JUHANI MYKKÄNEN | LINDA LIUKAS

KOODI2016

Ensiapua ohjelmoinnin opettamiseen peruskoulussa.

JUHANI MYKKÄNEN | LINDA LIUKAS

KOODI2016

Ensiapua ohjelmoinnin opettamiseen peruskoulussa.

TEKIJÄT

Ensimmäinen painos.

ULKOASU
Salla Koivu

VALOKUVAT
Maija Tammi

PAINOPAIKKA
Lönnberg Print
Helsinki, 2014

ISBN 978-952-93-4082-8 (sid.)
ISBN 978-952-93-4083-5 (sähk.)

TUKIJAT

Liikenne- ja viestintäministeriö
SITRA
Reaktor
SUPERCELL
Teknologia teollisuus
OHJELMISTOYRITTÄJÄT RY
Futurice
ROVIO
Startup Sauna FOUNDATION
DIACOR
TEK TEKNIIKAN AKATEEMISET
eficode

ESIPUHE

Marraskuu, 2013. Satakunta ihmistä puputtaa cocktailpaloja ravintola Virgin Oilissa Helsingin keskustassa. Ollaan Elinkeinoelämän valtuuskunnan tilaisuudessa, jossa Eva julkaisee yhteiskunnan tuloeroja käsittelevän pamfletin.

Me – **Linda Liukas** ja **Juhani Mykkänen** – osallistumme kumpikin tuohon tilaisuuteen ja saamme idean omasta pamfletistamme: tehdään näkyväksi se, mitä ohjelmointi on, miksi se on tärkeää ja miksi se pitäisi tuoda peruskouluihin.

Ajatus tulee mieleen, sillä Lindalla on taustaa ohjelmoinnin opettamisesta: hän on perustanut ohjelmointia naisille opettavan Rails Girls -hankkeen ja työskennellyt koodausta verkossa opettavassa Codecademyssä.

Juhani on sekä it-puolen diplomi-insinööri että tiedon selkeän esittämisen ammattilainen. Hän on työskennellyt Helsingin Sanomille edelliset kuusi vuotta ja toiminut muun muassa Nyt-liitteen esimiehenä.

Toteamme, että meidän kahden olisi syytä lyödä päämme yhteen. Alkaisimme poliittisesti ja taloudellisesti riippumattomaksi iskujoukoksi ja pyrkisimme helppotajuisesti avaamaan päättäjille ja opettajille, miksi ohjelmointi on tärkeää Suomen ja suomalaisten kannalta.

Olemme lähestyneet vasta paria ensimmäistä rahoittajaa työllemme, kun opetusministeri **Krista Kiuru** linjaa julkisesti: ohjelmointi tulee peruskouluun seuraavassa, vuoden 2016 opetussuunnitelmassa.

Olemme onnessamme.

Alkuperäinen etäinen haaveemme ohjelmoinnin roolin nostamisesta kouluissa on toteutumassa, vaikka olemme ehtineet vasta uneksia.

Tässä vaiheessa päätämme kohdentaa työmme uudestaan: opas tuleekin suunnata suoraan sekä peruskoulun opettajille että opetussuunnitelmien laatijoille. Heille, joille ohjelmointi ei ole tuttua puuhaa, mutta jotka joutuvat silti painiskelemaan ohjelmoinnin opettamisen kanssa.

Mitä konkreettisemmaksi suunnitelma opettajille laaditusta käytännönläheisestä oppaasta muotoutuu, sitä innokkaammiksi myös yhteistyötahot käyvät. Mukaan lähtevät lopulta Liikenne- ja viestintäministeriö, Sitra, Teknologiateollisuus ry, Ohjelmistoyrittäjät ry, Startup-säätiö ja Tekniikan akateemiset TEK sekä yrityksistä it-talot Reaktor, Futurice ja Eficode, peleistään tunnetut Supercell ja Rovio sekä terveydenhuoltaja Diacor.

Nyt edessäsi on opaskirjanen, joka on ensiapupakkaus. Se on tarkoitettu jokaiselle, joka haluaa parilla istumalla ymmärtää hiukan lisää siitä, mitä ohjelmointi on ja mitä tietokoneiden avulla voi luoda.

Ennen kaikkea kohderyhmä ovat peruskoulun opettajat ja opetusalan päättäjät.

Kun tämän oppaan selaa läpi, ohjelmointi ei toivottavasti enää tunnu täysin vieraalta asialta. Lukija saa käsityksen siitä, miksi ohjelmoinnin opettamisen aloittaminen on elintärkeää ja millaisista asioista oppitunnit voivat koostua.

Tämä ei ole oppimateriaali tai opetussuunnitelma vaan johdatus aiheeseen ja lisätietoon.

Tämä opas on ensiapupakkaus jokaiselle, joka haluaa ymmärtää, mitä ohjelmointi on.

Lopuksi pari sanaa oppaan riippumattomuudesta.

Ensinnäkin: Hankkeen tukijat eivät ole vaikuttaneet oppaan sisältöön. He ovat ainoastaan nähneet työmme tärkeänä ja halunneet mahdollistaa sen taloudellisella tuella.

Tukijat kuitenkin näkyvät oppaassa. Kunkin tahon edustaja kertoo, miksi juuri heidän mielestään ohjelmointiopetukseen panostaminen on tärkeää. Nämä puheenvuorot on toteutettu haastatteluina, ja ne löytyvät oppaasta *Miksi tämä on tärkeää?* -vinjetin alta. Kyseessä ei ole sponsorisanahelinä vaan olennainen osa oppaan sisältöä.

Oppaassa on myös pari kohtaa, jossa tietyn viestin välittämiseksi on lainattu jonkin tukijatahon edustajaa. Näissä tilanteissa sisältövalinnan olemme tehneet me.

Helsingissä 28. 5. 2014
Linda Liukas ja Juhani Mykkänen

TÄMÄ OPAS 9 KOHDASSA

1. **OHJELMOINTI TULEE** peruskouluihin syksyllä 2016. Ohjelmoinnista ei tule omaa oppiainettaan, vaan aikaa sen opetukselle otetaan matematiikan tuntijaosta. Ohjelmointia alkavat opettaa tavalliset opettajat kaikilla luokilla ykkösestä ysiin.

2. **YKSITTÄINEN OPETTAJA** on yhteys oppilaan ja ohjelmoinnin mahdollisuuksien välillä. Siksi tämä opas on suunnattu ensiavuksi opettajille ja opetusalan päättäjille. Opas kertoo, mitä ohjelmointi on, miksi se on tärkeää, miten opetussuunnitelma muuttuu ja miten ohjelmoinnin alkeita voi opettaa peruskoulussa.

3. **YHÄ SUUREMMAN** osan maailmasta rakentavat he, jotka osaavat luoda taidetta ja tiedettä tai vaikkapa älypuhelinsovelluksia ja internetpalveluita tietokoneen avulla. Siksi lapsilla ja nuorilla on oikeus oppia koulussa ohjelmoinnin perusasiat.

4. **PERUSKOULUSSA OPETETAAN** biologiaa, vaikka kaikista ei haluta biologeja. Samalla tavalla kaikista oppilaista ei pidä tulla koodareita – mutta kaikilla pitää olla mahdollisuus innostua siitä, mitä ohjelmoimalla voi rakentaa. 2000-luvulla ohjelmointikielen perusteiden ymmärtäminen on englannin kielen osaamisen tapaista yleissivistystä.

Jokaisella lapsella on oikeus oppia, mitä ohjelmoimalla voi luoda ja rakentaa.

5. **OHJELMOINNIN OPETTAMINEN** lapsille ja nuorille on välttämätöntä myös Suomen tulevaisuuden ja kilpailukyvyn kannalta. Huippukoodareista on pula jo nyt. Vuoteen 2019 mennessä Suomen ict-alalla on 17 000 työpaikan vaje. Muut maat, kuten Viro ja Britannia, ovat jo aloittaneet koodauksen opettamisen peruskoulussa.

6. **NAISET EDUSTAVAT** tällä hetkellä muutamaa prosenttia töissä olevista ohjelmoijista. Tuomalla ohjelmoinnin peruskouluun saamme parhaimmillaan moninkertaistettua osaavien naiskoodareiden määrän. Meillä ei ole varaa eristää ohjelmointia miesten etuoikeudeksi.

7. **OHJELMOINNIN OPETTAMINEN** peruskoulutasolla on ennen kaikkea ohjelmoinnillisen ajattelun perusteiden oppimista sekä tekemistä ja harjoittelua. Ohjelmoinnillisen ajattelun perusteita ovat esimerkiksi kyvyt pilkkoa ongelma osiin, antaa yksikäsitteisiä komentoja tietokoneelle ja pohtia, mitkä komennot missä järjestyksessä ratkaisevat ongelman. Suomalaisen opettajan ei tarvitse huolia: hän on maailman parhaimmistoa ja oppii tällaisten asioiden opettamisen.

8. **TÄMÄ OPAS** noudattaa opetussuunnitelman raameja: 1–2-luokilla ohjelmointia opetetaan leikkien avulla. Ne opettavat antamaan yksikäsitteisiä komentoja. 3–6-luokilla tutustutaan verkosta löytyviin visuaalisiin ohjelmointiympäristöihin ja opitaan käskemään tietokonetta ilman pelkoa virheiden tekemisestä. 7–9-luokilla perehdytään johonkin oikeaan ohjelmointikieleen esimerkiksi alkeista lähtevän verkkokurssin avulla.

9. **TÄSTÄ SELVITÄÄN** hienosti. Nyt on tärkeää informoida ja täydennyskouluttaa opettajat. Ohjelmointia ei kuitenkaan opita vain peruskoulussa. Apua tarvitaan vanhemmilta, iltapäiväkerhoilta ja yrityksiltä sekä peruskoulun jälkeisiltä oppilaitoksilta. Tärkeintä on antaa jokaiselle oppilaalle peruskäsitys ohjelmoinnista ja taata edistymisen mahdollisuudet niille, jotka innostuvat eniten. Näin tulevaisuuden Suomea rakentavat jo lapsena aloittaneet koodaritaiturit.

SISÄLLYS

JOHDANTO

Lapsilla on oikeus ymmärtää, mitä tietokoneiden avulla voi rakentaa. Ohjelmoinnin tulo peruskouluun voi muuttaa Suomen tulevaisuuden.

Heidän nuoruutensa tulee olemaan aivan erilainen kuin sinun tai minun. Kuten myös heidän aikuisuutensa. Siis lasten, jotka aloittavat koulun syksyllä 2016.

He tulevat oppimaan 7-vuotiaasta alkaen, miten tietokoneen avulla rakennetaan asioita.

Miten koneelle annetaan ohjeita siitä, kuinka piirtää kokonainen maailma elokuvaan? Kuinka kone opetetaan selvittämään, mikä tauti mummoa vaivaa?

Suomalaisten ihmisten, yritysten ja koko maan tulevaisuuteen vaikuttaa, innostuvatko nämä 2000-luvulla syntyneet ihmiset ohjelmoinnista.

Saavatko he jo nuorena vihiä siitä, mitä kaikkea pelkällä näppäimistöllä voi rakentaa, parantaa ja luoda?

Kerrotaanko heille siitä kaikesta innostavasti?

Opetusministeri Krista Kiuru linjasi tammikuussa 2014, että ohjelmointi tulee osaksi peruskoulun opetusohjelmaa.

Muutos on tärkeä. Opetussuunnitelmaa muutetaan kerran noin 10 vuodessa. Seuraava muutos tapahtuu 2016, joten sitä seuraava kierros olisi jo auttamattoman myöhässä. Esimerkiksi Britanniassa ohjelmointia aletaan opettaa jo syksyllä 2014.

2016 suomalaiset koulut ottavat käyttöön Opetushallituksen johdolla muokatun opetussuunnitelman, joka määrittelee, mitä asioita nykynuorten tulee koulussa koodaamisesta oppia.

Nyt käsissäsi oleva opaskirja (ja sen verkkoversio) on syntynyt halusta vaikuttaa siihen, miten ohjelmointia lähestytään koulussa.

Mistä tässä oppaassa sitten on kyse? Olemme halunneet tehdä tiiviin ja ymmärrettävän esityksen neljästä asiasta:

1. Mitä ohjelmointi on ja miksi sen opettaminen on niin tärkeää?
2. Miten opetussuunnitelma muuttuu ja miksi?
3. Miten ohjelmointia voi opettaa peruskoulussa?
4. Mitä muut kuin koululaitos voivat tehdä?

Ensimmäinen kohta – ja samalla oppaan ensimmäinen osio – liittyy siihen, että vaikka monella on jonkinlainen käsitys siitä mitä ohjelmointi on (”tietokoneella kirjoitetaan jotain, ja tuloksena syntyy ohjelmia tietokoneille ja älypuhelimille”), käsitys on hatara. Kynnys päästä aivan perusymmärryksestä pykälää pidemmälle on suuri – varsinkin, jos ei ole tiedossa, miksi asiasta pitäisi olla ylipäätään kiinnostunut.

Haluamme siis tarjota lukijalle ikään kuin tiiviin lunttilapun: tästä on yleisesti ottaen kyse, esimerkiksi tältä ohjelmakoodi voi näyttää, tällaisia asioita ohjelmoimalla voi saada aikaan.

Toisessa osassa valotamme, miten opetussuunnitelma näillä näkymin muuttuu syksyllä 2016. Vastaamme myös opettajien pelkoihin ja kysymyksiin aiheesta.

Kolmannessa osassa tarjoamme konkreettisen apumme peruskoulussa käsillä olevaan muutokseen: Tällaisia asioita ohjelmoinnin opetuksessa kannattaa huomioida, tällaisia apuvälineitä voi käyttää. Tällaiset harjoitukset innostavat, täältä niitä löytää.

Oppaan kolmannessa osiossa kerrotaan, miten ohjelmointia voi opettaa.

Neljännessä osassa esitämme haasteita päättäjille, koodareille, vanhemmille ja meille kaikille.

Olemme laatineet esityksemme pohjaten kymmeniin haastatteluihin opettajien ja opetuksen asiantuntijoiden sekä tietenkin ohjelmoijien ja ohjelmoinnin opetuksen tuntijoiden kanssa. Pohjaamme myös verkossa saatavilla oleviin tietolähteisiin ja projektin aikana lukemattomilta vapaaehtoisilta tulleisiin ideoihin.

Ohjelmoinnin opettamiseen liittyy selvästi erilaisia epävarmuuksia. Siksi olemme pyrkineet olemaan mahdollisimman kouriintuntuvia.

Totuuden hetki koulussa on aina opettajan ja oppilaan välinen vuorovaikutus.

Oppilaan kannalta tavoitteenamme on, että hän saisi peruskoulussa – jo lapsena – mahdollisuuden innostua ohjelmoinnista.

Tässä osiossa kerrotaan, mitä ohjelmointi on ja miksi se on tärkeää.
Lukija saa esimerkkejä siitä, mitä ohjelmoija tekee työkseen
ja miltä varsinainen ohjelmakoodi näyttää.

1 | Miksi ohjelmointi on tärkeää?

MITÄ OHJELMOINTI ON?

Jo kuusivuotias saattaa osata tehdä voileivän, mutta tietokoneelle se on vaikeaa.

Minkä leivän valitsen? Milloin lopetan leikkaamisen?

Mitä tapahtuu, jos juusto loppuu kesken?

Kun tietokone – tai leivän tekemisen tapauksessa vaikkapa tietokoneen ohjaama robotti – laitetaan leiväntekoon, pienkinkin yksityiskohta on selitettävä täydellisen tarkasti ja yksikäsitteisesti. Kaikkiin mahdollisesti eteen tuleviin pulmiin on varauduttava.

Mutta selittäminen kannattaa.

Kun tietokone lopulta osaa tehdä yhden leivän, se tekee sen täydellisesti. Joka kerta. Toisin kuin kuusivuotias tai edes aikuinen, yhden voileivän tehtyään tietokone voi tehdä tuhansia ja taas tuhansia leipiä. Se ei koskaan kyllästy.

Tietokone on kärsivällinen ja tarkka, varma ja nopea. Siksi se on loistava työkumppani tekemään asioita ihmisen ohjeiden perusteella.

IHMINEN KÄSKEE TIETOKONETTA

Tietokone ei tiedä itse mitään.

Tai tarkemmin sanoen se ei tiedä mitään, mitä ihminen ei ole sille opettanut (tai edistyneimpien tekoälyjen tapauksessa: mitä ihminen ei ole sitä opettanut oppimaan). Se ei myöskään tee mitään, mitä ihminen ei ole käskenyt sitä tekemään.

Se, että voimme pelata videopeliä, surffata netissä tai ajaa nykyaikaista autoa perustuu sille, että ohjelmoija on kertonut tietokoneelle tarkalleen, miten toimia.

Pelin tapauksessa ohjelmoija kertoo koneelle, millaisia hahmoja ruudulle piirretään, minkä näköisessä maailmassa ne liikkuvat, ja mitä hahmon pitää ruudulla tehdä, kun pelaaja painaa ohjaimesta nappia.

Netti puolestaan koostuu toisiinsa yhteydessä olevista tietokoneista. Ihmisen on pitänyt kertoa koneille, millaisten sääntöjen mukaisesti ne lähettävät toisilleen dataa, josta taas muodostuvat lopulta käyttäjän ruudulla näkyvät tekstit, kuvat ja videot.

Dataa ei saa hävittää matkalla, ja yksittäisen verkkosivun pitäisi toimia hyvin sekä matkapuhelimella että videotykillä katseltuna. Koneelle on annettava ohjeet eri tilanteisiin.

Kaupasta saa jo auton, joka osaa parkkeerata itsensä. Pian meillä on autoja, jotka ajavat kaupungissa itse itseään. Nämä molemmat asiat ovat mahdollisia, koska ihminen on pohtinut kaikkia mahdollisia tilanteita, joita autoa ajaessa voi tulla vastaan. Sitten hän on antanut koneelle ohjeet siitä, miten näissä tilanteissa toimitaan.

Miten kaikki nämä asiat opetetaan tietokoneelle?

Sitä kutsutaan ohjelmoinniksi.

OHJELMOINTI ON OHJEIDEN ANTAMISTA TIETOKONEELLE

Ohjelmointia voisi verrata siihen, että ruoanlaiton osaava kokki kirjoittaa reseptin aloittelevalle leipurille.

Ohjeiden on oltava niin täsmällisiä, että niitä ei voi mitenkään ymmärtää väärin, jos niitä seuraa täydellisen tarkasti. Ohjeiden on oltava tarpeen mukaan muutettavia. Niiden on katettava kaikki mahdolliset tilanteet ja kysymykset, jotka uudelle leipurille saattavat tulla mieleen. Ohjeiden on myös oltava juuri oikeassa järjestyksessä.

Ohjelmointikieliä on satoja, mutta niiden kaikkien perusajatus on, että ohjelma saa tietoja – esimerkiksi ihmiseltä –, käsittelee tiedot annettujen sääntöjen mukaan ja saa aikaan halutun lopputuloksen.

Kaikkein teknisimmällä tasolla ajateltuna tietokoneessa on muistipaikkoja, joihin voi tallentaa ykkösiä ja nollia. Lyhyiden tai hyvin, hyvin pitkien ykkösten ja nollien jonojen avulla voidaan kuvata esimerkiksi kirjaimia, koneelle annettavia komentoja tai vaikkapa kokonaisia valokuvia.

Kaikki asiat joiden parissa tietokone työskentelee – Googlelle annetut hakusanat, uuden ihmisen kirjaaminen Facebook-kavereiden listaan sekä herätyskellon soiminen – kääntyvät lopulta jonoihin ykkösiä ja nollia, jotka tietokone käsittelee.

Ohjelmoija ei kuitenkaan kirjoita ykkösiä ja nollia. Hän naputtelee hitusen normaaliin puhekieleen verrattavissa olevia komentoja, jotka kone ymmärtää, koska ne on merkitty tarkalleen yhteisesti sovitulla tavalla.

Osa kielistä on nopeakäyttöisiä, osa on mahdollisimman luotettavia. Osaa voi lukea kuin englantia.

Kielet ovat kehittyneet kuin talonrakennustarvikkeet: jos joskus on pitänyt aloittaa valamalla

omat tiilensä, nyt taloja rakennetaan valmiista elementeistä. Nykyohjelmoijat rakentavat vankkojen perustusten päälle, niin paljon helppokäyttöisemmiksi ja enemmän valmiita palikoita sisältäväksi esimerkiksi työelämässä käytettävät ohjelmointikielet ovat kehittyneet muutamassa vuosikymmenessä.

