOOC-kursseja, joilla oppilas (tai opettaja itsekin) noudattaa kurssin materiaalin mukaisia ohjeita, suorittaa automaattisesti testattavia ohjelmointitehtäviä ja kerää kustakin vaiheesta suorituspisteitä. MOOC-kursseja voi hakea esimerkiksi Linkki-resurssikeskuksen sivuilta tai MOOC.fi-sivuston kautta suoraan. Mahdollisia vaihtoehtoja ovat Tie Koodariksi -sivuston Python-oppimismateriaali, Linkin Python-oppimismateriaali sekä Helsingin yliopiston Ohjelmoinnin MOOC -kurssi.

Toisen haasteen opiskeluprojekteissa muodostaa usein englanninkielinen toimintaympäristö, joten "tietokone-englannin" perusteiden hallitseminen on toivottavaa.

Oppiminen ohjelmakoodin rakenteesta voidaan jakaa neljään kehitysaskeleen:

- Konkreettinen komentosarja, joka toimii vain yksittäisessä tilanteessa eikä ole yleisen haasteen ratkaisu.
- Yleistetympi ratkaisu, joka sisältää käskysarjojen kopiointia uusien suorituskertojen toteuttamiseksi.
- Yleistetympi ratkaisu, jossa selkeät erot eri toimintosarjoihin haarautumisessa ja jossa käytetään ehtolausekkeita ja silmukkarakenteita.
- Aliohjelmien tai moduulien rakentaminen ja parametrien välittäminen.

Oppilaita on syytä rohkaista etenemään ratkaisuissaan viimeisintä tyyliä kohti. Ilman tietokonetta voidaan päästä tasolle 2 tai 3. Olennaista tasolla 2 on, etteivät datan ja ohjelman suorittimen roolit sekaannu. Vaiheessa 4 uusi oivallus on se, että sama toimintosarja voidaan tehdä eri muuttujille tilanteesta riippuen, jolloin saadaan uudelleenkäytettävä ohjelmamoduuli. Uudelleenkäytettävyys onkin ohjelmoijien työn eräs peruspyrkimys.

Kun oppilaat pääsevät ohjelmointiprojektiensa kanssa liikkeelle esimerkiksi Scratch-ympäristössä tai jossakin tekstuaalisessa ympäristössä, syntyviä tuotoksia voi arvioida muutaman kynnystaidon kautta. Kynnystaidolla tarkoitetaan toiminnallista taitoa, jonka oppiminen hyväksyttävällä ja riittävällä tavalla auttaa laadukkaan ohjelmointiprojektin toteuttamisessa tai oppimistehtävän suorittamisessa olennaisesti, mutta jonka puuttuminen estää projektista hyötymisen.

Ohjelmointiprojektin arvioinnissa voidaan kiinnittää huomiota seuraaviin kynnystaitoihin:

- toimintatapojen selostaminen,
- vaihtoehtojen vertailu,
- ratkaisurakenne sekä
- testaus, skaalautuvuus ja käytettävyys.

Oppilaille olennainen taito on oman ohjelman toimintatavan ja ratkaisurakenteen perusteleminen. Lisäksi on toivottavaa, että ohjelman erilaisista kehitysasteista osataan kertoa ja perustella miten uudet lisäykset parantavat toiminnallisuutta tai käytettävyyttä. Keskustelutaito on kognitiivisen ohjauksen edellytys, mutta ohjaa myös ammattimaiseen toimintatapaan: käyttämällä oikeita nimityksiä, jotka ryhmän muut jäsenet ymmärtävät täsmällisesti, keskustelusta saa vertaispalautetta. Tämä vertaispalaute voi toimia vahvempana kannustimena kuin opettajan palaute. Keskusteluissa oppilaat näkevät myös muiden tekemiä ratkaisuja, joista voi havaita uusia malleja tai oppia esimerkin kautta abstraktimpaa ajattelua.

Ratkaisujen yksityiskohdat ovat toisaalta varsin vapaita innovaatiokohteita, mutta ratkaisujen vertailua esimerkiksi suorituskyvyn suhteen on hyvä oppia tekemään. Skaalautuvuudella tarkoitetaan sitä, että ohjelman suoritusaika ei kasva kohtuuttomasti vaikka syötteiden tai objektien määrää kasvatetaan.

Testauksella tarkoitetaan systemaattista ohjelman kokeilua niin, että varmistutaan ettei ohjelmassa ole toimintalogiikkavirheitä. Testausmenetelmiä on useita: voidaan esimerkiksi suunnitella testeille syötteitä niin, että jokainen erilainen suorituspolku tulee suoritetuksi ainakin kerran. Toisaalta voidaan miettiä käyttäjän erilaisia reaktioita ohjelman

tarjoamiin mahdollisuuksiin. Käytettävyys tarkoittaa lähinnä luontevaa käyttötuntumaa: ohjelmia ei laadita vain omaan käyttöön, vaan muidenkin on voitava niitä käyttää ilman laajaa ohjeistusta.

Yläkoulussa ja lukiossa tavoitteena voi pitää projektityöskentelyä, jossa graafisilla tai tekstipohjaisilla välineillä ohjelmoiden tutustutaan valittuun ilmiöön. Yläkoululaiset opettelevat tekemään oman ohjelman ja ratkaisemaan siihen liittyviä ongelmia. Tarkoituksenmukaista on tutustua johonkin perusalgoritmi- tai tietorakennetyyppiin, esimerkiksi peruuttavaan tai ahneeseen algoritmiin tai puumaisiin tietorakenteisiin.