Kuten missä tahansa kielissä, myös ohjelmointikielissä on oma sanastonsa ja kielioppinsa. Koodi muodostuu kullekin ohjelmointikielelle kuuluvista ilmaisuista (tätä kutsutaan *syntaksiksi*), sekä sanoista ja kuvauksista, jotka ohjelmoija luo omiin tarpeisiinsa kielen sääntöjen puitteissa.

Koodia luetaan rivi riviltä, mutta rivien joukot muodostavat kokonaisuuksia, jotka edustavat jotain tiettyä tehtävää. Yhdellä rivillä saatetaan esimerkiksi tallentaa muuttujaan tietty lukuarvo, mutta yhdistettynä tuo rivi ja sen yllä ja alla olevat rivit voivat yhdessä muodostaa kokonaisuuden, jossa esimerkiksi pankin tietokanta saa tietää, että ihmisen pankkitilille talletetaan rahaa. Tällöin esimerkiksi muuttujaan ”saldo” tallennetaan lukuarvo siitä, paljonko tilillä on talletuksen jälkeen rahaa.

Jos kuulostaa monimutkaiselta, ei huolta. Tästä kaikesta on tuonnempana esimerkkejä!

Hyvällä ohjelmoijalla on rikas sanavarasto eli käsitys ohjelmointikielen tarjoamista työkaluista. Eikä haittaa, vaikka osaisi useampaa kieltä.

Kuten missä tahansa kielissä, myös ohjelmointikielissä on oma sanastonsa ja kielioppinsa.

Teknologia ja kielet myös kehittyvät koko ajan. Siksi ohjelmoija saa käyttöönsä vuosi vuodelta tehokkaampia työkaluja pulmiensa ratkaisuun.

Nykyaikainen ohjelmointi on luovaa ongelmanratkaisua. Ohjelmoijan tehtävä on ratkaista ongelma, esimerkiksi rahan siirto pankin tietojärjestelmässä tililtä toiselle. Tätä varten ohjelmoija muuttaa ongelman pala palalta muotoon, jonka tietokone ymmärtää.

Usein ohjelmoijan tehtävänä on kirjoittaa ohjelma, joka ottaa vastaan tietoa, käsittelee sitä ja tulostaa ulos uutta tietoa.

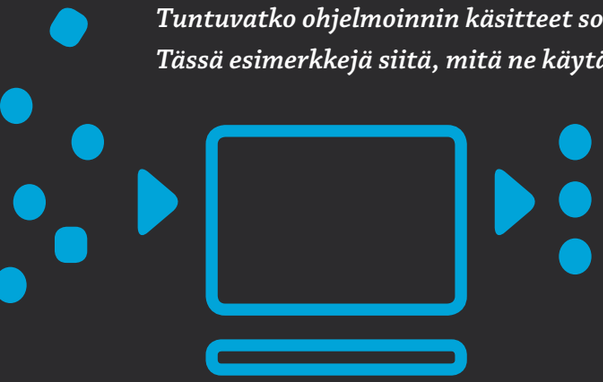
Esimerkiksi uuden kirjan lisääminen ostoskoriin Amazonin verkkokirjakaupassa saa aikaan Amazonin tietokoneilla uusia pieniä komentoja. Niistä yksi muuttaa asiakkaan ostoskorin sisältämien kirjojen lukemaa, toinen muokkaa verkkokaupassa jäljellä olevien kirjojen määrää ja kolmas päivittää listaa kirjoista, joita asiakkaalle jatkossa suositellaan.

”Olen aina valmis oppimaan – vaikka en aina pidäkään siitä, että minua opetetaan.”

– Winston Churchill

HYVIN LYHYT OHJELMOINTISANASTO

Tuntuvatko ohjelmoinnin käsitteet solmuisilta?
Tässä esimerkkejä siitä, mitä ne käytännössä tarkoittavat.



TIETOKONE. Tietokoneen tehtävä on yksinkertaistettuna ottaa vastaan tietoa, suorittaa ohjelma, joka käsittelee tietoa ihmisen kannalta aiempaa hyödyllisempään muotoon ja näyttää lopputulos. Ohjelmoijan tehtävänä on varmistaa, että tietokone saa tiedon muodossa, jonka se ymmärtää, kirjoittaa ohjelma jonka tietokone ymmärtää – ja ihailla lopputulosta.

MUUTTUJA. Muuttuja on ohjelmakoodissa nimetty paikka, jonka sisälle voi tallentaa erilaista tietoa. Tieto voi olla esimerkiksi numero, pätkä tekstiä tai tieto siitä, onko jokin väite totta tai ei.

Ajatellaan vaikkapa tilannetta, jossa asiakas tulee pankkivirkailijan luo ja antaa tilinumeronsa:

Virkailijan syöttämä tilinumero kulkeutuu pankin tietokantaan, jossa ohjelmisto tarkistaa, löytyykö tietyllä tilinumerolla tietokannasta tili. Tällöin tietty muuttuja saa arvon ”totta” viitaten siihen, että kyseinen tili todella on juuri tässä pankissa.

Seuraavaksi toiseen muuttujaan voidaan tallentaa esimerkiksi tieto siitä, paljonko tilillä on

rahaa ja välittää muuttujan mukana tieto pankkivirkailijan näytölle. Virkailija voi tällöin kertoa saldon tilinumeron antaneelle asiakkaalle.

ALGORITMI. Algoritmi on kuvaus jonkin tehtävän suorittamiseksi tarvittavista toimenpiteistä. Esimerkiksi tavallinen voileivän tekeminen voidaan esittää algoritmina, joka alkaa leivän ottamisesta pussista ja päättyy voin laittamiseen takaisin jääkaappiin. Ohjelmoinnin perusopinnoissa algoritmit liittyvät usein esimerkiksi asioiden lajitteluun ja luokitteluun.

FUNKTIOT. Funktiot ovat ohjelman sisäisiä pieniä itsenäisiä osasia. Funktion tarkoitus on, että konetta opetetaan kerran tekemään jokin asia, ja kun kone on sen oppinut, asian tekemiselle annetaan nimi. Esimerkiksi näytölle voitaisiin piirtää yksinkertainen talo komennolla ”piirräTalo”.

Kun sama asia pitää tehdä uudestaan, konetta riittää kutsua ”piirräTalo”-komennolla. Talo piirtyy silloin uudestaan, kuten aiemmin on opetettu.

Yhdessä funktiossa voi olla monta algoritmia, ja yhtä funktiota voi käyttää uudelleen ja uudelleen eri paikoissa ohjelmakoodia.

Ohjelmoinnissa ei ole järkeä, jos saman asian pystyy tekemään nopeammin itse. Yksinkertaisen talon piirtäminen kerran on kätevää käsin. Mutta jos tarkoitus on piirtää sata taloa esittämään vaikkapa tietokoneella tehdyssä piirroselokuvassa kaupunkia, on helpompaa pyytää piirraTalo-funktiota piirtämään sata taloa. Hieman funktiota kehittelemällä se osaa jopa piirtää eri kokoisia ja eri värisiä taloja eri paikkoihin.

LISTA. Ohjelmointikielissä on paljon erilaisia tapoja säilyttää tietoa ja järjestellä sitä. Listat ovat eräänlaisia lokerikkoja, joista voi poimia muuttujia helposti, laskea kuinka monta muuttujaa yhteensä on ja käsitellä tietoa kokonaisuutena. Voileivän muodostavat ainekset voivat alhaalta ylös olla järjestyksessä leipä, voi, makkara ja juusto. Kun ne laitetaan listaan, lista tietää, että yhteensä ainesosia on neljä, ja kolmas niistä on makkara.

TIETOKULMA

TOISTORAKENNE JA EHTOLAUSE. Tietokoneet tekevät päätöksiä ehdollisten rakenteiden avulla. Ehtolauseiksi kutsutaan rakenteita, joissa tietokoneita opetetaan tekemään päätöksiä. Jos makkara on loppu, laita leivän päälle vain juustoa. Jos juustokin on loppu, anna ilmoitus, että leivän täytteet ovat lopussa.

Toistorakenteiksi tai silmukoiksi kutsutaan taas sitä, että tietokone toistaa jotakin asiaa, kunnes toinen asia tapahtuu. Esimerkiksi: laita kunkin leivän päälle siivu makkaraa, kunnes makkara loppuu.

VALOKUVAT: MAIJA TAMMI

Valokuvaaja **Maija Tammi** teki *Koodi2016*-opasta varten tilaustyönä teossarjan *Sekunti*. Sarjan jokainen kuva koostuu 24 eri kuvasta. Valinta viittaa elokuvan tekniikkaan, jossa liikkeen illuusio luodaan näyttämällä 24 kuvaa sekunnissa.

Sekunnin tapa hajottaa kuva osiin liittyy myös ohjelmoinnilliseen ajatteluun (*computational thinking*, ks. sivut 76–78): ohjelmoinnissa ongelma tulee osata jakaa pienempiin osiin, hahmottaa kaavamaisuuksia ja osata soveltaa havaintoja uudessa tilanteessa.

Teoksia on kuusi, ja ne on siroteltu ympäri opasta. Ensimmäinen on seuraavalla aukeamalla. Loput ovat aukeamilla 38–39, 60–61, 82–83, 110–111 ja 134–135.

Kuvat sisältävät taiteilijan lyhyet saatesanat.

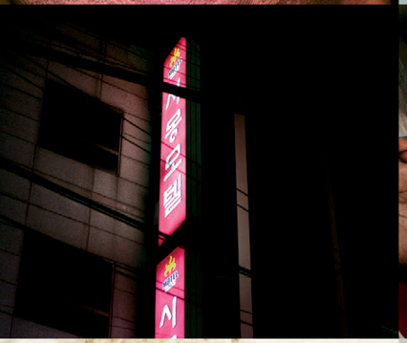
Valokuvaajasta

MAIJA TAMMI (s. 1985) on suomalainen valokuvaaja ja taiteilija. Tammi on palkittu sekä Vuoden kuvajournalistina 2010 että Fotofinlandia-palkinnolla 2011. Hänen työnsä ovat olleet esillä näyttelyissä niin Suomessa, Ranskassa, Espanjassa kuin Yhdysvalloissa. Tällä hetkellä Tammi tekee taiteellista väitöskirjaa Aalto-yliopistossa Helsingissä.
maijatammi.com




5379757

Kuvaaja Maija Tammi:
”Ihmiset käyttävät numeroita 0–9, tietokoneet vain numeroita 0 ja 1. Kuvat muodostavat binääriluvun 01010010000101101010101, joka vastaa kymmenjärjestelmän lukua 5379757.”





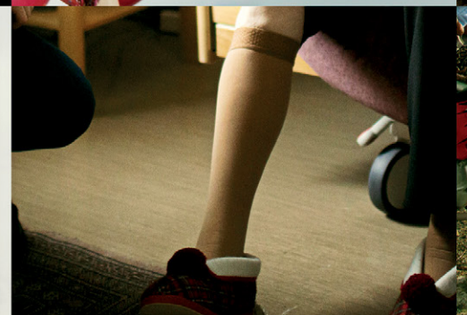













MILLAISIA ASIOITA OHJELMOIJA LUO?

Seuraavassa on esitelty monipuolinen mutta satunnainen valikoima oikeaa elämää sivuavia, yksinkertaistettuja esimerkkejä siitä, millaisissa yhteyksissä tarvitaan ohjelmointitaitoja – aina Harry Potter -elokuvien tekemisestä itsensä parkkeeraavaan autoon ja jätteitä lajittelevaan robottijärjestelmään.

WHATSAPP-VIESTIOHJELMA

Ohjelmoijan tehtävänanto: Luo älypuhelimelle sovellus, jonka avulla ihmiset voivat lähettää toisilleen perinteisten tekstiviestien tapaisia viestejä, mutta internetin välityksellä.

Mitä ohjelmoija tekee: Ohjelmoija pohtii esimerkiksi, miten viestit saadaan kulkemaan puhelimesta toiseen käyttäen puhelinverkon sijaan nettiyhteyttä. Ohjelmoijan haaste on ohjeistaa tietokonetta siten, että viestit kulkevat mahdollisimman nopeasti, ja sovelluksen toiminta ei hidastu, vaikka sitä käyttäisivät kymmenet miljoonat ihmiset yhtä aikaa.

Huomioitavaa: WhatsAppin kaltaisessa sovelluksessa äärimmäisen tärkeää on, että sovellusta on helppo ja miellyttävää käyttää. Siksi WhatsAppin tapauksessa käyttöliittymän suunnittelijalla on valtava rooli: sovelluksen pitää näyttää ymmärrettävältä ja intuitiiviselta välittömästi.

ITSENSÄ PARKKEERAAVA AUTO

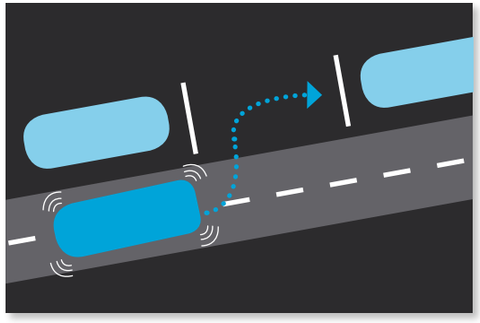
Ohjelmoijan tehtävänanto: Suunnittele autoa ohjaava ohjelmisto, joka osaa osaa taskuparkkeerata auton itsestään.

Mitä ohjelmoija tekee: Ohjelmoija saa tietoja auton etu- ja takaosissa olevilta sensoreilta. Ne kertovat esimerkiksi, kuinka kaukana auto on milläkin hetkellä katukivetyksestä, muista autoista tai ohi kulkevista ihmisistä. Parkkeerattava auto ei saa törmätä mihinkään näistä.

Ohjelmoija opettaa auton sisällä olevalle tietokoneelle sääntöjä: kun olet näin kaukana katukivetyksestä, käännä renkaita tämän verran ja paina kaasua tämän verran. Kun olet näin ja näin lähellä estettä, pysähdy heti, jotta et törmää.

Kun ohjelmoija on pohtinut kaikki mahdolliset tilanteet ja asennot joissa auto voi olla, hän antaa tietokoneelle toimintaohjeet eri tilanteisiin. Tämän jälkeen tietokone osaa automaattisesti parkkeerata auton juuri oikein ja tarkasti yhä uudestaan.

Huomioitavaa: Itsensä parkkeeravan auton kehittäminen vaatii rutkasti tuotekehitystä, yrityksiä ja erehdyksiä. Kaikkia eri tilanteita on lähes mahdoton kuvitella ennen kuin autoa on testattu erilaisissa arkielämän tilanteissa yhä uudestaan. Vasta vuosien testien jälkeen auto on valmis kauppoihin.



Itsensä parkkeerava auto aistii ympäristöään sensoreilla ja osaa siksi taskuparkkeerata törmäämättä.

HARRY POTTER -ELOKUVAT

Ohjelmoijan tehtävänanto: Harry Potter -elokuvissa pelataan Huispausta. Huispaus on peli, jossa Tylypahkan oppilaat jahtaavat lentävillä luudilla Sieppi-nimistä lentävää esinettä. Suuri osa kohtauksista on animoitava tietokoneella, koska Sieppiä tai lentäviä luutia ei ole oikeassa elämässä. Suunnittele ohjelma, jonka avulla voidaan piirtää Siepille erilaisia lentoratoja.

Mitä ohjelmoija tekee: Ohjelmoija lukee ensin Harry Potter -kirjat ymmärtääkseen, miten Sieppi lentää. Hän oppii, että Sieppi pysähtyy välillä leijumaan paikoilleen ja singahtaa sitten johonkin satunnaiseen suuntaan.

Ohjelmoija opiskelee tai palauttaa mieleensä

hiukan matematiikkaa ja fysiikkaa. Niiden pohjalta hän muodostaa esimerkiksi kiihtyvyyden, painovoiman ja nosteen kaavojen tapaisia yhtälöitä. Niitä tarvitaan, jotta tietokonekin ymmärtää, miten Siepin liikeradat pitää mallintaa, jotta ne näyttävät ihmisen silmään luonnollisilta.

Ohjelmoija kokeilee, millaiset kaavat saavat yksinkertaisen palluran liikkumaan ruudulla, kuten hän haluaisi Siepin liikkuvan elokuvassa. Kun ohjelmoija on tyytyväinen, hän luovuttaa ohjelman animointiosaston ihmisille. He animoivat varsinaiseen elokuvaan yksinkertaisen palluran tilalle oikean Siepin näköisen esineen.

Huomioitavaa: Tosielämässä elokuvien erikoistehosteita suunnitellaan ohjelmilla, jotka tuntevat valmiiksi valtavan määrän fysikaalisia liikkeitä. Tämän vuoksi Harry Potterin kohdalla ei ole luultavasti tarvinnut lähteä enää alusta, vaan ohjelmoija on selvinnyt jalostamalla olemassaolevaa ohjelmakoodia.

SAIRAALAN POTILASTIETOJÄRJESTELMÄ

Ohjelmoijan tehtävänanto: Luo järjestelmä, johon sairaala tallentaa jokaisen uuden potilaan tiedot, ja josta lääkärit voivat tutkia potilaan historiaa.

Mitä ohjelmoija tekee: Ohjelmoija ottaa selvää, mitä tietoja potilaasta pitää tallentaa. Esimerkiksi nimi, osoite, ikä, sukupuoli, pituus, paino, aiemmat sairaudet, aiemmat sairaalakäynnit, rokotukset, leikkaukset, lääkitykset ja niin edelleen.

Ohjelmoija käskee tietokonetta muodostamaan tältä pohjalta tietokannan, johon hän varaa tilaa kaikille tarvittaville tiedoille ja kymmenilletuhansille potilaille. Ohjelmoija työskentelee yhdessä käyttöliittymäsuunnittelijan kanssa luodakseen sairaalan henkilökunnan tietokoneille näkymän, jonka avulla tietoja on helppo syöttää järjestelmään ja toisaalta saada sieltä tarvittaessa ulos.

Huomioitavaa: Terveydenhuollon tietojärjestelmät ovat monimutkaisia hankkeita, koska vanhat ja uudet järjestelmät joutuvat usein olemaan yhteyksissä toisiinsa. Tämä luo haasteita ohjelmointiprojekteihin, sillä vanhat järjestelmät on saatettu ohjelmoida tavoilla ja kielillä, joilla ei ole paljonkaan yhteistä nykyaikaisten tekniikoiden kanssa.

ANGRY BIRDS -MOBIILIPELI

Ohjelmoijan tehtävänanto: *Angry Birds* -pelissä yritetään ampua ritsalla lintuja siten, että ne osuvat vihreisiin possuihin. Possut ovat linnoittautuneet puusta, kivistä tai vaikkapa jäästä tehtyihin linnakkeisiin. Ohjelmoi peli siten, että kun lintu osuu possun linnakkeeseen, linnake hajoaa tavalla, jolla sen voisi kuvitella hajoavan oikeassa elämässä.

Mitä ohjelmoija tekee: Ohjelmoija selvittää pelin luovalta tiimiltä, millaisia linnakkeita pelissä tarvitaan. Puu on heikompaa kuin jää, ja jää on heikompaa kuin kivi. Kaikki materiaalit hajoavat siis eri tavalla, kun lintu osuu niihin. Ohjelmoija saa peliä pyörittävältä niin kutsutulta pelimoottorilta jo valmiiksi pelin sisäistä dataa siitä, missä kulmassa ja millä nopeudella lintu tulee linnaketta kohti.

Hänen tehtävänään on opettaa tietokoneelle, miten linnakkeen pitää sortua, kun lintu osuu sen eri kohtiin.

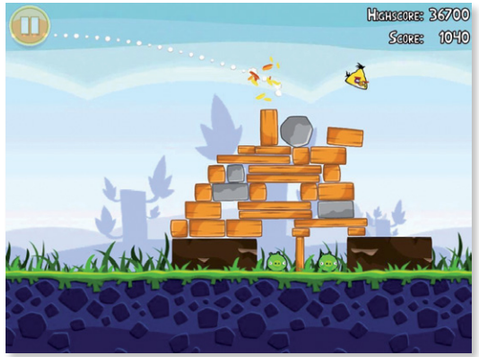
Ohjelmoija opettaa tietokoneelle luonnonlakeihin liittyviä matemaattisia kaavoja: painovoiman, kitkan, liikemäärän ja niin edelleen. Kun ne on opetettu tietokoneelle oikein, linnakkeet alkavat sortua kuten ne sortuisivat oikeassa elämässä.

Tämän jälkeen ohjelmoija voi tehdä muutoksia esimerkiksi siten, että kivi ei olekaan ihan yhtä kestävää kuin oikeassa elämässä, tai että sininen lintu särkee jäätä tehokkaammin kuin punainen lintu.

Huomioitavaa: Esimerkki on kirjoittajien kuvittelema. Pelien kehittelyssä paukut käytetään usein aluksi siihen, että luodaan niin sanottu työkalukokoelma, jonka avulla pelin kehittäminen ja testaaminen on helpompaa.

Esimerkiksi *Angry Birdsin* tapauksessa voidaan aluksi aloittaa ohjelmoimalla työkalu, jolla pelin kehittäjä voi rakentaa ruudulle erilaisia linnakkeita yksinkertaisesti raahaamalla näytöllä puu- ja kivipalikoita erilaisiksi kasoiksi. Kun tuo työkalu on valmis, yhä uusien pelin maailmojen ja tasojen luominen on helppoa, sillä pohjatyö on tehty kunnolla.

Asiaa voisi verrata siihen, että puita sahaava firma kehittää moottorisahan: tavallisella sahalla pääsee nopeammin alkuun, mutta kun moottorisaha on lopulta saatu kehitettyä, sillä tekeminen on paljon tehokkaampaa. Se ottaa äkkiä tavallisella sahalla saadun etumatkan kiinni ja menee pian heittämällä ohi.



Angry Birds *-pelissä pommitetaan linnuilla vihreiden possujen linnakkeita. Kuva: Rovio.*

JÄTTEITÄ LAJITTELEVA ROBOTTI

Ohjelmoijan tehtävänanto: Käytössämme on robottikoura (ks. oikean palstan kuva), joka osaa tarttua liukuhihnalla kulkeviin asioihin. Lisäksi meillä on digitaalinen videokamera, joka kuvaa liukuhihnaa. Liukuhihnalla kulkee sekaisin kiviä sekä roskia. Opeta robottikouraa poimimaan hihnalta kivet roskien seasta.

Mitä ohjelmoija tekee: Ohjelmoija laittaa videokameran ottamaan tuhansia kuvia liukuhihnalla kulkevasta materiaalista. Ohjelmoija näyttää kuvia tietokoneelle ja kertoo sille, missä kohden kuvissa on kivi.

Ohjelmoija käskee konetta tallentamaan muistiin kaikki kuvat ja tiedon siitä, missä kohden niissä näkyy kivi. Kun tätä on jatkettu jonkin aikaa, kone alkaa itsenäisesti hahmottaa, millaiset asiat kuvassa ovat luultavasti kiviä.

Seuraavaksi ohjelmoija antaa koneen yrittää kivien tunnistamista itse. Hän näyttää koneelle kuvia, ja kone yrittää arvata, missä kohden kivi on.

Kun kone onnistuu, ohjelmoija kertoo sille, että kone oli oikeassa. Kun kone epäonnistuu, ohjelmoija kertoo, ettei kohdassa ollut kiveä.

Näin tietokone oppii yhä lisää, ja ohjelmoija säästää kivien osoittamiseen kulunutta omaa aikaansa.

Kun tätä on jatkettu tarpeeksi kauan, kone alkaa tunnistaa kiven esimerkiksi 9 kertaa 10:stä. Tässä vaiheessa prosentti saattaa – käyttötarkoituksesta riippuen – olla jo riittävän hyvä siihen, että koneen voi laittaa ohjaamaan robottikouraa, jonka tehtävä on poimia kiviä.

Huomioitavaa: Suomalainen ZenRobotics-yritys valmistaa jätteenlajittelujärjestelmää, jonka yhtenä osana on yllä kuvaillun kaltainen robottikoura. Kouran ja sitä ohjaavan tietokoneen tarkoitus on lajitella erilaisia käyttökelpoisia asioita talteen rakennusjätteen seasta.

Yllä kuvattu työ on tarkemmin sanottuna tekoälyn kehittämistä, jossa yhtenä osa-alueena tarvitaan ohjelmointitaitoa.

Suomalaisen ZenRobotics-yrityksen tekoälyllä toimiva jätteidenlajittelujärjestelmä. Kuva: ZenRobotics.

MUSIIKKITAIDETEOS KIASMAAN

Ohjelmoijan tehtävänanto: Luo Kiasmaan taideteos, joka soittaa tyhjässä valkoisessa huoneessa erilaista äänimaisemaa sen perusteella, kuinka monta ihmistä huoneessa on.

Mitä ohjelmoija tekee: Ohjelmoija tai hänen apurinsa asentaa huoneen kattoon nettikameran ja kytkee sen tietokoneeseen. Tietokoneen hän kytkee myös kaiuttimiin. Ohjelmoija haluaa, että huoneessa olevat ihmiset vaikuttavat siellä kuuluviin ääniin – toisin sanoen äänten pitää olla erilaisia riippuen siitä, millaista kuvaa verkkokamera välittää huoneesta.

Ohjelmoija avaa valmiin musiikinteko-ohjelman, jonka avulla voi soittaa tietokoneversioita erilaisista instrumenteista. Seuraavaksi ohjelmoija kirjoittaa koodia, joka yhdistää verkkokamerasta tulevan kuvan musiikkiohjelmaan. Kun kuvassa näkyy pelkkää valkoista, kuvassa ei selvästikään ole ketään. Tällöin huoneessa voi soida esimerkiksi pelkkä tasainen pianon pimputus, joka tulee ohjelmistosta.

Kun joku kävelee kuvaan, ohjelmoija voi opettaa tietokoneelle esimerkiksi, että vasemmassa laidassa näkyvä liike käynnistää musiikkiohjelmasta viidakkorumpuääniä. Jokainen oikeassa laidassa näkyvä liikahdus puolestaan soittaa jonkin sävelen viulun ääntä.

Kun ihmisiä nyt tulee kuvaan oikeasta laidasta, huoneessa alkaa soida rumpukomppi, jonka päälle jokainen liike kuvan vasemmassa laidassa alkaa säestää pianoa viulun avulla. Taideteos on valmis.

Huomioitavaa: Uudenaikaisilla pelikonsoleilla toimivat, pelaajan liikkeen tunnistavat urheilu- ja tanssipelit perustuvat pohjimmiltaan samankaltaiseen ajatteluun kuin yllä on esitetty: digitaalinen kamera välittää liikkeen tietokoneelle, joka analysoi, mitä sen edessä tapahtuu.

Kun ihmisiä saapuu huoneeseen, kone alkaa soittaa rumpukomppia.

Kysymyksiä ja vastauksia ohjelmoinnista:

ONKO SE VAIKEAA?

ONKO OHJELMOINTI VAIKEAA?

Ohjelmointi on ohjeiden antamista tietokoneelle – ja sen pohtimista, miten ohjeet kannattaa antaa. Kun kysytään, onko ohjelmointi vaikeaa, on kysymys hiukan samanlainen kuin se, onko kirjoittaminen vaikeaa.

Lapsikin oppii kirjoittamaan yksinkertaisia asioita. Toisaalta tullakseen kirjallisuuden Nobel-voittajaksi tulee tuntea laaja valikoima käytössä olevia kirjoittajan työkaluja ja niiden parhaita käyttötapoja. On osattava peilata omaa tekemistä olemassa olevaan ja toisaalta oltava tarpeeksi luova tehdäkseen omia oivalluksia.

Siis: Ohjelmointi ei välttämättä ole vaikeaa, mutta se tarjoaa loputtomasti haasteita. Ohjelmoinnissa ei ole koskaan mielestään täysin oppinut samalla tavalla kuin kirjailija ei ole koskaan mielestään täydellinen kirjoittaja.

MITÄ OHJELMOIJA TEKEE TYÖKSEEN?

Ohjelmoijia ja ohjelmointiin liittyviä työtehtäviä on hyvin erilaisia. Jotkut ohjelmoijat keskittyvät kirjoittamaan ohjelmakoodia ja miettimään, miten tietty sovellus tai ohjelmisto tai sen osa kannattaa järkevimmin toteuttaa. Toisena päivänä sama ihminen voi keskittyä testaamaan toisen tekemää ohjelmaa tai etsimään kirjoitetusta koodista ajatus- tai muita virheitä.

Joskus ohjelmoija keskittyy pohtimaan projektin pohjapiirrustusta: mikä on paras yhdistelmä erilaisia palikoita, jotka yhdessä muodostavat sujuvasti, nopeasti ja virheettä toimivan ohjelmiston.

Ohjelmointi on toisaalta ongelmanratkaisua, toisaalta taidetta.

Ohjelmointiin kuuluu myös paljon yhteistyötä: yhdessä pohtimista ennen koodauksen aloittamista, sen aikana sekä ohjelmiston testausvaiheessa.

Kun ohjelmoijan asiantuntijuus kasvaa, hänestä voi tulla esimerkiksi muista ohjelmoijista koostuvan tiimin vetäjä. Tällöin päätehtävä on auttaa muita suoriutumaan parhaansa mukaan, kysyä kriittisiä kysymyksiä ja tarjota pohdinta-apua pulmatilanteisiin.

KUKA PÄÄTTÄÄ, MILTÄ VALMIS OHJELMA NÄYTTÄÄ KÄYTTÄJÄLLE?

Tyypillinen tietokoneohjelma, esimerkiksi älypuhelimella käytettävä sovellus, jakaantuu eri osiin. Yksi ”osa” sovellusta on se, miltä se näyttää älypuhelimen ruudulla sitä käyttävälle ihmiselle. Tuota osaa kutsutaan käyttöliittymäksi.

Käyttöliittymällä tarkoitetaan esimerkiksi älypuhelimen Facebook-sovelluksessa kaikkea sitä, minkä käyttäjä näkee ja mitä hän voi sormillaan koskea tehdäkseen Facebookissa asioita.

Käyttöliittymän suunnittelu on yhden tai useamman käyttöliittymäsuunnittelijan tehtävä.

Käyttöliittymän suunnittelijan vastuulla on paitsi se, miltä valmis ohjelmisto näyttää käyttäjälle, myös se, kuinka nopeaa ohjelmiston käyttö on oppia ja kuinka tehokasta ja tarkoituksenmukaista käyttö on pidemmällä aikavälillä.

Käyttöliittymäsuunnittelija ei välttämättä ohjelmoi. Hänen tehtävänsä saattaa olla esimerkiksi puhua ohjelmiston tulevien käyttäjien kanssa ja piirtää – joko käsin tai tietokoneella – keskustelujen pohjalta ehdotuksia mahdollisiksi käyttöliittymämalleiksi.

MIKÄ TEKEE HYVÄN OHJELMOIJAN?

Hyviä ohjelmoijia on hyvin erilaisia. Kuitenkin ainakin seuraavat piirteet voidaan useimpien mielestä liittää hyvään ohjelmoijaan:

Halu ja kyky oppia uutta. Ohjelmointi on toisaalta käsityötä, toisaalta taidetta ja toisaalta ongelmanratkaisua. Kuten kaikissa noissa osa-alueissa, myös ohjelmoinnissa tulee sitä paremmaksi, mitä enemmän sitä tekee. Ihminen oppii esimerkiksi näkemään erilaisia tapoja ohjeistaa tietokonetta yhä tarkemmin, lyhyemmin ja elegantimmin.

Kunnianhimo. Hyvä ohjelmoija miettii aina, mi-

Monilla osa-alueilla voi pärjätä ohjelmoijana, vaikka ei olisi ollut luokan matikkanero.

ten tietyn koodinpätkän voisi kirjoittaa vieläkin paremmin. (Tässä ”paremmin” tarkoittaa samaa kuin yllä: tarkemmin, lyhyemmin, elegantimmin.)

Luovuus. Ohjelmoinnissa samaan lopputulokseen voi yleensä päätyä kymmenillä eri tavoilla, ja näistä tavoista useakin erilainen voi olla vieläpä täysin järkevä. Hyvä ohjelmoija ei jumitu toistamaan itseään vaan pohtii ongelmaa monilta eri suunnilta. Jos ongelma on vähänkin monimutkaisempi, kaksi huippuohjelmoijaakin ratkaisee sen todennäköisesti hiukan eri tavoilla.

Tarkkuus. Kun tietokoneelle antaa ohjeita, on otettava huomioon monia asioita oikeakielisyydestä siihen, että kone ei varmasti voi ymmärtää eri tavoin ihmisen tarkoitusperiä. Siksi ohjelmoijan on oltava tarkkana kuin porkkana. Tarkkuus kehittyy kuitenkin tekemisen kautta. Ohjelmoijaksi haluavan ei tarvitse olla valmiiksi erityisen pikkutarkka luonne.

PITÄÄKÖ OHJELMOIJAN OSATA MATEMATIIKKAA?

Ohjelmointitaito ja matemaattiset kyvyt liitetään usein toisiinsa. Joissain asiayhteyksissä matematiikasta onkin hyötyä, sillä ohjelmoija saattaa joutua kertomaan tietokoneelle, millainen käyrä sen on osattava piirtää tai ymmärrettävä, että tietty ongelma ratkeaa parhaiten hyödyntämällä vaikkapa jakojäännöksen käsitettä laskutoimituksissa.

On kuitenkin tärkeää ymmärtää, että ohjelmoinnissa oleellisempaa on yleinen kyky pohtia asioiden loogisia syitä ja seurauksia kuin laskennallinen matemaattinen osaaminen. Monilla osa-alueilla voi pärjätä ohjelmoijana, vaikka ei olisikaan ollut luokan matikkanero. Monille halu oppia ohjelmoimaan paremmin antaa kuitenkin lisämotivaatiota ymmärtää myös matematiikkaa paremmin.

MITÄ EROA ON ”OHJELMOINNILLA” JA ”KOODAAMISELLA”?

Ohjelmointi ja koodaaminen tarkoittavat käytännössä samaa asiaa eli tietokoneen ymmärtämän ohjelmakoodin kirjoittamista.

Tämä ei tarkoita, etteikö termeihin liittyisi sekaannusta:

Tvt, atk, it, ict, tietotekniikka, tietojenkäsittelyoppi, tietojenkäsittelytiede, digitaalinen lukutaito… Tietokoneisiin liittyvä sanasto on ongelmallista, sillä käytämme eri ilmaisuja sekaisin.

Myös ”tietotekniikkataidot” voivat nykyisin olla mitä tahansa lähtien hiiren käyttötaidosta ja taulukkolaskennasta aina siihen, että rakentaa itse tietokoneen, kirjoittaa ohjelman tai pystyttää lähiverkon. Samalla tavalla tietotekniikka koulussa voi tarkoittaa oppiainetta jossa opetetaan turvallista internetin käyttöä tai sitä, miten lähiverkko yhdistää printterit ja koneet toisiinsa.

MIKSI ON NIIN MONIA OHJELMOINTIKIELIÄ? MITEN NE EROAVAT TOISISTAAN?

Ohjelmoijat rakastavat työkaluja, ja samalla tavalla kuin puusepät tai arkkitehdit rakentavat omia erikoistuneita työkalujaan, ohjelmoijat voivat kirjoittaa uusia ohjelmointikieliä ratkaisemaan erilaisia ongelmia.

Jotkut kielistä antavat ammattilaiselle vapauden lähes kaikkeen (C), toiset ovat hyviä verkkosivustokäyttöön (JavaScript) ja kolmannet helppoja aloittaa (Ruby). Jotkut ovat hyviä matematiikassa (Haskell), toiset tilastotieteissä (R). Jotkut sopivat Applen älypuhelimille (Objective-C), toiset vaikkapa puhelinverkkojen toteuttamiseen (Erlang).

Peruskoulussa hyviä perusvarmoja ja monipuolisia aloituskieliä ovat esimerkiksi JavaScript, Ruby ja Python (ks. myös sivut 40–44 ja seuraava kohta.)

Peruskoulussa hyviä aloituskieliä ovat esimerkiksi JavaScript, Python ja Ruby.

MIKÄ OHJELMOINTIKIELI ALOITTELIJAN KANNATTAA VALITA?

Aloittelijan kannattaa valita kieli riippuen siitä, mitä haluaa saada aikaan. Tietokoneen työpöydällä, internetissä ja puhelimessa olevat sovellukset on saatettu koodata eri kielillä. Jos tietää mitä haluaa tehdä, kannattaa googlata mistä ideaasi muistuttava sovellus koostuu. Kaikki kielet muistuttavat kuitenkin perusperiaatteiltaan toisiaan, joten aloittaa voi melkein mistä vain.

Monille hyvä ensimmäinen ohjelmointikieli on sellainen, jolla saa nopeasti aikaan jotain näkyvää, eikä asentamiseen ja työympäristöön hallintaan tarvitse käyttää paljoa aikaa. Jos haluaa rakentaa verkkosivuston, riittää opetella HTML:ää ja

CSS:ää, jotka eivät ole edes varsinaisia ohjelmointikieliä (ks. sivu 122). Jos verkkosivustosta haluaa interaktiivisen, kannattaa opetella JavaScriptiä. Lisää vinkkejä ohjelmoinnin aloittamiseen löytyy tämän oppaan osiosta 3 alkaen sivulta 76.

VOIKO SAMALLA OHJELMOINTIKIELELLÄ OHJELMOIDA TIETOKONEELLE JA ÄLYPUHELIMELLE?

Riippuu tapauksesta, sillä eri älypuhelimien käyttöjärjestelmille ohjelmoidaan eri ohjelmointikielillä. Esimerkiksi Android-nimistä käyttöjärjestelmää käyttäville Samsungin tai HTC:n puhelimille tarkoitetut sovellukset ohjelmoidaan Java-nimisellä ohjelmointikielellä, jota käytetään myös muualla. Applen valmistamien iPhone-puhelimien iOS-käyttöjärjestelmälle taas ohjelmoidaan Objective-C:llä.

Eri älypuhelimille ohjelmoidaan eri ohjelmointikielillä.

”JOKA PÄIVÄ MIETIN, ETTÄ OLISIPA KIVA, KUN YMMÄRTÄISI PAREMMIN OHJELMOINTIA”

Taru Rastas, *viestintäneuvos, Liikenne- ja viestintäministeriö*

”Digitaalisuus tulee toimialasta riippumatta arkeen. Julkiset palvelut, yritysten liiketoimintaprosessit, sairaalat, kaupat... Maailma vain digitalisoituu, ja meidän pitää huolehtia siitä, että meillä on tekijöitä.

Jotta voimme vastata tarpeeseen, meidän pitää tarttua asiaan kaikilla ikäasteilla. Lapset on saatava digitaaliseen maailmaan koulusta lähtien siten, että he kykenevät ottamaan tekijän roolin.

Ja mihin ikinä se nuori haluaa tulevaisuudessa erikoistua – digitaalisuus tulee opinnoissa hyödyksi. Luonnontieteissä tai bisnesmaailmassa.

Liikenne- ja viestintäministeriö kehittää digitaalista Suomea. Me luomme mahdollisuuksia sille, että kuluttajilla ja yrityksillä olisi käytössään paras mahdollinen tieto- ja viestintätekniikka, ja siitä löydettäisi uutta kasvua Suomelle.

Usein kuullaan, että Nokia on romahtanut, ja tän alan insinööreille ei ole kysyntää. Mutta kyse on siitä, että IT:tä hyödyntävät alat kehittyvät valtavasti, ja sitä osaamista pitää koko ajan kehittää.