Lukioon soveltuu ensi kertaa ohjelmoiville samantyyppinen haaste, edistyneemmille ilmiöpohjainen projekti. Ilmiöpohjaisessa projektissa keskiössä on ongelmanratkaisu teknologiaympäristössä ja ohjelmointi itsessään on välineroolissa. Ohjelmointi voi siten liittyä esimerkiksi ilmiön simulointiin tai laskennalliseen tietojenkäsittelyyn, jossa graafisesti ratkaistaan lausekkeiden nollakohtia tai jossa tuotetaan grafiikkaa (esimerkiksi fraktaaleja) matemaattisten lausekkeiden avulla. Mekaanisten systeemien simuloinnissa PyBullet-ohjelmasta tai Unity-pelimoottorista voi olla hyötyä.

Yläkoulun ja lukion opiskelijat voivat pohtia ohjelmien käytettävyyttä, suorituskykyä ja tilavaativuutta eli helppokäyttöisyyttä, nopeutta ja muistitilatarpeita arkisten ilmiöiden reflektoinnin kautta.

Lukiossa joko aloitetaan ohjelmointitaidon kehittäminen, mikäli koulussa ei aiemmin ole ollut siihen tilaisuutta, tai mieluummin sovelletaan jo hankittua taitoa erilaisiin opiskelutarpeisiin. Ohjelmointi tai valmiiden ohjelmistojen hyödyntäminen saa välinearvon ja haasteena on ennemminkin oikeiden välineiden valinta käsillä olevaan haasteeseen. Välineeksi tässä katsotaan myös ohjelman loogisen mallin valinta.

Luonnolliselta tuntuu siis valita opiskelijan kiinnostusten kohteiden ja projektitarpeiden mukaan sopiva algoritmityyppi ja sitä tukeva ohjelmistoarkkitehtuuri ja tietorakenteet. Valintaa tehdessä muutkin vastaavantasoiset vaihtoehdot tulevat käyttötarkoituksiltaan tutuksi.

Sopivia ja käyttökelpoisia algoritmikategorioita ovat esimerkiksi ahneet algoritmit ja peruuttavat algoritmit. Ahneita algoritmeja käytetään esimerkiksi

ajanvarauksissa, kankaanleikkauksen ohjauksessa ja pakkaamisessa. Ajatuksena on hakea ajanvarauskalenterista tai kankaalta ensin suurin mahdollinen kappale ja vasta sen jälkeen kokojärjestyksessä muiden tarpeiden mukaisesti. Arkikokemuksesta tiedämme tämän menettelyn olevan järkevä, vaikka lähestymistavalle on esitetty teoreettistakin todistelua. Pakkaustehtävissä usein harjoitustehtävissä käytetään reppua, vaikka varsinaiset haasteet voivat olla esimerkiksi junien kuormaamista paperitehtaan jättirullilla tai laivakonttien pakkaamista keittiölaitteilla.

Peruuttavat algoritmit etsivät parasta mahdollista ratkaisua ehdokkaiden joukosta. Kaikki vaihtoehdot käydään systemaattisesti läpi ja niiden paremmuusjärjestys määrätään valitulla menetelmällä. Kaikki vaihtoehdot voidaan etsiä vaikkapa ratsun siirroille shakkilaudalla. Lisätietoa algoritmeista saa esimerkiksi Kisakoodarin käsikirjasta.

Ohjelmointiin ja tietojenkäsittelyyn liittyvää harrastustoimintaa on hyvä tuoda esiin monissa eri asiayhteyksissä.

Edistyneemmille yläkoululaisille tai lukiolaisille voi suositella Helsingin yliopiston MOOC-kursseja kuten Johdatus tietoliikenteeseen, Elements of AI, Web-palvelinohjelmointi, Cyber Security Base sekä Tietokoneen toiminnan perusteet -kurssi.

Alakoulussa yksityisyyteen, datan keräämiseen ja tietoturvaan liittyviä teemoja on jo hieman käsitelty. Yläkoulussa ja lukioissa näiden aiheiden käsittelyä jatketaan, sekä näiden kolmen lisäksi tarkasteluun otetaan datan keskittyminen ja uudet läpimurtoaan tekevät teknologiat. Erilaisten tulevaisuuden skenaarioiden avulla arvioidaan näiden kaikkien edellä mainittujen asioiden pitkän aikavälin yhteiskunnallisia vaikutuksia. Oppilaita on hyvä kannustaa käyttämään erilaisia yksityisyyttä suojaavia avoimen lähdekoodin ohjelmia.

Oppilaiden kanssa myös käsitellään tietoliikenteen perusteita ja konkreettisella tavalla tietoturvaa perusteellisemmin. Tietoliikenteen toiminnan, tiedostojen salaamisen ja kyberturvallisuuden osalta yliopistojen on tarpeellista tuottaa tarkoitusta varten tehtyjä opetusvideoita ja muita oppimismateriaaleja. Suomen Kyberturvallisuuskeskukselta voi myös löytyä hyödynnettävää materiaalia. Heiltä voi esimerkiksi tilata ilmaiseksi Turvallisesti netissä -opasvihkosia.

Kurssilta Algoritmisen ajattelun kehittäminen

19.05.20