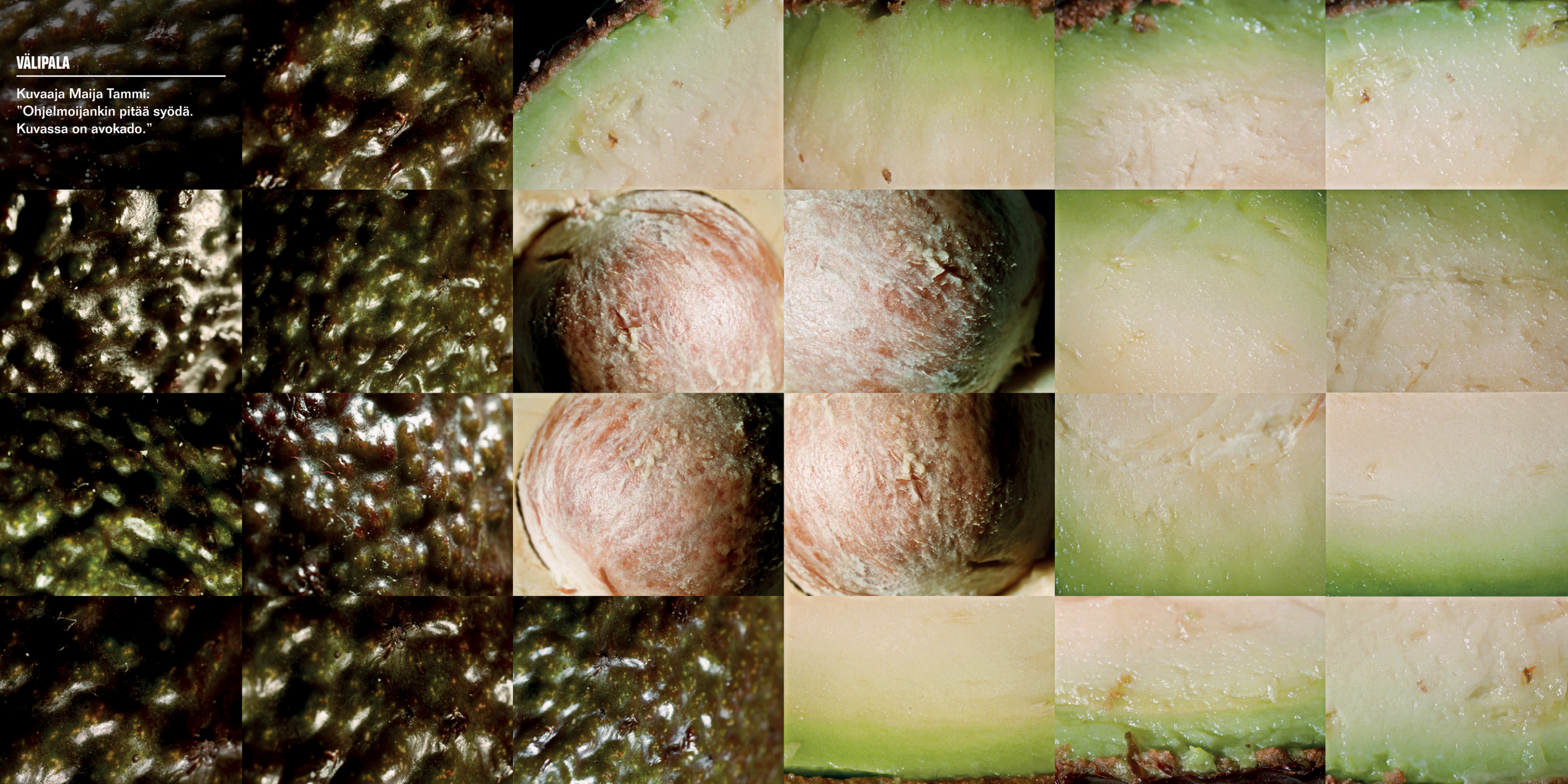
Usein uuden oppiminen vaatii poisoppimista siitä, että kun on aina tehty yhdellä tavalla, niin miksi pitäisi alkaa nyt tehdä asioita uudella tavalla.

Sen auttamiseksi pitäisi saada läpi ajatus siitä, että ohjelmointi ei ole erityistaitoa jollekin tietylle pienelle porukalle.

Joka päivä mietin itse, että olisipa kiva, kun ymmärtäisi paremmin jotain ohjelmointiin liittyvää. Ei ole tarkoitus, että jokaisesta meistä tulee hyvä koodari, mutta sen ymmärtäminen, että mihin kaikkeen sillä pystytään, ja millaisia asioita ratkotaan, ja miten se linkittyy erilaisiin toimintoihin – se on tärkeää.”

VÄLIPALA

Kuvaaja Maija Tammi:
”Ohjelmoijankin pitää syödä.
Kuvassa on avokado.”



ESIMERKKEJÄ KOODISTA

Tämän oppaan piirissä ei ole tarkoituksenmukaista tai edes mahdollista mennä siihen, miten varsinaista ohjelmakoodia kirjoitetaan. Eri ohjelmointikieliä on satoja, ja niiden oppimista varten löytyy lukemattomia oppaita ja verkkoresursseja.

Tämän osion tarkoituksena on kuitenkin vilauttaa maallikolle hiukan sitä, miltä kirjoitettu ohjelmakoodi esimerkiksi näyttää – ja miten eri kielet ja kielten tyypit karkeasti ottaen eroavat toisistaan esimerkiksi sen suhteen, millä tarkkuudella niillä operoidaan.

Seuraavat esimerkit havainnollistavat asiaa.

ESIMERKKI 0: VOILEIVÄN TEKEMINEN

Ohjelmoinnin tavoitteena on opettaa tietokonetta tekemään toistuvasti sellaisia asioita, joita ihminen ei halua tehdä joka kerta käsin erikseen. Viimeisen kahdenkymmenen vuoden aikana kieliä on kehitetty ja koodia on kirjoitettu ja niin paljon, että monet sellaiset tehtävät, jotka ennen veivät satoja tai jopa tuhansia rivejä koodia saa moderneissa kielissä ilmaistua muutamalla rivillä.

Palataan sivun 16 voileipäesimerkkiin:

Eri ohjelmointikielet vaativat eri tarkkuuden kuvailua siitä, mitä halutaan tehtäväksi.

Joissain kielissä tietokoneelle pitää selittää jokainen vaihe hyvin tarkasti, mutta lopputuloksena ohjelmoijalla on täydellinen vapaus luoda täysin sellaista jälkeä, kuin itse haluaa.

Toiset kielet lähtevät siitä, että ohjelmoijalle riittää hyvä perussuoritus. Sen vuoksi kielen kehittäjät ovat tehneet yleisiin ongelmiin tukun valmiita ratkaisuja, joita ohjelmoija voi käyttää hyväkseen. Tämä nopeuttaa koodin kirjoittamista ja ohjelmointiongelmien ratkaisemista.

Asiaa voisi verrata vaikkapa siihen, haluatko maustaa lihakeiton liemen kasvattamalla itse omat yrttisi ja mausteesi (yksityiskohtaisia ohjeita vaativat kielet) – vai riittääkö sinulle, että laitat keitetyn veden sekaan kaupasta ostetun lihaliemikuution (modernit helppokäyttöiset kielet).

Monille meistä jälkimmäinen riittää, mutta ammattikokki voi haluta täyden kontrollin.

Voileipäohjeiksi käännettynä muutama eritasoinen ohjelmointikieli voisi näyttää tältä ihmisen kielellä kirjoitettuna:

Valmiin toiminnallisuuden sisältävä kieli, esimerkiksi JavaScript tai Ruby

```
Tee voileipä.
```

Keskitason ohjeita vaativa kieli, esimerkiksi C

```
Kävele kaapille.
Ota kaapista leipäpussi.
Aseta leivät pussista pöydälle.
Voitele leipä.
```

Hyvin yksityiskohtaiset ohjeet vaativa kieli, esimerkiksi konekieli Assembly

```
Ota askelia kohtisuoraan, kunnes
saavut keittiön kaapille.
Kun saavut kädenmitan päähän
kaapista, pysähdy.
Nosta vasen kätesi.
Avaa kaapiston vasen ovi tasan 90
asteen kulmaan kaapin alareunaan
nähden.
Käyttäen oikeaa kättäsi tartu
ylimmällä hyllyllä olevaan
leipäpussiin.
Siirrä pussi pöydälle.
Jos pussissa on suljin, poista se.
Tartu pussissa olevaan ensimmäiseen
leipään.
Ota leipä pussista ulos.
Laske leipä pöydälle.
...
```

ESIMERKKI 1: HELLO WORLD!

Ohjelmointipiireissä on 1970-luvulta lähtien yleistynyt sympaattinen tapa: kun uutta kieltä lähdetään opettamaan, perinteisesti ensimmäinen asia on opetella, miten kielen avulla tulostetaan tietokoneen näytölle sanat ”Hello World!”.

Hello World! -fraasin tulostaminen tapahtuu useimmissa kielissä yksinkertaisilla koodiriveillä, jotta aloittelija voi opastettuna seurata ohjelman toimintaa ja ymmärtää ohjelmointikielen perusteita.

Alla on esitetty neljä esimerkkiä siitä, miten Hello World! -sanat tulostuvat näytölle eri ohjelmointikielillä.

Lukijan ei tässä tarvitse ymmärtää, miten kielet tarkemmin ottaen toimivat. Esimerkkien tarkoitus on esitellä, miten erilaiselta sama tehtävä voi näyttää eri kielillä.

Huomaa, että alla läpikäydyt neljä ohjelmointikieltä ovat samat kuin yllä mainitussa voileipäesimerkissä ja etenevät yksinkertaisesta monimutkaisempaan suuntaan.

Ruby – ei näytä kovin vaikealta

```
puts "Hello World!"
```

JavaScript – mukiinmenevää tämäkin

```
console.log("Hello World!")
```

C-kieli – vaatii jo perehtymistä

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
 printf("Hello World!\n");
 return EXIT_SUCCESS;
}
```

8086 Assembly – monimutkaista (eikä juuri enää käytössä)

```
DOSSEG
.MODEL TINY
.DATA
TXT DB "Hello World!$"
.CODE
START:
       MOV ax, @DATA
       MOV ds, ax

       MOV ah, 09h          ; prepare output function
       MOV dx, OFFSET TXT   ; set offset
       INT 21h              ; output string TXT

       MOV AX, 4C00h        ; go back to DOS
       INT 21h
END START
```


ESIMERKKI 2: NUMERONARVAUSPELI

Eri ohjelmointikieliä kirjoitetaan eri merkinnöin, ja samoja asioita ilmaistaan hiukan eri näkökulmista.

Alla olevalla koodilla saa luotua pienen pelin, jossa tietokone arpoo ensin jonkin "ajattelemansa" numeron, jonka jälkeen käyttäjä koettaa arvata tuon numeron.

Monissa ohjelmointikielissä on esimerkiksi valmiiksi kirjoitettuja toimintoja (näitä kutsutaan funktioiksi), jotka valitsevat automaattisesti satunnaisen numeron ja muuttavat sen kokonaisluvuksi. Jotta ohjelmointia tuntematon lukija voisi seurata alla olevia esimerkkejä edes päällisin puolin, koodipätkiä on selitetty auki tavallisella kielellä – käytäntö, jota myös koodarit harrastavat!

Älä missään nimessä ahdistu, jos alla oleva näyttää vaikealta. Ohjelmointia opetellaan askeleittain, ja vaikka alla olevat esimerkit ovat suhteellisen yksinkertaisia, niitä ei käytännössä voi täysin ymmärtää ilman kyseisen kielten perusteiden opettelua.

Coffee Script

```
num = Math.ceil(Math.random() * 10)
guess = prompt "Arvaa mitä lukua välillä 1-10 ajattelen!"
while parseInt(guess) isnt num
 guess = prompt "Hävisit! Arvaa uudestaan."
alert "Oikein arvattu!"
```

Aluksi määritellään funktio nimeltä "num". Se valitsee satunnaisen luvun väliltä 1–10. Apuna tässä käy-

tetään CoffeeScript-kielen valmiita työkaluja (”Math.ceil”- ja ”Math.random”-funktiot). Valittu luku on siis pelin ”oikea vastaus”, joka käyttäjän pitää arvata.

Seuraavaksi määritellään ”guess”-niminen funktio. Se näyttää käyttäjän näytöllä viesti-ikkunan, joka tiedustelee käyttäjää arvaamaan, mitä numeroa välillä 1–10 tietokone ”ajattelee”.

Sitten todetaan, että jos käyttäjän syöttämä arvattu numero ei vastaa ensimmäisen rivin satunnaisesti valittua, tietokoneen ”ajattelemaa” numeroa, näytetään käyttäjälle rivillä neljä oleva viesti. Sen mukaan käyttäjä ei arvannut oikein.

Jos käyttäjä arvaa oikein – eli syötetty luku on sama kuin ensimmäisellä rivillä oleva num – näytetään käyttäjälle viesti, jonka mukaan hän on arvannut oikein.

PYTHON

```
import random

target, guess = random.randint(1, 10), 0
while target != guess:
guess = int(input('Arvaa ajattelemani luku välillä 1–10 kunnes saat sen
oikein: '))
   print('Oikein arvattu!')
```

Tässä ”random.randint” on Python-kieleen sisäänrakennettu funktio, joka valitsee sattumanvaraisen numeron yhden ja kymmenen välillä. Nolla lopussa tekee luvusta kokonaisluvun (kuten sanottua, älä huoli, jos esimerkit eivät tunnu ymmärrettäviltä ilman perehtymistä kielen logiikkaan!).

Koodissa todetaan, että niin kauan kuin käyttäjän arvaus ”guess” ja tietokoneen ajattelema luku ”target” eivät ole yhtäsuuria, käyttäjää pyydetään arvaamaan yhä uutta numeroa.

Seuraavaksi esitetään viesti-ikkuna, joka pyytää käyttäjää syöttämään luvun välillä 1–10, kunnes hän arvaa tietokoneen ”ajatuksen” oikein.

Kun ”guess” ja ”target” ovat lopulta samat, tietokone kertoo, että käyttäjä on arvannut oikein.

”DIGITALISAATIO EI TARKOITA, ETTÄ KÄDENTAIDOT VÄHENEVÄT”

Tapio Tuomi, *asiantuntija, Sitra*

”Kun maailma muuttuu, ja jos Suomi haluaa pärjätä, niin digitalisaatio on homman juoni. Siihen tarvitaan erilaisia taitoja kuin nykyään. Paras tapa varmistaa menestys on se, että uusia taitoja opetetaan peruskoulussa kaikille suomalaisille.

Kyse on tasa-arvosta. Kun ohjelmointi laitetaan peruskouluun, se antaa kaikille samat mahdollisuudet jatkaa eteenpäin.

Toinen tärkeä asia on kädentaitojen uudistaminen. Villamyssyä voidaan aina virkata käsityötunnilla, mutta myös uusia taitoja täytyy olla. Koodatessa tekee itse jotain, ja syntyy jotain uutta.

Minä jauhan esimerkiksi itse kotona jyvistä jauhoni. Ymmärrys siitä, mistä asiat rakentuvat, on ohjelmoinninkin osalta hyödyllistä.

Digitalisaatio ei tarkoita sitä, että kädentaidot vähenevät, ne vaan muuttavat muotoaan. Ohjelmointikyky on työkalu, jonka avulla rakennetaan asioita. Voidaan myös ajatella niin, että halutaan pitää huolta Suomen ja suomalaisen koulusysteemin maineesta.

Muut maat saavat koko ajan Suomea kiinni ja menevät ohi. On hyvä päivittää ja uusia koulun sisältöjä – kuten nyt vaikka sitä kautta, että ilmeisesti nyt jakokulma on jäämässä matematiikasta pois.

Peruskoulutusosastolla tajutaan itse muuttaa asioita oikeaan suuntaan, se on minusta hienoa.

Pysähtyminen pitkäksi aikaa on pahasta. Tärkeämpää on mennä johonkin suuntaan. Liian valmiiksi sitä ei voi suunnitella, muuten ei tehtäisi yhtään mitään. Sitten jos tulee vastaan ongelmia, ratkaistaan ne sitten.”

Tässä osiossa kerrotaan, miten ohjelmointi näkyy uudessa opetussuunnitelmassa ja miksi muutos on tärkeä. Osio pyrkii vastaamaan opettajan huoliin ja kertoo myös, miten muut maat ovat lähestyneet ohjelmoinnin opettamista.

2 | *Miten OPS muuttuu – ja miksi?*

TÄSTÄ ON KYSE OPSIN UUDISTUKSESSA

Tiedossa on, että syksyllä 2016 käyttöön tulee uusi opetussuunnitelma ja se sisältää ohjelmointia.

Millaisia asioita koulujen on ajateltu käytännössä opettavan?

Tätä kirjoittaessa viitekehyksenä on vuoden 2016 opetussuunnitelman luonnos. Se on tätä opasta laadittaessa kouluilla kommenttikierroksella. Muutoksia voi tulla, mutta perusajatus on selkeä, ainakin jos kysyy Opetushallituksen matematiikan opetussuunnitelmatyöryhmän puheenjohtajalta **Leo Pahkinilta**:

”Pääasia on halu tuoda esille, mistä ohjelmoinnissa on kysymys. Ja innostaa nuoria huomaamaan, että ohjelmointi on jännittävää. Parhaimmillaan kouluille olisi hyvä saada myös pöhinää, jonka myötä oppilaille järjestettäisiin erilaisia kerhoja ja valinnaisopintoja.”

MITÄ ERI LUOKILLA OPISKELLAAN?

Muutokset eri luokka-asteilla menevät näin:

Luokilla 1–2 opetellaan antamaan yksikäsitteisiä komentoja ihmiseltä toiselle. Ohjelmointi on käskyjen antamista tietokoneelle, ja tässä pohjustetaan sitä.

”Opetellaan antamaan tarkkoja ohjeita, esimerkiksi ota ‘kolme askelta eteenpäin’ – ei ‘ota kolme askelta’, jotka voisivat olla sivulle tai taakse”, Pahkin sanoo.

”Tässä opitaan sitä, että täsmälliset ohjeet tuottavat täsmällistä toimintaa, ja epämääräiset ohjeet tuottavat epämääräistä toimintaa.”

Luokilla 3–6 aletaan perehtyä tekemiseen, joka on lähempänä ohjelmointia. Työkaluna ei vielä ole varsinainen ohjelmointikieli vaan jokin visuaalinen ohjelmointiympäristö, jossa työskennellään käytännössä hiiren avulla, ei kirjoittamalla.

Luokilla 3–6 perehdytään visuaaliseen ohjelmointiympäristöön.

”Tässä kohden vaihdetaan ihminen koneeseen. Silloin välillä pitää olla jokin kieli – näillä luokilla käytännössä jokin graafinen ohjelmointikieli”, Pahkin selittää.

”Ohjelmointia voidaan harjoitella raahaamalla tai siirtelemällä asioita. Löydetään niitä erilaisia komentoja, joita on esimerkiksi [MIT:ssä kehitetyssä lasten ohjelmointiympäristössä] Scratchissakin vain muutama. Sen jälkeen ruvetaan näillä muutamilla komennoilla löytämään esimerkiksi toistoja. Kaikkia näitä voidaan nopeasti oppia.”

Luokilla 7–9 aloitetaan perehtyminen johonkin oikeaan ohjelmointikieleen. ”Emme ota kantaa siihen mikä se kieli on”, Pahkin sanoo. ”Mutta tarkoituksena on, että ymmärretään perusasioita kielestä ja osataan tulkita ohjelmakoodia – tuossa kohdin ohjelma ottaa jonkin luvun sisään, tuossa se tekee sille jotain ja niin edelleen.”

ALOITTAVATKO KAIKKI LUOKAT OHJELMOINNIN SYKSYLLÄ 2016?

Tavoite on, että syksyllä 2016 kullakin alakoulun luokalla ykkösestä kutoseen aletaan harjoittelemaan ohjelmointia. Samalla edessä on muutaman vuoden siirtymäaika, jolloin esimerkiksi kolmosluokkalainen aloittaa ohjelmoinnin parissa, vaikka hänellä ei ole pohjalla luokkien 1–2 oppeja.

”Luokilla joudutaan tässä suhteessa hiukan improvisoimaan, koska oppilaan polku ei ole edennyt ekalta luokalta lähtien. Se voi olla aluksi haasteellista, muttei mahdotonta”, OPH:n Leo Pahkin sanoo.

Nykysuunnitelman mukaan luokat 7–9 siirtyvät ohjelmoinnin pariin vuosi kerrallaan: syksystä 2017 alkaen ohjelmointia opiskellaan luokilla 1–7, syksystä 2018 alkaen luokilla 1–8 ja 2019 alkaen luokilla 1–9.

MISTÄ OPETUKSELLE SAADAAN AIKAA?

OPH:n suunnitelmassa aika ohjelmoinnille otetaan matematiikan tuntijaosta. Tarkoitus on kuitenkin, että ohjelmointiajattelua linkitettäisiin myös muihin oppiaineisiin mahdollisuuksien mukaan. Ajatuksia siitä, millaisia ohjelmointiin liittyviä asioita voi nostaa esiin eri oppiaineiden tunneilla on esitetty sivuilla 124–126.

KUKA SUOMESSA OPETTAA OHJELMOINTIA?

Suomessa ohjelmoinnin opetusvastuu on vuonna 2016 alakoulussa peruskoulun opettajilla ja yläkoulussa todennäköisesti matematiikan opettajilla. Tietotekniikan aineopettajia on melko vähän, ja heitä valmistuu Jyväskylän, Joensuun ja Turun yliopistoista.

Helsingin yliopisto lakkautti tietotekniikan aineenopettajankoulutuslinjansa keväällä 2005. Uusia opettajia koulutettaessa uudet vastuualueet olisi syytä ottaa huomioon. Varmaa on myös, että nykyiset luokanopettajat tulevat tarvitsemaan täydennyskoulutusta.

Toisaalta se, että monenlaiset ihmiset alkavat opettaa ohjelmointia on myös iso mahdollisuus: kun ohjelmointi ei ole oma oppiaineensa eikä muodollisia pätevyysvaatimuksia sen opettamiselle ole, syntyy erilaista osaamista, innostuneisuutta ja kokeilemisen kulttuuria.

Ohjelmointi on klassisesti ollut hyvin rajatun ihmisjoukon harrastus. Nyt sillä on mahdollisuus demokratisoitua todella.

Ohjelmointi on klassisesti rajatun joukon harrastus. Nyt sillä on mahdollisuus demokratisoitua.

TIETOKULMA

NIHKEÄ NYKYTILA: YHÄ HARVEMPI VALITSEE TIETOTEKNIIKAN

Tietotekniikkaa on opetettu Suomen kouluissa 1960-luvulta saakka. Yllättäen tietotekniikan suosio valinnaisena aineena on laskenut viime vuosina.

Tietotekniikan opettaminen alkoi Suomessa 1960-luvulla. Osa vanhemmista muistaa 1980-luvulta ohjelmointiopetuksen, jossa kielinä saatettiin käyttä Logoa, Pascalia tai C:tä. 1990-luvun alussa tietotekniikka menetti asemansa erillisenä opetettavana aiheena. Opetussuunitelmissa puhuttiin esimerkiksi kymmensormijärjestelmistä sekä tiedon hankinta- ja prosessointitaidoista.

Tietotekniikan opetusta on muokattu viimeksi vuonna 2004. Opetussuunnitelma painotti tuolloin entistä enemmän oppiaineiden välistä yhteistyötä ja integraatiota.

Suomessa on nyt voimassa tietotekniikan opetussuunnitelma, johon kuuluvat osa-alueet ovat "tietotekniikan perusteet", "laitteen käyttö ja tiedon hallinta", ”tekstinkäsittely", ”taulukkolaskenta", ”tietokannat", ”grafiikka" sekä ”internet". Käytännössä osa-alueet on sulautettu osaksi muuta opetusta.

Muun muassa Matemaattisten aineiden opettajien liitto MAOL ry on laatinut ohjeistuksen otsikolla "Mitä peruskoulun päättävän oppilaan tulisi tietää tietotekniikasta", ja opetussuunnitelmissa määritellään eri tavoitetasot oppilaan taidoille.

Samaan aikaan kun tieto- ja viestintätekniikan taidot ovat yhä tärkeämpiä, tietotekniikan valinnaisena valinneiden oppilaiden määrä vähentyi 10 prosenttia välillä 2008–2010.

Lisäksi sukupuolten väliset erot ovat vahvoja: esimerkiksi tietotekniikan vähintään 95 tuntia vuodessa kestäviä opintoja vuosiluokilla 7–9 opiskelevista vain neljäsosa oli tyttöjä.

10 SYYTÄ SISÄLLYTTÄÄ OHJELMOINTI OPSIIN

Vaikka ohjelmointia pitäisi hyödyllisenä taitona, voi silti herätä kysymys, miksi sitä pitää opettaa jo peruskoulussa. Seuraavassa on esitetty 10 hyvää syytä.

1. OPPILAILLA ON OIKEUS OPPIA OHJELMOINTIA

Maailmaan rakentavat uutta yhä useammin ihmiset, jotka osaavat käskeä tietokonetta.

Oli kyse sitten tieteestä, taiteesta, terveydenhuollosta, yhteiskunnan tietojärjestelmistä, ohjelmistoista liiketoiminnan kehittämisessä tai ylläpidossa, älypuhelimista, maailmanlaajuisesta tietoverkosta, sähköisestä viestinnästä tai sosiaalisesta mediasta – taustalla pyörii yhä useammin ohjelmakoodi.

Lapset ja nuoret tarvitsevat ymmärrystä nykyaikaisen maailman rakennuspalikoista.

Jos emme tarjoa oppilaille mahdollisuutta ymmärtää ympäröivää maailmaa, he jäävät ikävään väliinputoajan asemaan tietoyhteiskunnassa.

Perusasioiden ymmärtäminen ohjelmoinnista on verrattavissa siihen, että ymmärtää perusasiat biologiasta, maantiedosta, äidinkielestä tai kuvaamataidosta: elämässä tulee nykyään ohjelmistoja vastaan vähintään yhtä usein kuin tulee tarve erottaa rauduskoivu hieskoivusta tai pakottava mielihalu maalata akryyliväreillä.

Ohjelmointia voidaan soveltaa myös halki rajojen eri oppiaineiden eri osa-alueilla (sivuilla 124–126 on pohdittu, miten ohjelmointia esimerkiksi voidaan ottaa esille eri oppiaineissa).

2. OPPILAAT HALUAVAT LISÄÄ TVT:TÄ

Helsinkiläisten peruskoulujen oppilaat, lukiolaiset ja Stadin ammattiopiston opiskelijat esittivät huhtikuussa 2014 julkilausuman toiveistaan opetuksen kehittämisessä. Siinä lapset ja nuoret toivoivat, että tieto- ja viestintäteknologiaa hyödynnettäisiin opetuksessa nykyistä enemmän.

Erityisesti peruskoulujen oppilaat toivoivat lisää tieto- ja viestintäteknologiaa opiskeluun. He peräänkuuluttivat myös mielenkiintoisia ja monipuolisia opetusmenetelmiä sekä enemmän taitoaineita, jollaiseksi myös ohjelmointi voidaan laskea.

Julkilausumaa oli laatimassa yli 5 000 oppilasta. Lisätietoja saa osoitteesta www.hel.fi/www/uutiset/fi/helsinki/oppilaiden-julkilausuma.

Maailmaan rakentavat uutta yhä useammin ihmiset, jotka osaavat käskeä tietokonetta.

3. SUOMI JANOAA IT-OSAAJIA

”Kauhea pula on osaajista. Päättäjillä ja jopa meidän alalla toimivilla isoilla johtajilla on väärä käsitys asiasta.”

Näin sanoo esimerkiksi it-alan asiantuntijatalon Reaktorin teknologiajohtaja **Hannu Terävä**, joka on vetänyt firman rekrytointia vuosia.

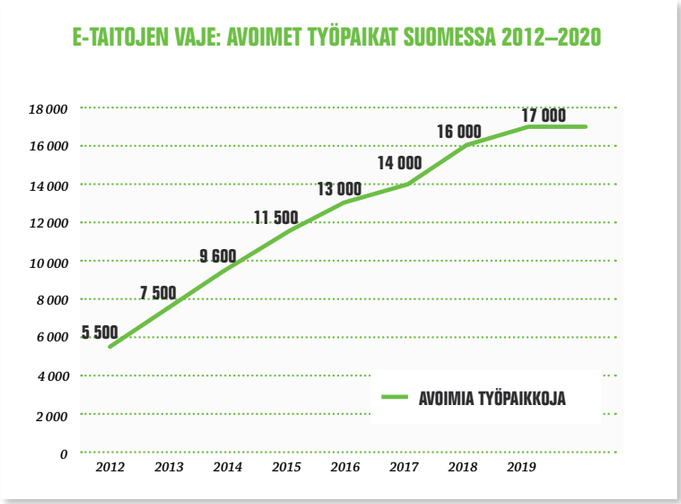
Samaa mieltä on Futuricen **Yrjö Kari-Koskinen**:

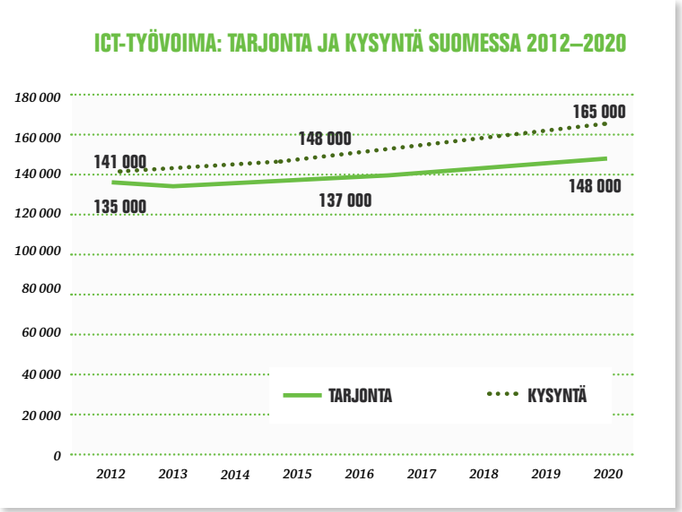
”Rekrytointi on koko ajan haastavaa. Kun yritys menestyy ja kasvaa, niin haasteena on, että saa kasvua tukemaan riittävästi osaavia tyyppejä.”

Peruskoulun opetussisältöjä ei voi lähteä muuttamaan yksittäisten yritysten osaamistarpeiden perustella. Silti myös alan puolueettomat selvitykset kertovat, että pula it-alan osaajista on kansainvälinen trendi.

Esimerkiksi maailman johtava liikkeenjohdon konsulttitalo McKinsey arvioi, että Yhdysvalloissa tulee olemaan nykyisillä työllistymis- ja koulutusmäärillä vuoden 2018 loppuun mennessä peräti 140 000–190 000 analytiikan syväosaajan vaje.

Kansantalouden analyysejä tuottavan Gartnerin mukaan pelkästään niin kutsutun *big datan* eli erilaisten suurten tietovarantojen käsittelyyn ja hyödyntämiseen tarvitaan 4,4 miljoonaa työntekijää maailmanlaajuisesti jo vuonna 2015, mutta vain kolmasosaan näistä paikoista löydetään osaaja.





Mikä sitten on tulevaisuus Suomen osalta? Vasemmalla esitetyt kaksi Empirican arvioihin (2013) pohjaavaa kuvaajaa kertovat samaa tarinaa:

Vuoteen 2020 meikäläisittäin on peräti 17 000 työntekijän vaje ict-alan työpaikoissa.

Siis: panostamalla ohjelmointiosaamiseen Suomi sekä paikkaa alan työntekijävajetta että antaa nykyperuskoululaisille osaamista, jolla saa tulevaisuudessa töitä.

4. JOS EMME SKARPPAA, MUUT MAAT MENEVÄT EDELLE

Suomalainen koulutus on maailman huippua. Sijoitumme Pisassa yhä loistavasti, ja muiden maiden edustajat käyvät Suomessa tutustumassa, kuinka fiksusti ja joustavasti olemme järjestäneet peruskouluopetuksemme.

Ei ole syytä antaa tilanteen lipsua ohjelmoinnin opettamisen osalta.

Sivuilla 62–66 on esitetty, miten muissa maissa on jo aloitettu ohjelmointiosaamisen kehittäminen. Lähin esimerkki löytyy Viron kouluista. Myös Britanniassa ohjelmointia opiskellaan syksystä 2014 alkaen peruskoulussa.

Jos Suomi ei ryhdy välittömiin toimiin lasten ja nuorten ohjelmointitaitojen kehittämisessä, muut maat menevät tulevaisuuden kannalta keskeisissä taidoissa ohi.

5. OHJELMOIMALLA VOI PELASTAA KANSANTALOUDEN

Yksi keskeisimpiä kansantalouden mittareita on tuottavuus: kuinka tehokkaasti tuotamme tietyn panoksen. Tuottavuutta voi parantaa tekemällä samassa aikayksikössä enemmän työtä tai tekemällä saman työn nopeammin.

Ohjelmistoihin perustuvat tuotteet kuten tietokonepelit, älypuhelinsovellukset tai vaikkapa digitaalisessa muodossa jaeltavat musiikkikappaleet ovat kaikki omiaan lisäämään tuottavuutta, sillä niitä voidaan monistaa rajattomasti.

Kun vaikkapa Marimekon suunnittelija suunnittelee uuden tyylikkään paitakuosin, varsinaista vaatetta on silti valmistettava tehtaassa kaikille markkinoille. Vaate on fyysinen asia, jonka monistaminen maksaa. Vaatetta on myös fyysisesti kuljetettava ympäri maailmaa lentokonein, laivoin ja rekoin, mikä lisää entisestään kustannuksia.

Kun taas suomalainen pelitalo Supercell kehittää tabletilla pelattavan pelin, sekä tuotteen monistaminen että jakeleminen ympäri maailmaa on käytännössä lähes ilmaista. (Käytännössä esimerkiksi pelin jakelija kuten Applen verkkokauppa ottaa myynnistä provision, mutta tätä tapahtuu muidenkin tuotteiden jälleenmyyjien kohdalla.)

Juuri tämä on selitys sille, miksi Supercell teki

vuonna 2013 tulosta peräti *kolme miljoonaa euroa* jokaista yrityksen noin sataa työntekijää kohden.

Kun tuotteen monistaminen ja jakelu on ilmaista ja tuotteesta tulee maailmanlaajuisesti suosittu, sen tuottomahdollisuudet ovat rajattomat. Tällaisissa yhteyksissä tuottavuuden lisäyksessä ei puhuta yksittäisistä prosenteista vaan jopa sadoista prosenteista.

Tämän takia ohjelmistopohjaiset tuotteet ja palvelut luovat ratkaisevaa kilpailuetua niille kansantalouksille, joiden ihmiset osaavat ohjelmoida.

6. KOODAUSTA EI VOI ULKOISTAA NIIN HELPOSTI ULKOMAILLE KUIN LUULLAAN

Usein kuulee puhuttavan, että koodaaminen siirtyy vähitellen Kiinaan tai Intiaan. Kokemus on kuitenkin osoittanut, että koodaamisen ulkoistaminen halvan työvoiman maihin ei ole niin helppoa, kuin vielä muutama vuosi sitten ajateltiin.

Ohjelmistojenkehitysprosessit ovat monimutkaisia hankkeita. Ne vaativat usein jatkuvaa kasvokkain tapahtuvaa kommunikaatiota sekä hanketta toteuttavan tiimin sisällä että asiakkaan kanssa. Osaamisen on oltava parasta laatua, sillä pienikin puute koodarin tai ohjelmistoarkkitehtuurin suunnittelijan taidoissa voi johtaa katastrofaalisiin seurauksiin ja projektin valmistumisen viivästymiseen.

Projekti ei myöskään välttämättä ole lähes koskaan ”valmis”, joten yhteistyön sujuvuus korostuu kehityksen jatkuessa vuosien kuluessa.

Usein suuret ohjelmistohankkeet ovat myös yritysten liiketoiminnan kannalta niin merkittäviä, että ohjelmoijien palkkakustannusten erot esimerkiksi Suomen ja Intian välillä eivät näyttele kokonaisuudessa ratkaisevaa roolia. Kun hankkeen vaikutukset mitataan sadoissa miljoonissa euroissa, ja kokonaisuuden on yksinkertaisesti pakko onnistua, kynnyskysymykseksi ei muodostu se, maksetaanko toteuttajalle esimerkiksi 0,5 miljoonaa euroa vai 5 miljoonaa euroa.

7. OHJELMOINTIA ON TÄRKEÄÄ YMMÄRTÄÄ, VAIKKEI KOODAISI KOSKAAN ITSE

Miksi pitäisi opetella ohjelmointia, jos tietää, ettei aio koskaan ruveta koodariksi?

Ohjelmistot tulevat olemaan läsnä yhä useammilla toimialoilla tieteestä taiteeseen ja metsäkoneista vaateteollisuuteen. Monet yritysten henkilöstöasiantuntijat sanovat, että hyvät it-taidot tulevat olemaan tulevaisuudessa huomattavasti useamman laadukkaan työpaikan saamisen edellytys – samalla tavalla kuin englannin kielen taito on nykyään täysin keskeistä ja lähtökohtainen oletus rekrytoinneissa.

Ihminen joka ymmärtää perusasiat ohjelmoinnista voi myös viestiä sujuvasti koodaavien kollegoidensa kanssa. Ja kun edessä on ohjelmistopäivityksen tai tietojärjestelmän ostoprosessi omaan organisaatioon, hän osaa arvioida tarjoajan esittämiä työmääriä eikä tule huijatuksi.

Peruskoululainen opettelee biologiaa vaikkei aikoisi biologiksi ja maantietoa vaikkei aikoisi geologiksi. Myös ohjelmoinnin ymmärtäminen on yleissivistystä, eikä koodaamisen opettelu edellytä urasuunnitelmia ohjelmoijana.

8. OHJELMOINTI ON MOTIVOIVAA JA OPETTAA AJATTELUA

Oppilas saa ohjelmoinnista suoraan ainakin kahdenlaista iloa. Toisaalta oppilas saa tietokoneelta suoran palautteen siitä, kun hän saa ohjelmistonsa toimimaan tai jotain syttymään eloon näytöllä. Tuotoksen voi esitellä heti kavereille ja usein jakaa myös verkossa.

Toisaalta ohjelmoinnista saa myös älyllistä iloa joka muistuttaa sitä, kun matematiikassa saa sievennettyä monimutkaisen yhtälön tai englannissa löytää juuri oikean ilmaisun.

Ohjelmoinnin avulla oppii myös yleisesti hyödyllisiä kognitiivisia taitoja. Looginen ja luova ajattelu, tarkka työskentely, kyky hahmottaa ongelma ja muodostaa sille erilaisia ratkaisuvaihtoehtoja sekä visualisoida ja käsitteellistää ne ovat hyödyllisiä kaikissa aineissa ja elämässä itsessään.

Parhaimmillaan ohjelmointi myös innostaa yläkouluikäisen opiskelemaan lisää muita asioita – diskreettiä matematiikkaa 3d-peliohjelmointia varten tai dna-visualisointeja biologiasta innostuneelle.

9. OHJELMOINNIN OPETTAMINEN KAIKILLE SAA TYTÖT MUKAAN

Suomalaisessa it-talossa ohjelmoijan paikkoja hakevien naisten määrä on tyypillisesti korkeintaan muutaman prosentin. Esimerkiksi tamperelainen Vincit-ohjelmistotalo valittiin vuonna 2014 kokoluokassaan Suomen parhaaksi työpaikaksi, mutta siitä huolimatta ”työnhakijoista naisia on alle prosentti”, kertoo **Pasi Kovanen** yrityksestä.

Suomessa it-alalla on naisia vain noin 23 prosenttia – ja lukuun on laskettu mukaan kaikki

Suomalaisessa it-talossa ohjelmoijan paikkoja hakevista on naisia vain muutama prosentti.

it-alan yrityksissä työskentelevät ihmiset erilaisia tukitoimintoja myöten. Tilanne ei ole menossa itsestään tasa-arvoisempaan suuntaan: esimerkiksi Aalto-yliopistossa syksyllä 2013 tietotekniikan koulutusohjelmassa aloittaineista vain 4 prosenttia oli naisia.

Ongelma ei ole tyttöjen taidoissa. Teknologiateollisuus ry:n *Naisia ICT-alalle!* -tutkimuksessa todetaan, että jo alakouluikäiset tytöt ovat suhteessa taitavia tieto- ja viestintätekniikan käyttäjiä.

Samasta tutkimuksesta on nähtävissä, että lukioikäisillä tytöillä ei ole selkeää mielikuvaa it-alasta, siihen liittyvistä työtehtävistä ja vaatimuksista. Mielikuvissa it-alan töissä ”istutaan vain jossain koneen ääressä kaikki päivät”.

Myönteistä on, että kuluvan kevään aikana erilaisissa yrityksissä järjestetyt lasten Koodikoulut (ks. esim. haastattelut sivuilla 67 ja 139) ovat houkutelleet runsaasti myös tyttöjä.

Ohjelmointia on tärkeää alkaa opettaa peruskoulussa, jotta saamme mukaan kokonaisen puolikkaan ikäluokasta, joka on toistaiseksi syyttä aliedustettuna.

Suomella ei ole varaa pitää ohjelmointitaitoja miesten etuoikeutena.

”Vahvat pedagogiset taidot, innostunut, kannustava…” Ideaali kuulostaa aika paljon suomalaiselta opettajalta.

10. SUOMESSA ON LOISTAVAT OPETTAJAT MUUTOKSEEN

Millainen on ideaali luokanopettaja, joka syksyllä 2016 aloittaa alakouluikäisten kanssa ohjelmoinnin?

Hänellä on vahvat pedagogiset taidot ja käsitys siitä, mikä luokkahuoneessa toimii ja mistä lapset innostuvat.

Hän jakaa tietoa ja materiaalia muille opettajille ja mentoroi mielellään epävarmempia.

Hän osaa kannustaa erilaisia oppijoita ja osoittaa uranäkymiä. Hän pystyy ja haluaa kehittää omaa osaamistaan.

Ja ennen kaikkea: hän on innostunut ja peloton.

Ei kuulosta kovin erilaiselta verrattuna tyypilliseen suomalaiseen opettajaan.

Me pystymme tähän.

MIKSI TÄMÄ ON TÄRKEÄÄ? **KOODI2016:N TUKIJAT KERTOVAT**

”OHJELMOINTI ON OSA ELÄMÄÄ”

Peter Vesterbacka, *Mighty Eagle, Rovio*

”On tärkeää, että koulutus seuraa yhteiskunnan kehitystä. Pointti ei ole, että kaikista pitää tulla ohjelmoijia, mutta nykyään maailma rakentuu siten, että ilman ohjelmointiymmärrystä on vaikea hahmottaa, miten maailma toimii.

Ohjelmoinnin ymmärtäminen on perusoikeus. On katastrofi, jos sitä ei ymmärretä.

Samalla on tärkeää, että ohjelmointia demystifioidaan. Se on yhtä tärkeää kuin kielten tai historian opiskelu – yleissivistystä.

Ohjelmointia aletaan nyt opettaa osana matematiikan tuntijakoa, mutta yhtä lailla se voisi olla osa musiikkia tai kuvaamataitoa. Sitä ei pidä niputtaa pelkkään matematiikkaan.

Joskus oli atk-luokkia, ja ne nähtiin erillisenä saarekkeena, mutta ohjelmointi ei ole sitä. Se on osa elämää.

Jos esimerkiksi eduskunnassa tai hallituksessa olisi perusymmärrys siitä, mitä ohjelmointi on ja miten tietojärjestelmät toimivat, voisi tulla parempia it-ratkaisuihin liittyviä päätöksiä.

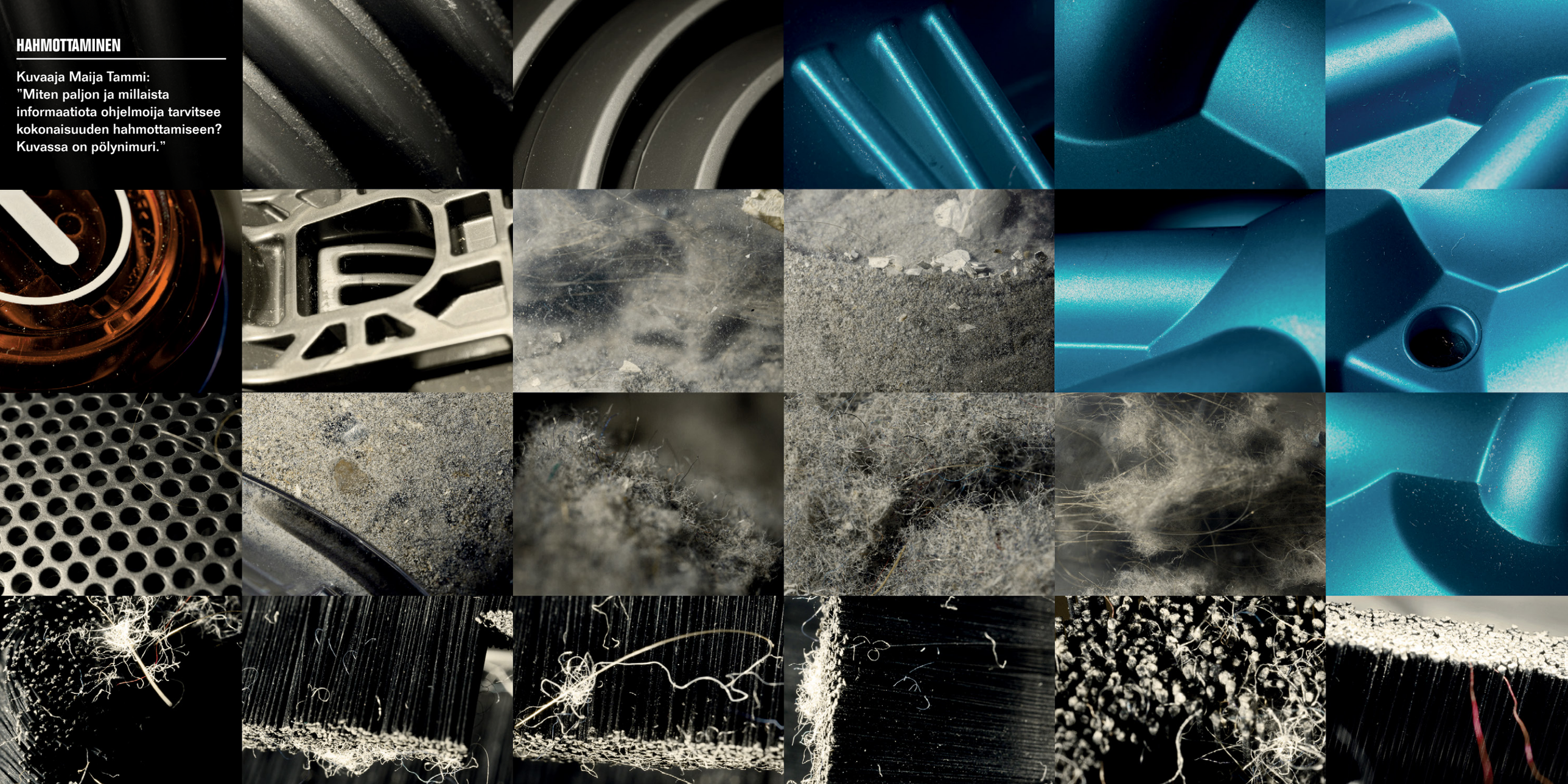
Se, että kaikki lukee vähän kemiaa ei tarkoita, että kaikista tulee kemistejä. Ohjelmoinnin lukeminen ei tee kaikista ohjelmoijia, mutta sitä kautta ymmärtää, miten Rovion pelejä voi tehdä, miten kommunikoit ystäviesi kanssa Facebookissa tai miten Hesarin verkkopalvelu toimii.

Tämä pitää nähdä isona, asioita ja ihmisiä yhdistävänä juttuna. Samalla tavalla kuin sinä voit tehdä hyllyn tai ommella vaatteen itse, voit myös rakentaa digitaalisia asioita itse.

Lapset voivat vaikka tehdä itse omat lelunsa ja tulostaa ne 3d-tulostimella.”

HAHMOTTAMINEN

Kuvaaja Maija Tammi:
”Miten paljon ja millaista informaatiota ohjelmoija tarvitsee kokonaisuuden hahmottamiseen? Kuvassa on pölynimuri.”



SUOMI EI OLE YKSIN

Muutkin maat aloittavat ohjelmoinnin opetuksen – Viro ja Britannia etunenässä.

Hyvin erilaiset maat ovat heränneet viime aikoina siihen, että ne haluavat opettaa ohjelmointia lapsille ja nuorille. Tällä hetkellä pisimmällä ovat Britannia ja Viro. Näissä maissa koodaus on jo tuotu systemaattisesti osaksi peruskoulua.

Virossa ensimmäiset alakoululaiset aloittivat uuden opetussuunnitelman kanssa syksylllä 2013. Britanniassa muutos, joka kattaa kaikki 5–16-vuotiaat, astuu voimaan syksyllä 2014.

Suomen kanssa suurin piirtein samassa tilanteessa ovat esimerkiksi Singapore, Etelä-Korea ja USA. Päätös koodaamisen tuomisesta opetusohjelmaan on joko tehty (Etelä-Korea), tai sitä tutkitaan (Singapore, USA).

Syyt muutokselle ovat maasta toiseen samanlaisia kuin tämän kirjan osiossa 1 on kuvailtu, mutta haasteet poikkeavat toisistaan:

Singaporessa monet vanhemmat toivovat lapsistaan pankkiireita, ei koodareita.

Britanniassa osaavia opettajia on niukasti.

Etelä-Koreassa taas opettajia löytyy, mutta tiukka koulutusjärjestelmä ajaa lapset valitsemaan aineita, joiden osaamista mitataan sikäläisissä ylioppilaskirjoituksissa.

Jopa Yhdysvaltain presidentti **Barack Obama** totesi toivovansa taitoa koodata, kun kauan odotettu terveydenhuoltouudistuksen portaali kaatuili ohjelmisto-ongelmien alla.

Kaikille haastavaa on pedagogisen ymmärryksen puute, kun puhutaan alakoululaisten koodaamisesta: pitäisikö aloittaa tekemisellä vai teorialla? Mitä ohjelmointikieltä pitäisi opettaa? Kuinka hallita eritasoisia oppilaita samassa ryhmässä? Miten ohjelmointi sidotaan osaksi eri aineita ja osaksi tieto- ja viestintäteknisen osaamisen kokonaisuutta?

Virossa uusi opetusohjelma otettiin käyttöön jo 2013.

ESIMERKKEJÄ MAAILMALTA

Britanniassa muutos alkoi helmikuussa 2011, kun Googlen silloinen toimitusjohtaja ja nykyinen hallituksen puheenjohtaja **Eric Schmidt** haastoi britit miettimään uudelleen koulujen teknologiakoulutusta.

Kahden vuoden aikana Brittien opetussuunnitelmaa työsti monipuolinen joukko opettajia, etujärjestöjä, yrityksiä ja yhdistyksiä. Syksyllä 2014 kaikki 1–4-luokat siirtyvät opiskelemaan uutta *Computing*-nimistä oppiainetta.

Opettajien ohjelmointiin liittyviä opetustaitoja kouluttaa vuoteen 2015 mennessä 400 opettajan erityisjoukko. Jokainen näistä ohjelmoinnin opeteukseen valjastetuista opettajista – niin sanotuista *master teachereista* – vastaa neljästäkymmenestä paikallisesta koulusta, jotka taas puolestaan kouluttavat lisää opettajia. Valtio tukee stipendein opettajien kouluttautumista.

Opettajat ovat kuitenkin melko yksin muutoksen kanssa, sillä yhtä yhtenäistä tuntisuunnitelmaa ei kouluille ole tarjolla.

Jokainen opettaja päättää itse, miten uutta ainetta aikoo opettaa.

Virossa uusi opetusohjelma otettiin käyttöön syksyllä 2013, mutta runsaasta 300 peruskoulusta keväällä 2014 vasta 55 on kokeillut ohjelmaa. Opettajat koulutettiin neljän viikon e-kurssilla, ja heitä kannustetaan vuorostaan kouluttamaan uusia opettajia.

MUUTOS EI LÄHDE VAIN KOULUISTA

Koululaitosten ulkopuolella tapahtuu myös. Kun ohjelmointiopetus on laahannut muodollisessa koulutuksessa jäljessä, on maailmalla syntynyt satoja erilaisia vapaaehtoisten vetämiä koodikouluja. Vapaaehtoisista koodiopettajista kerrotaan lyhyesti sivuilla 67 sekä 139.

Yhdysvalloissa Code.org-järjestö on saanut peräti 28 miljoonaa lasta kokeilemaan koodaamista ja rekrytoinut poptähti Will.i.amin sekä koripalloguru **LeBron Jamesin** kaltaisia julkkiksia puhumaan ohjelmoinnin puolesta. Kymmenissä maissa ohjelmointia opettavat myös vapaaehtoisuuteen perustuvat irlantilainen CoderDojo sekä brittiläinen Code Club. Molempien sivuilta löytyy materiaaleja meillekin opetusohjelman suunnitteluun:

code.org
coderdojo.com
codeclub.org.uk

MITÄ ERI MAISSA OPETETAAN NYT?

Ohjelmoinnin opettaminen peruskoulussa on uutta, mutta jostain on lähdettävä. Esimerkisi Britanniassa on aloitettu oma Computing-niminen oppiaineensa.

BRITANNIA

Britanniassa opettajilla on iso valta päättää, mitä ja miten opetetaan. Jokaiselle luokkatasolle on kuitenkin kirjattu joukko tavoitteita, joita lasten ja nuorten tulisi oppia. Keinot tavoitteisiin pääsemiseen saa päättää luokittain.

Huomioitavaa on, että siinä missä Suomessa ohjelmoinnin opetuksella otetaan aikaa muista oppiaineista, Britanniassa se liittyy omaan Computing-oppiaineeseensa, minkä myötä aikaa sisältöjen oppimiselle on enemmän.

Seuraavassa on esimerkkejä brittien eri luokka-asteiden tavoitteista.

Opetussuunnitelmiin voi perehtyä osoitteissa: www.naace.co.uk/naacecurriculum ja computingatschool.org.uk/index.php?id=cacfs

1. vaihe, luokat 1–2

1. vaiheen tavoitteet ovat kokonaisuudessaan:

- oppilas tietää, mitä algoritmit ovat ja miten annetaan tarkkoja ohjeita
- oppilas luo ja korjaa yksinkertaisen ohjelman
- oppilas käyttää loogista päättelyä ennustaakseen yksinkertaisen ohjelman lopputuleman
- oppilas käyttää teknologiaa tavoitteellisesti luodakseen, järjestääkseen, muokatakseen ja hakeakseen digitaalista sisältöä.
- oppilas tunnistaa, miten informaatioteknologiaa käytetään koulun ulkopuolella
- oppilas tietää, miten teknologiaa käytetään turvallisesti ja yksityisyyttä kunnioittaen

Muista vaiheista on seuraavassa esimerkkejä:

2. vaihe, luokat 3–6

Oppilas muun muassa suunnittelee, kirjoittaa ja korjaa ohjelmia saavuttaakseen tavoitteen. Oppilaat ratkaisevat isoja ongelmia pilkkomalla ne pienempiin.

3. vaihe, luokat 7–9

Oppilas muun muassa käyttää kahta tai useampaa ohjelmointikieltä, joista ainakin yksi on tekstipohjainen ratkaistakseen erilaisia ongelmia. Oppilas käyttää ongelmaan sopivia tietorakenteita ja suunnittelee ja ohjelmoi modulaarisia ohjelmia.

4. vaihe, lukio/ammattikoulu

Oppilas kehittää osaamista, luovuutta ja tietoja muun muassa tietojenkäsittelytieteessä, digitaalisessa mediassa ja informaatioteknologiassa.

USA:ssa Code.org-järjestö on kehittänyt alle 8-vuotiaille 25 tunnin opetussuunnitelman.

YHDYSVALLAT

Yhdysvalloissa ohjelmointiopetus päätetään liittovaltiotasolla. Erilaiset edunvalvontajärjestöt, kuten ohjelmointiopetusta edistävä Code.org ja tietotekniikan opettajien yhdistys Computer Science Teachers Association (CSTA) ovat kuitenkin rakentaneet omia opetussuunnitelmiaan, joita yksittäiset opettajat ja koulut voivat halutessaan hyödyntää.

CSTA on kehittänyt jo vuosia sitten alle 12-vuotiaiden opetussuunnitelman, jota käytetään standardina monille muille opetussuunnitelmille. Se koostuu neljästä teemasta: tietojenkäsittelytieteen perusteet, tietojenkäsittelytiede ja ohjelmointi osana modernia maailmaa, tietojenkäsittely analyysin ja suunnittelun välineenä sekä neljäs, vaativampien tietojenkäsittelytieteen opintoihin valmistava kokonaisuus: csta.acm.org/Curriculum/sub/K12Standards.html

Code.org-järjestö on kehittänyt alle 8-vuotiaille 25 tunnin opetussuunnitelman, jossa opitaan esimerkiksi, mitä tietojenkäsittelytiede on, mitä ohjelmoija tekee, miten ohjelmointia käytetään maailmassa, miten ohjelmia korjataan sekä joukon ohjelmoinnin peruskäsitteitä (peräkkäisyys,

silmukat, ehtolauseet, funktiot ja muuttujat) sekä tietokoneperusteisen ongelmanratkaisun teemoja (ongelmien purkaminen, hahmontunnistus, algoritmit). Opetussuunnitelma löytyy osoitteesta code.org/educate/curriculum.

Virossa opettajaksi halutut rekrytoidaan harmillisen nopeasti muihin kuin opetustöihin.

VIRO

Virossa on ollut haastavaa löytää opettajia, sillä tietojenkäsittelytieteitä opiskelleet rekrytoidaan yleensä nopeasti muihin kuin opettajantöihin.

Viron opetussuunnitelma painottaa käytännön tekemistä:

- 1.–3. luokka: Oppimisympäristönä toimii visuaalinen ohjelmointiympäristö Kodu Game Lab, sekä MSW Logo, joka on niin sanottu miniohjelmointikieli. Lisätietoja kummastakin on esitetty tämän oppaan osiossa 3.
- 4.–6. luokka: Oppilaat saavat valita joko Lego-robottien ohjelmoinnin tai verkko-ohjelmoinnin. Tässä vaiheessa esitellään myös MIT:ssä kehitetty lapsille suunnattu visuaalinen Scratch-ohjelmointiympäristö. Lisätietoja kummastakin on osiossa 3.

”KAIKKEA EI VOI PERUSTELLA NUMEROILLA”

Juha Paananen, *ohjelmistoarkkitehti, Reaktor*

”Koodauksen opettaminen on tärkeää, jos Suomesta halutaan tehdä koodauksen supermaa. Meidän on jatkossakin pysyttävä kehityksen kärjessä.

Reaktor tekee aktiivisesti omia juttuja asian eteen. Tällainen asia on esimerkiksi niin kutsuttujen Koodikoulujen järjestäminen.

Keksin idean Koodikoulusta, kun halusin opettaa omalle lapselleni ohjelmointia. Koodikoulussa opetetaan lapsille ja heidän vanhemmilleen ohjelmoinnin alkeita hauskojen ja innostavien esimerkkien kautta.

Jos olemme Suomessa tarpeeksi hyviä, niin ohjelmointihommia tehdään jatkossakin täällä, ja tänne tulee työpaikkoja ja hyvinvointia kaikille. Parhaat jutut tapahtuvat nimenomaan Suomessa.

Suomi ei ole mitenkään jälkijunassa näissä asioissa. Vapaan lähdekoodin ohjelmistoissa meillä on jo hieno edustus. Moni keskeinen softa, kuten Linux-käyttöjärjestelmä ja MySQL-tietokanta, on suomalaisten duunaama.

Sillä tiellä pitäisi jatkaa.

Toisaalta internetpalvelupuolella Suomi ei ole kauheasti ollut esillä – Skype ja Spotify ovat kummatkin naapurimaistamme. Eli petrattavaa on.

Reaktor järjestää myös Hello World Open -nimisiä ohjelmoinnin MM-kisoja. Kutsumme ohjelmoijat ympäri maailman mittelemään taitojaan ja finalistit tapaamaan toisiaan Helsinkiin.

Se ei ole täysin rationaalinen bisneshomma – me vaan halutaan tehdä se. Se on siisti juttu.

Kaikkea ei voi perustella numeroilla. Mutta toivottavasti siitä tulee jotain makeeta, jota ei ole aiemmin tapahtunut. Samalla luomme Suomesta kuvaa it-alan edelläkävijänä.”

Kysymyksiä ja vastauksia ohjelmoinnista:

SEITSEMÄN OPETTAJAN PELKOA

Olemme haastatelleet tätä opasta varten joukon opettajia ja opetusalan päättäjiä niin Suomessa kuin ulkomaillakin erilaisissa seminaareissa, sosiaalisessa mediassa, verkkokyselyin ja satunnaisissa kohtaamisissa.

Tässä luvussa käydään läpi muutamia esiin nousseita opettajien huolenaiheita.

1. MITEN VOIN OPETTAA OHJELMOINTIA, KUN EN OLE KOSKAAN OHJELMOINUT?

Ohjelmoinnin opettamisen aloittaminen arveluttaa ja aiheuttaa helposti hämmennystä, jos ei ole koskaan itse ohjelmoinut. Moni opettaja tietää ohjelmoinnista lähinnä jotain sen suuntaista, että tietokoneelle kirjoitetaan koodikieltä, jonka lopputuloksena syntyy tietokoneohjelmia, joissa kone tekee ihmisen pyynnöstä asioita.

Ei huolta.

Ohjelmoinnin opettaminen peruskoulutasolla – varsinkin alakoulussa – on vielä kaukana varsinaisesta monimutkaisten ohjelmien teosta, johon paneudutaan esimerkiksi yliopistotasolla.

Peruskoulussa tärkeintä on saada lapset ja nuoret ymmärtämään ohjelmoinnissa hyödyllisten ajattelumallien perusteet. Se, millaisia asioita tietokone on hyvä tekemään, ja miten tietokoneelle annetaan yksikäsitteisiä ohjeita.

Tällaisiin tarkoituksiin löytyy huomattava määrä erityisesti lapsille ja nuorille suunnattuja harjoituksia, verkkoresursseja, pelejä ja niin edelleen. Opettajan ei tarvitse osata syvällisesti ohjelmoida. Minimissään riittää, että opettaja osaa selittää, miksi ohjelmointi on tärkeää (ks. osio 1), ja ohjaa oppilaat harjoitusten pariin sekä saattaa heidät niiden läpi. Tähän auttaa ennakkoperehtyminen harjoituksiin.

Yläkoulutasolla ruvetaan perehtymään jo varsinaiseen ohjelmointikieleen. Tämä on toki pykälää haastavampaa, mutta ei missään nimessä ylitsepääsemätöntä. Myös oikeiden ohjelmointikielten opettelua varten on olemassa alkeista pienin askelin eteneviä oppimisympäristöjä, jotka on varta vasten suunniteltu vasta-alkajille.

Muista: harjoitukset ja oppisisällöt on valittava siten, että koko ikäluokka kykenee saamaan niistä irti. Opettajan omat oppimisvalmiudet ovat varmuudella niin hyvät, että hän kykenee omaksumaan asiat, jotka on tarkoitettu peruskouluikäisten opittaviksi.

Tämän oppaan osiossa 3 on esitelty verkkoresursseja ja harjoituksia, joita voidaan ottaa luokissa käyttöön ala- ja yläkoulutasoilla.

2. OHJELMOINTI EI KIINNOSTA – MIKSI OPETTELISIN UUTTA?

Kukaan ei voi pakottaa sinua innostumaan ohjelmoinnista. Mutta vaikka itse et olisi aiheesta innoissasi, olisi tärkeää, että kaikki oppilaasi saisivat mahdollisuuden siitä innostumiseen.

Kyse ei ole vain opettajan velvollisuudesta – kyse on lasten ja nuorten *oikeudesta* tulevaisuuteen, jossa heillä on hyödyllisiä taitoja.

Yhä suuremman osan maailmasta rakentavat ihmiset, jotka osaavat luoda uusia asioita tietokoneen avulla. Jos opettaja ei pidä huolta siitä, että omat oppilaat altistuvat ohjelmointiajattelulle, nämä jäävät paljosta paitsi.

Jos et millään kykene tai halua opetella opettamaan ohjelmointia, sopikaa innostuneemman opettajan kanssa tuntien vaihtamisesta päikseen.

Voitte kokeilla myös menetelmää, jossa toinen opettaja opettaa ja sinä kuuntelet oppilaiden mukana. Tämä voi olla kivuton tapa sepä oppia lisää että päästä jyvälle siitä, miten ohjelmointia voi opettaa.

Opettaja kykenee varmuudella oppimaan asiat, jotka on tarkoitettu peruskouluikäisten opittaviksi.

Koodi2016 | Miten OPS muuttuu – ja miksi?

3. MISTÄ OHJELMOINNIN OPETTAMISELLE LOHKAISTAAN TUNTEJA?

Opetussuunnitelmassa aika ohjelmoinnille on tarkoitus varata matematiikan tuntijaosta. Matematiikasta on jäämässä vuoden 2016 opetussuunnitelmassa vastaavasti pois joitain asioita, kuten jakokulman opettelu.

Käytännössä ohjelmoinnille ei anneta opetussuunnitelmassa kiinteää tuntimäärää, vaan asia jää koulujen ja opettajien päätettäväksi. Esimerkiksi alakoulun luokanopettaja voi omalla pelisilmällään sijoittaa ohjelmoinnin oppeja ja harjoituksia muihinkin aineisiin kuin matematiikkaan, ja tämä on myös suotavaa.

Sivuilla 124–126 on esimerkkejä siitä, kuinka ohjelmointia voidaan linkittää eri oppiaineisiin.

Tästä aikaa ohjelmoinnille: jakokulma jää pois matematiikan opsista 2016.

4. TIETOKONETAIDOT MUUTTUVAT NOPEASTI – EIVÄTKÖ NÄMÄ JUTUT VANHENE HETI?

Teknologia muuttuu nopeasti, ja ohjelmointikielet kehittyvät koko ajan.

Silti ohjelmoinnin opettamisen perusta on ajattelussa, joka ei lähitulevaisuudessa ole muuttumassa miksikään. Ihmiseen pitää jatkossakin ymmärtää ongelma, osata jäsentää se tietokoneen hahmottamiin osiin ja kyetä kertomaan tietokoneelle, mitä sen pitää tehdä.

Perustavanlaatuinen ymmärrys tietokonepohjaisesta ongelmanratkaisusta ja uuden luomisesta on yleissivistystä. Passiivisen ohjelmistojen käytön opettaminen on nopeammin vanhentuvaa tietoa.

5. KOULUSSAMME EI OLE IT-RESURSSEJA

Resurssitilanne käytettävissä olevien pöytäkoneiden, läppärien, tablettien, atk-luokkien, videotykkien ja vastaavien laitteiden kanssa vaihtelee suomalaisissa kouluissa valtavasti. Monissa kouluissa resurssit ovat opettajien mielestä riittävän hyvässä kunnossa, kun taas joissain kouluissa koetaan, että tietokoneita on liian vähän – ja niiden harvojen käynnistyminen vie puolet oppitunnista.

Me ulkopuoliset koodausintoilijat voimme

Monia ohjelmoinnin ajatusmalleja voi opettaa myös ilman tietokonetta.

vain toivoa, että resurssit laitetaan ylhäältä käsin kuntoon ennen syksyä 2016. Alla on kuitenkin pari vinkkiä siihen, miten niukkuudesta huolimatta voi pärjätä.

- Jokainen lapsi ei tarvitse omaa tietokonetta oppiakseen ohjelmointia. Ammattiohjelmoijatkin työskentelevät usein pareissa, ja mikä tärkeämpää, monia tässä oppaassa esitettyjä verkkoharjoituksia voi pohtia ja tehdä 2–4 hengen ryhmissä.
- Monia ohjelmoinnin peruskonsepteja voi opettaa ilman tietokonetta. Esimerkiksi sivusto csunplugged.org esittelee pelkästään ohjelmointiharjoituksia, joita voi tehdä ilman laitteita. Sivuilla 86–89 on havainnollistettu ja esitelty tällaisia harjoituksia.
- Monet ohjelmointiympäristöt eivät vaadi ollenkaan asentamista koneelle, vaan alkuun pääsee suoraan selaimessa. Jotkin näistä ympäristöistä puolestaan toimivat kaikissa päätelaitteissa älypuhelimista tabletteihin ja läppäreihin. Tällöin oppilaat voivat käyttää oppimiseen omia laitteitaan.

Sivuilla 112–118 on esitelty mobiililaitteille tarkoitettuja oppimisresursseja.

6. VOINKO SAADA TÄYDENNYSKOULUTUSTA OHJELMOINNIN OPETTAMISTA VARTEN?

Todennäköisesti voit. Opettajana asian tietänetkin, mutta yleisimmät neljä täydennyskoulutusvaihtoehtoja ovat seuraavat:

1. Kouluttautumiseen voi käyttää virkaehtosopimuksen mukaisia veso-koulutuksia, joita opettajalla on kolme päivää vuodessa. Työnantaja päättää, mitä koulutus sisältää, mutta toivoa sopii, että kun opetussuunnitelmaan tulee ohjelmoinnista ilmeinen uusi osa, niin koulut ja kunnat kohdentaisivat osan vesopäivistä ohjelmoinnin opettamiseen.

2. Opetushallitus järjestää vuosittain täydennyskoulutusta. OPH:n matematiikan opetussuunnitelmatyöryhmän puheenjohtajan Leo Pahkinin mukaan on todennäköistä, että koulutuksia kohdennetaan ohjelmoinnin alueelle lähivuosina.

3. Myös opetus- ja kulttuuriministeriö rahoittaa opettajille täydennyskoulutuksia. OPH kilpailuttaa niiden tarjoajat ja sisällöt.

Toivottavasti mukaan saadaan ohjelmointiopetuksen koulutuksia.

4. Opettaja voi itse hakea – omaan piikkiinsä – ohjelmoinnin täydennyskoulutusta esimerkiksi alueensa yliopistolta.

Ehdotuksia siitä, mitä eri tahot voivat tehdä ohjelmointiopetuksen edistämiseksi on esitetty osiossa 4.

Ohjelmointiharjoituksiin on paras tutustua tekemällä niitä etukäteen itse.

7. HYVÄ ON, HALUAN YRITTÄÄ. MISTÄ LÄHDEN LIIKKEELLE?

Osiossa 3 on esitelty resursseja joiden kautta päästä liikkeelle eri luokka-asteilla. Osion 3 sivuilla 120–121 on myös esitetty opettajan tarkistuslista ensimmäisen harjoituksen vetämiseen.

Ennen ensimmäistä oppituntia opettajan on hyvä selata tämä opaskirjanen läpi.

Seuraavaksi on hyvä perehtyä oman luokka-asteen oppilaiden tasolle sopiviin harjoituksiin.

Harjoituksiin on paras tutustua tekemällä niitä itse. Tällöin huomaa, millaisia ongelmia oppilaillakin luultavasti tulee vastaan, ja opettaja on sinut harjoitusten kanssa siinä vaiheessa, kun niitä ruvetaan tekemään luokan kanssa.

Onnea matkaan!

MIKSI TÄMÄ ON TÄRKEÄÄ? | **KOODI2016:N TUKIJAT KERTOVAT**

”KOKO AJATTELUTAPA MUUTTUU, KUN LAPSI TAJUAA KEHITTÄJÄN ROOLIN”

Juho Kokkola, *asiamies, Startup-säätiö*

”Maailma digitalisoituu. On tärkeää, että saadaan suomalaiset tulevaisuuden sukupolvet varhaisessa vaiheessa muutokseen mukaan, jotta heillä on tarvittavat taidot uuden yritysmaailman tarpeisiin.

Startup-säätiön tarkoitus on edesauttaa Suomea parempaan tulevaisuuteen yrittäjyyden kautta. Käytännössä haluamme olla luomassa toimivaa yrittäjyysekosysteemiä. Se, että Suomessa syntyy entistä parempaa ohjelmointiosaamista vastaa suoraan kasvuyritysten tarpeisiin. Softa on äärimmäisen merkittävässä roolissa tällä hetkellä.

Softan merkitys ei ainakaan vähene siitä, että ihmiset alkavat yhä paremmin ymmärtämään teknologian roolia. Se, että lapsi tai nuori tajuaa olevansa kehittäjän roolissa kuluttajan sijaan, muuttaa koko ajattelutapaa. Tämä vaikuttaa suoraan siihen, miten oppilas näkee omat tulevaisuuden kykynsä ja sen tuomat mahdollisuudet. Hän tajuaa, että on entistä enemmän työkaluja merkityksellisen elämäntyön tekemiseen.

Kyse ei ole siitä, että ilman koodaustaitoa ei voisi tehdä merkityksellisiä asioita, vaan siitä, että ohjelmoinnin tarjoamien vaihtoehtojen tajuaminen kasvattaa tätä mahdollisuuksien spektriä.

Itselleni on ollut erittäin paljon hyötyä siitä, että opettelin yliopistossa ohjelmointikielen. Kun näen koodia niin ymmärrän, mitä siinä lukee. Pystyn keskustelemaan ammattiohjelmoijan kanssa ja viemään palautetta, mielipiteitä sekä vaikka asiakkaan toiveita hänen suuntaansa. Tämä on avannut minulle uusia mahdollisuuksia siihen, mitä tehdä työelämässä. Lisäksi, jos haluaa perustaa oman firman, monialainen ymmärrys on ihan välttämätöntä.”

Tämä osio antaa käsityksen siitä,
miten ohjelmoinnin opetusta voi lähestyä peruskoulussa.
Kyseessä ei ole oppimateriaali vaan kokoelma esimerkkejä,
jotka toivottavasti sekä informoivat että inspiroivat opettajaa.

3 | *Miten ohjelmointia opetetaan?*

KOODAUKSEN OPETUS ON AJATTELUN OPETTAMISTA

Tässä osiossa lähestytään ohjelmoinnin opettamista peruskoulussa kahden pääharjoituskategorian kautta. Nämä ovat yhtäältä leikit sekä toisaalta erilaiset verkossa tarjolla olevat ympäristöt ohjelmointia ja sen opiskelua varten.

Jotta opettaja ja oppilas ymmärtäisivät, miksi esitellyt leikit tai verkkoharjoitukset auttavat oppimaan ohjelmoimaan, on syytä käydä läpi, mistä niin kutsutussa *ohjelmoinnillisessa ajattelussa* on kyse.

Ohjelmoinnillinen ajattelu on hiukan hapuileva suomennoksemme englannin kielen termistä *computational thinking*. (Toinen mahdollinen suomennos voisi olla vaikkapa algoritminen ajattelu. Computational thinkingistä löytyy oivallinen tulostettava luokkahuoneharjoitus Code.org-sivustolta: learn.code.org/unplugged/unplug2.pdf)

Ohjelmoinnilliselle ajattelulle on erilaisia määritelmiä, mutta usein sen esitetään koostuvan esimerkiksi seuraavassa esitetyn kaltaisista osa-alueista, jotka kaikki liittyvät siihen, minkälaisia ajattelutaitoja ihminen tarvitsee pystyäkseen ratkaisemaan ongelmia tietokoneen avulla.

Ongelman palastelua osiin tarvitaan arkielämässä ja ohjelmoinnissa.

Huomionarvoista on, että listatut taidot eivät ole ainoastaan ohjelmointiin liittyviä. Samanlaisia ongelmanratkaisun taitoja tarvitaan koulussa esimerkiksi filosofian, matematiikan ja vieraiden kielten tunneilla – arkisesta elämästä puhumattakaan.

ONGELMAN PURKAMINEN OSIIN

Ongelman purkaminen osiin (*decomposition*) on taito, jota tarvitaan sekä arkielämässä että ohjelmoinnissa, kun ongelma on hiukankin monimutkaisempi ratkaistavaksi kerralla.

Tyypillinen esimerkki ongelman purkamisesta osiin on, kun joku kysyy sinulta ajo-ohjeita Helsingin keskustasta luoksesi. Jos et asu Kolmen sepän patsaalla, on mahdotonta antaa yksikäsitteistä ohjetta yhdellä virkkeellä. Ongelma on hajotettava osiin: ”Lähde ensin Mannerheimintietä pohjoiseen. Kansallisoopperan kohdalla käänny oikealle. Jatka puoli kilometriä…” Ja niin edelleen.

Toinen esimerkki ongelman purkamisesta osiin olisi esimerkiksi siinä, kun maistat ystäväsi tekemää lihakeittoa ja haluat hahmottaa, miksi se on niin hyvää. Näet silmin, että liemessä on porkkanaa ja kaalia. Makuaistisi avulla pyrit tunnistamaan muita mausteita kuten suolan, pippurin ja vaikkapa valkosipulin.

Matematiikassa luvun 273 voi purkaa satoihin, kymmeniin ja ykkösiin: 2 x 100 + 7 x 10 + 3 x 1

Ohjelmoidessa vaikkapa verkkopankkia on välttämätöntä purkaa ongelma osiin:

Ketkä pankkia käyttävät? (Ihmiset, yritykset jne.)

Mitä käyttäjät haluavat pankissa tehdä? (Maksaa laskun, ottaa lainaa jne.)

Mistä toimenpiteet koostuvat? (Laskun maksaminen koostuu tunnusluvun antamisesta, summan syöttämisestä, vastaanottajan tilinumeron antamisesta jne.)

KAAVOJEN TUNNISTAMINEN

Kaavojen ja toistuvien sääntöjen tunnistaminen (*pattern recognition*) auttaa ratkaisemaan arkipäiväisiä ongelmia. Kun itkevä vauva hiljeni viime kerralla saadessaan tutin, sama ratkaisu voi auttaa tälläkin kerralla. Märkä puu näyttää syttyvän huonommin kuin kuiva.

Ohjelmoinnissa kykyä tunnistaa ongelmissa esiintyviä kaavoja on hyvin hyödyllinen, koska yhtäältä se auttaa ajattelussa kuten missä tahansa arkipäiväisessä ongelmassa, ja toisaalta kone on ongelmien ratkaisijana täydellisen kaavamainen: se toimii aina tarkalleen ja täydellisesti samalla tavalla suorituskerrasta toiseen. Siksi kerran oivallettua kaavaa voidaan soveltaa koneella yhä uudestaan oikein.

ALGORITMIEN LUOMINEN

Algoritmilla tarkoitetaan kuvausta jonkin tehtävän suorittamiseksi tarvittavista toimenpiteistä. Kun arkielämässä huippukokki kirjoittaa ylös

parhaan lihakeittonsa ainekset, niiden määrät ja suhteet sekä ohjeet ainesten valmistamiseksi keitoksi asti, kyseessä on paitsi keiton resepti myös sen valmistamiseen liittyvän algoritmin kuvaus.

Kokin kirjaamaa algoritmia tarkalleen toistamalla kuka vain voi kerta toisensa jälkeen tehdä yhtä hyvää lihakeittoa kuin huippukokki.

Ohjelmoinnissa ohjelmoija on kokki ja tietokone on apuri. Se tekee tarkalleen, mitä reseptissä eli algoritmissa lukee. Ohjeet riittää kirjoittaa kerran, jonka jälkeen kone toteuttaa ne joka kerta napin painalluksella.

RATKAISUN YLEISTÄMINEN JA AUTOMATISOINTI

Esimerkiksi matematiikassa pyritään ongelmia ratkaistaessa yleiseen muotoon. Lukiolainen saa toisen asteen yhtälön $2x^2 - 3x + 1 = 0$ ratkaisut selville suoraan toisen asteen yhtälön $ax^2 + bx + c = 0$ yleisestä ratkaisukaavasta (ei esitetty tässä), johon voimme sijoittaa luvut 2, -3 ja 1 lukujen a, b ja c tilalle.

Ohjelmoinnissa ongelmille haetaan hyvin usein yleistettyjä ratkaisuja. Kun ohjelman halutaan piirtävän näytölle suorakaiteen, olisi typerää kirjoittaa ohjelma siten, että se osaa tehdä vain suorakaiteita, joiden sivujen pituudet ovat aina 3 ja 5.

Sen sijaan ohjelma on järkevää kirjoittaa siten, että se osaa piirtää minkä tahansa kokoisen suorakaiteen riippuen siitä, mitkä mitat käyttäjä ohjelmaan syöttää.

Parasta ohjelmoinnissa on, että ratkaisuja voi automatisoida sälyttämällä vastuun niistä koneen harteille.

Kun ihminen ratkaiseen toisen asteen yhtälöä, hän joutuu sijoittamaan luvut kaavaan yhä uudestaan. Kun koneelle opettaa kaavan kerran, se antaa ajasta ikuisuuteen oikeat yhtälön ratkaisut nanosekunnissa riippumatta siitä, mitkä luvut a:n, b:n ja c:n tilalle laittaa.

Kun opettaja lähtee yksin tai luokan oppilaiden kanssa perehtymään tässä osiossa jäljempänä esiteltyihin leikkeihin tai harjoituksiin, hän huomaa pian, että ne kaikki opettavat tai harjoittavat yhtä tai useampia yllä esitetyistä taidoista: ongelman pilkkomista, kaavojen hahmottamista, algoritmien luomista ja ratkaisujen esittämistä yleistettävässä muodossa.

Ohjelmoinnin opettaminen onkin pohjimmiltaan näiden taitojen opettamista.

Toki: ohjeet annetaan tietokoneelle ohjelmointikielellä, jonka säännöt ja merkintätavat on myös tunnettava. Mutta mitä pidempään ihminen ohjelmoi, sitä pienemmäksi muuttuu syntaksin muistamisen rooli ja sitä enemmän korostuu yllä kuvattujen taitojen merkitys.

”Jos ymmärrät mitä teet, et opi mitään.”

– Abraham Lincoln

OHJELMOINNIN OPPIMINEN ON TEKEMISTÄ

Toinen hyvin keskeinen asia ohjelmoinnin opettamiseen liittyen on, että ohjelmointia ei todella voi oppia muuten kuin itse tekemällä.

Tässä mielessä ohjelmointi on kuin kuvaamataito, matematiikka, englanti, puutyöt tai vaikka jalkapallon pelaaminen. Kuka vain voi selostaa taustalla olevaa teoriaa ja sitä, mistä noin periaatteessa on kyse laveerauksessa, yhtälön ratkaisemisessa, keskustelemisessa britin kanssa, lankun höyläämisessä tai keskityksen antamisessa. Silti todellinen osaaminen tulee vain oman yrityksen ja erehdyksen kautta.

Ohjelmoijien kyvyt ovat usein pitkälle verrannollisia siihen, minkä verran ihminen on ohjelmoinut. Millaisia ongelmia hän on kohdannut, kuinka monta tuntia sinnikkäästi niiden parissa puurtanut ja intohimoisesti etsinyt parasta mahdollista toteutusta.

Lasten on päästävä heti tekemään ja näkemään työnsä tulokset.

Samasta syystä ohjelmoinnin opetuksen peruskoulussa ei tämän oppaan kirjoittajien näkemyksen mukaan ole syytä olla teorian tankkaamista. Päinvastoin: lasten ja nuorten on päästävä saman tien tekemään, saman tien kokemaan ja saman tien näkemään työnsä tulokset näytöllä.

Siksi seuraavassa läpikäytävät harjoitukset ja työkalut keskittyvät kaikki tekemiseen – eivät ohjelmoinnin syvälliseen filosofiaan tai vielä peruskoulutasolla esimerkiksi erilaisiin sovelluskehityksen suunnittelumalleihin.

MIKSI TÄMÄ ON TÄRKEÄÄ? | **KOODI2016:N TUKIJAT KERTOVAT**

”PELIT OVAT REALISTINEN TAPA VALLOITTAA MAAILMA”

Lassi Leppinen, *game lead, Supercell*

”Miksi tämä on tärkeää?

Lapselle perustelisin asiaa siten, että jos hän on innostunut peleistä, niin voisi olla myös kiva ymmärtää, miten ne on tehty – ja miten niitä tehdään lisää.

Suomessa on paljon menestyviä pelifirmoja, ja ala kasvaa nopeasti. Tällä alalla on aivan liian vähän hyviä tekijöitä verrattuna tarpeeseen.

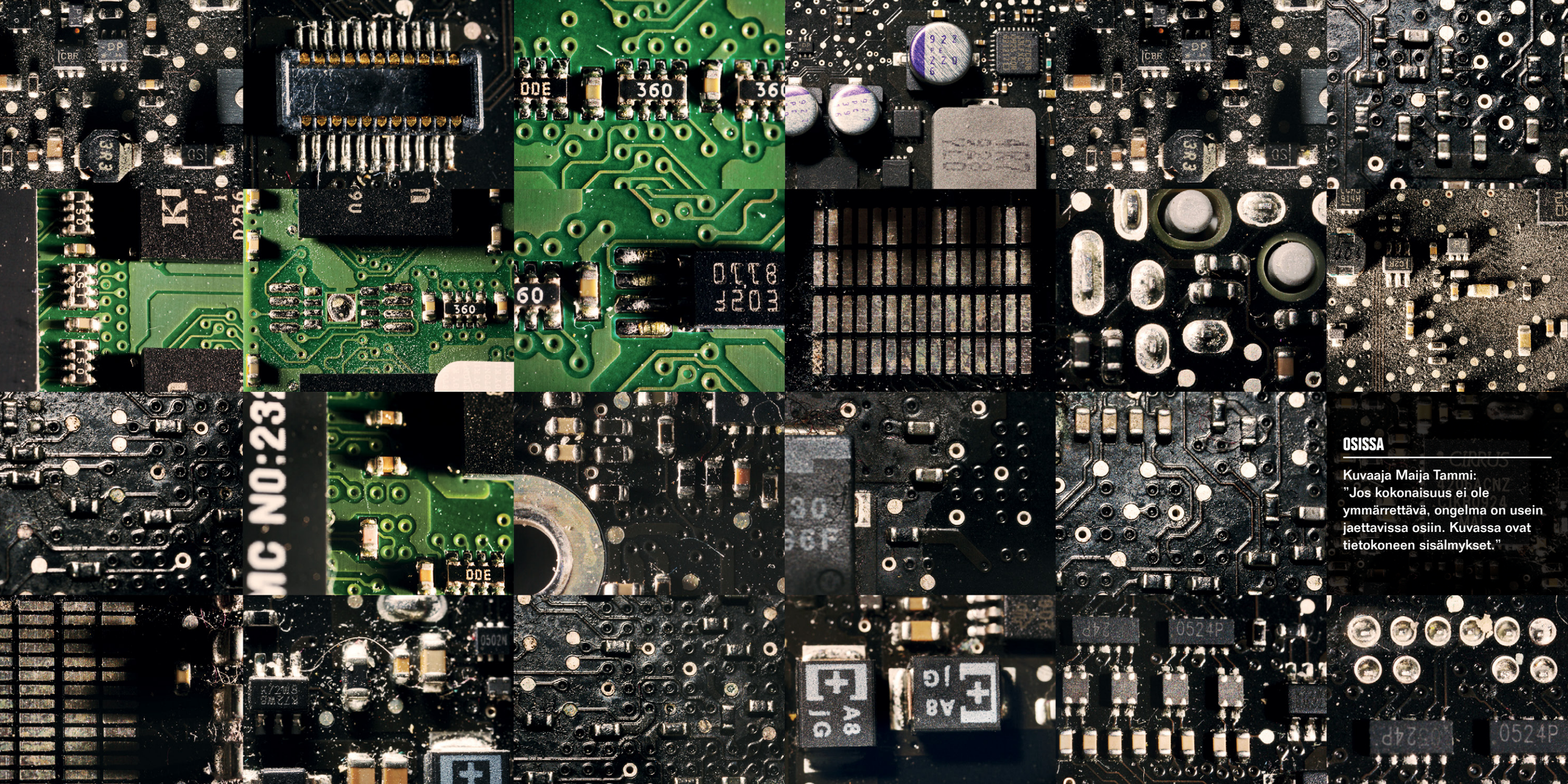
Esimerkiksi meille Supercell-pelitaloon hakijoita on paljon. Mutta sellaisia hakijoita, jotka olisivat tehneet kokonaisia peliprojekteja edes harrastuksena, on yllättävän vähän. Suuri osa ihmisistä, jotka ovat pystyneet toteuttamaan kokonaisen pelin harrastuksena, on myöhemmin päässyt töihin pelialalle.

Pelejä on hienoa rakentaa monestakin syystä. Olen tietysti itse tykännyt aina peleistä ja pelaamisesta. Pelinteossa on päässyt käyttämään luovuutta ja tekemään jotain, mitä muutkin pääsevät kokeilemaan.

Peleillä, kuten muillakin digitaalisilla tuotteilla, on myös käytännössä rajaton jakelumahdollisuus. Pelin voi laittaa esimerkiksi Applen tai Googlen sovelluskauppaan, ja hetkessä sillä voi olla miljoonia pelaajia. Pelin on tietysti oltava sellainen, josta ihmiset innostuvat.

Se ei vaadi rahaa samalla tavalla kuin fyysisen tuotteen rakentaminen ja monistaminen.

Pelit ovat siis realistinen tapa valloittaa maailma.”



OSISSA

Kuvaaja Maija Tammi: ”Jos kokonaisuus ei ole ymmärrettävä, ongelma on usein jaettavissa osiin. Kuvassa ovat tietokoneen sisälmykset.”

ENNEN HARJOITUSTEN ALOITTAMISTA

Sekä itse ohjelmointi että erilaiset ohjelmointikielet ovat ihmisten luomuksia. Niiden taustalla olevat ajatusmallit ovat kehittyneet enimmäkseen viime vuosikymmeninä ja elävät jatkuvasti. Siksi ne muuttuvat peruskoulumaisen opetuksen kannalta huomattavasti enemmän kuin esimerkiksi matematiikan tai biologian lainalaisuudet.

Myös ohjelmoinnin opettamiseen liittyvät työkalut kehittyvät jatkuvasti. Siksi ei ole yhtä oikeaa ja vakiintunutta tapaa opettaa ohjelmointia. Lisäksi erilaisille oppijoille sopivat erilaiset työkalut – toiset oppivat visuaalisesti, toiset puhetta kuuntelemalla, kolmannet kirjoista teoriaa lukemalla ja neljännet tekemällä.

Seuraaville sivuille on koottu esimerkkejä resursseista, joiden avulla opettaja pääsee alkuun ohjelmoinnin opettamisessa.

Käytännössä suurin osa listatuista työkaluista toimii netin välityksellä. Se tarkoittaa, että varsinaisia työkaluja ei ladata omalle tietokoneelle, vaan niitä käytetään Internet Explorerin, Firefoxin, Chromen tai Safarin kaltaisella nettiselaimella menemällä tiettyyn verkko-osoitteeseen – aivan kuin menisi Hesarin verkkosivuille

Suurin osa tässä suositelluista työkaluista toimii tavallisella nettiselaimella.

lukemaan uutisia tai Telkku.comiin katsomaan tv-ohjelmatietoja.

Yksi selaimella verkossa tekemisestä saatava hyöty on, että harjoitusympäristöjen alkuperäiset luojat voivat päivittää niitä jatkuvasti netissä yhä helppokäyttöisemmiksi ja paremmin toimiviksi. Opettajan – kymmenistä oppilaista puhumattakaan – ei siis tarvitsee päivittää ohjelmistoja omalla koneellaan.

Harjoitusympäristöt saattavat vaatia käyttäjälle oman profiilin tekemistä, mutta tästäkin seuraava hyvä puoli on, että tehdyt harjoitukset ja oppilaan kehitys säilyvät verkossa (siellä paljon puhutussa ”pilvessä”) tallessa ilman pelkoa tietojen katoamisesta fyysisen tietokoneen hajotessa.

Jos oppilaat ovat liian nuoria rekisteröitymään esimerkiksi omalla sähköpostiosoitteella, voi useimpia harjoituksista tehdä myös ilman rekisteröitymistä.

Jokaisessa alla luetelluista kategorioista on tässä oppaassa valittu esittelyyn yksi suositeltava opetuksen työkalu. Lisäksi on kerrottu siitä, minkä ikäisille työkalu soveltuu, miten tuntiin kannattaa valmistautua ja mitä työkalu vaatii käytettävältä laitteistolta. Lisäksi on annettu lyhyitä esimerkkejä muista työkaluista.

Opetusmateriaaleja on esitelty seuraavissa kategorioissa:

- *Leikit* – erilaisia ohjelmointiajattelua opettavia leikkejä, erityisesti luokka-asteille 1–2.
- *Visuaaliset ohjelmointiympäristöt* – luokka-asteille 3–6 sopivia työkaluja, jotka opettavat ohjelmointiajattelua ilman, että oppilaan tarvitsee kirjoittaa ohjelmakoodia.
- *Ohjelmointikielet* – luokka-asteilla 7–9 perehdyttäväksi tarkoitettuja oikeita ohjelmointikieliä, joista oppilaan on tarkoitus ymmärtää yläkoulun aikana perusteet.
- *Mobiiliapplikaatiot* – erityisesti älypuhelimille ja tableteille tarkoitettuja ohjelmoinnin oppimisen työkaluja.

Lisäksi osoitteessa Koodi2016.fi/koodauspaiva on esitelty resurssien pohjalta esimerkkejä niin kutsutuista koodauspäivistä, joita koulussa voidaan pitää eri luokka-asteilla.

1.–2. luokka:

NYT LEIKITÄÄN!

Ohjelmointiin sopivien ajattelumallien opettaminen ei vaadi aina tietokonetta. Peruskäsitteitä ja toimintatapoja voi opetella leikkien kautta.

Useimmissa ohjelmointiajattelua opettavissa leikeissä lapsi oppii ymmärtämään, millä tavalla tietokonetta on ohjeistettava, jotta se osaa ratkaista ongelmia. Tällöin leikissä pyritään siihen, että tietyt toimet suoritetaan tietyssä järjestyksessä tehtävän ratkaisemiseksi.

Luokanopettajan on hyvä hahmottaa, miten eri leikit liittyvät myöhemmillä luokilla esille tuleviin käsitteisiin, ja miten eri aineissa voi hyödyntää erilaisia leikkejä. Lapsille tärkeintä on tekemisen ilo ja myönteinen, innostunut altistuminen ohjelmointiin liittyville ajattelumalleille.

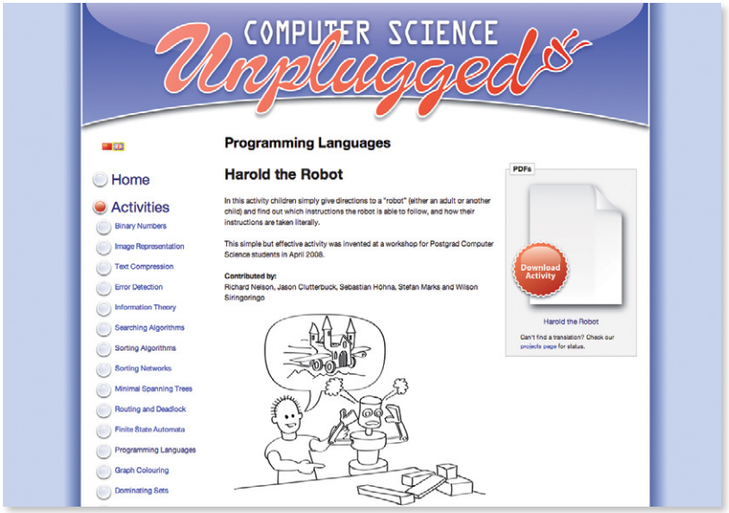
Kun lapsi etsii lyhimmän reitin ulos sokkelosta tai lajittelee laulut aakkosjärjestykseen, hän oppii myös ohjelmoinnissa hyödynnettäviä taitoja.

CS UNPLUGGED

Verkkosivusto CS Unplugged on kokoelma opettajien toistensa avuksi suunnittelemia ohjelmointia opettavia leikkejä, joiden leikkimiseen ei tarvita lainkaan tietokonetta tai muuta tekniikkaa. Sen sijaan CS Unpluggedin harjoitukset opettavat ohjelmoinnin perusteita pelien, liikunnan, paperille piirrettävien harjoitusten tai vaikkapa pingispallojen avulla.

Suureen osaan harjoituksista liittyy video. Harjoituksia tekevät vapaaehtoiset opettajat ympäri maailmaa. Opettajille on myös tehty oma, kevyt opaskirja.

Miten pääsee alkuun: Mene osoitteeseen csunplugged.org ja selaile sivustolla esitettyjä harjoituksia. Seuraavassa on myös esimerkki eräästä CS Unpluggedin harjoituksesta.



CS Unplugged -sivuston vasemmasta laidasta löytyy lista erilaisia ohjelmointia opettavia harjoituksia, joiden tekemiseen ei tarvita konetta. Opettaja voi ladata yksittäiset harjoitukset – kuten kuvassa esitetyn Harold-robotti-harjoituksen – tulostettavana pdf-tiedostona oikean laidan linkistä.

ESIMERKKI: HAROLD-ROBOTTI-HARJOITUS

Eräs esimerkki CS Upluggedin sisältämistä harjoituksista on nimeltään Harold-robotti (csunplugged.com/programming-languages-0).

Harjoituksessa opettaja esittää robottia, joka tottelee lapsia täydellisesti. Lapset puolestaan yrittävät saada opettajan suorittamaan jonkin tehtävän kuten kokoamaan palikoista tornin, tekemään voileivän tai juomaan kahvikupin.

Opettajan tehtävä on toimia tarkalleen lasten ohjeiden mukaisesti. Lapset oppivat pian, että komentojen on oltava hyvin tarkkoja, tai robotti toimii väärin.

Harjoitus opettaa – edelliseen kappaleeseen viitaten – esimerkiksi ongelman pilkkomista osiin

sekä algoritmista ajattelua.

Harjoitus voisi kulkea esimerkiksi seuraavasti:

Opettaja asettaa yksinkertaisen, esimerkiksi viisikerroksisen tornin kokoamiseen tarvittavat puupalikat luokan eteen pöydälle.

Opettaja seisoo paikoillaan tönkkönä ja alkaa esittää Harold-robottia. Harold pystyy vastaamaan vain tiettyihin komentoihin. Näitä komentoja ei anneta lapsille etukäteen.

Opettaja pyytää lapsia ohjaamaan Haroldia yksiselitteisillä komennoilla, kuten ”siirrä kättäsi vasemmalle” tai ”valitse kätesi vieressä oleva palikka”. Jos lapsi yrittää liian monimutkaista komentoa tai komentoa, joka ei sisälly Haraldin sanavarastoon (esimerkiksi ”laita kolme palikkaa päällekäin), opettaja pudistelee päätänsä.

Tehtävä on valmis, kun torni on rakennettu. Opettaja keskustelee lasten kanssa siitä, mihin komentoihin robotin voi olettaa voivan vastata ja mitä tietokonemaista robottia todennäköisesti ei voi pyytää tekemään.

TAVOITE: Lapsi oppii, minkälaisia ohjeita tietokone pystyy seuraamaan, ja miten kirjaimellisesti ohjeet on annettava, jotta aivoton kone voi seurata niitä.

KENELLE: Alakouluikäisille.

VAATIMUKSET JA VALMISTAUTUMINEN
Ei teknisiä vaatimuksia. Vaatii valmistautumista – opettaja ohjaa leikin.

KIELI: Englanti.

HINTA: Ilmainen, halutessaan voi ostaa kirjan jossa harjoituksia on kuvattu vielä yksinkertaisemmin.

OSOITE: csunplugged.org

MUITA 1–2-LUOKILLE SOPIVIA RESURSSEJA

	ROBOTTILEIKKI	COMPUTER SCIENCE-IN-A-BOX: UNPLUG YOUR CURRICULUM	COMPUTER SCIENCE FOR FUN
Lyhyesti	Kuuluisa labyrinttileikki, jossa lapset ohjaavat robottia kohti maalia.	CSUnplugged.org-sivuston pohjalta koottuja harjoituksia.	Aktiviteetteja alakouluikäisille brittiopettajilta.
Luokka	1–6	1–6	1–6
Valmistauminen	Vaatii valmistautumista, mutta harjoitukset ovat helppoja.	Vaatii valmistautumista – sisältää vastaukset ongelmiin ja pohjustavia videoita.	Vaatii valmistautumista – opetusmateriaalin valmistus.
Vaatimukset	Paperia, teippiä.	Luokkahuoneessa löytyvää tarpeistoa (paperia, magneetteja, liituja).	Paperia, pelikortit, videoprojektori.
Hinta	Ilmainen.	Ilmainen.	Ilmainen.
Kieli	Ohjeet englanniksi.	Ohjeet englanniksi.	Ohjeet englanniksi, saksaksi, ranskaksi ja venäjäksi.
Osoite	drtechniko.wordpress.com/2012/04/09/how-to-train-your-robot	ncwit.org/resources/computer-science-box-unplug-your-curriculum	cs4fn.org/teachers/activities

3.–6. luokka:

VISUAALINEN OHJELMOINTIYMPÄRISTÖ

Verkosta löytyy useita lapsille suunnattuja visuaalisia ohjelmointiympäristöjä sekä miniohjelmointikieliä. (”Miniohjelmointikieli” viittaa englannin termiin *educational programming language*, jonka voisi kääntää myös ”opetuskäyttöön luoduksi ohjelmointikieleksi”, mutta tässä käytetään lyhyempää muotoa ”miniohjelmointikieli”.)

Visuaalisia ohjelmointiympäristöjä ovat esimerkiksi Logo ja Scratch, jotka opettavat koodaamisen perusteita omassa näkymässään. Visuaalisia ohjelmointiympäristöjä käyttämällä ei voi koodata vaikkapa iPhonelle, mutta ne opettavat ajatusrakenteita ja taitoja, joita voi hyödyntää myöhemmin niin sanotusti oikeasti ohjelmoidessa.

Esimerkiksi Scratchin avulla tehdyt pelit ja muut tuotokset voi myös jakaa internetissä vaikkapa kavereiden ja sukulaisten saataville.

Visuaalisessa ympäristössä on varsinaista ohjelmointiympäristöä helpompi välttää virheitä, jotka vievät aloittelijan motivaation: Kun ohjelmoinnissa pilkku menee väärään paikkaan tai jostain puuttuu kirjain, sitä kutsutaan kirjoitusvirheen sijaan syntaksivirheeksi. Ne saattavat turhauttaa vasta-alkajaa, sillä yksikin virhe saattaa estää koko ohjelman toiminnan. Visuaalisissa ympäristöissä komentoja annetaan esimerkiksi raahaamalla ruudulla erilaisia palikoita. Sitä kautta päästään tekemisiin suurten linjojen kanssa, eikä pieniä tarkkuusvirheitä pääse syntymään.

Miniohjelmointikielet puolestaan näyttävät oikealta ohjelmoinnilta, mutta ne ovat oikeita kieliä helpompia ja nopeampia oppia. Monissa oikeissa ohjelmointikielissä on valtava määrä ominaisuuksia, joita tarvitaan ammattilaisten työympäristöissä monimutkaisia ohjelmistoja kehitettäessä. Tällöin yksinkertaisenkin asian tekeminen voi vaatia paljon koodia ja kokonaisen ohjelmointiympäristön hallintaa.

Miniohjelmointikielissä keskitytään yksinkertaisiin alkeisiin. Ne opettavat ohjelmoinnin perusteita nopeasti ja näkyvin tuloksin. Monet miniohjelmointikieliin perustuvat ajatukset liittyvät tutkija **Seymour Papertin** työhön, ja lähes kaikista suosituista ohjelmointikielistä löytyy yksinkertaistettu aloittelijaversio. Pitkä lista näistä ohjelmointikielistä löytyy Wikipediasta: en.wikipedia.org/wiki/Educational_programming_language

SCRATCH

Scratch on tätä kirjoittaessa tunnetuin visuaalinen ohjelmointiympäristö, joka on – mahtavaa! – saatavilla keskeisiltä osin myös suomeksi.

Arvostetussa yhdysvaltalaisessa teknisessä yliopistossa MIT:ssä kehitetty Scratch on ilmainen ohjelmointiympäristö, joka opettaa ongelmanratkaisutaitoja ja alkeellista ohjelmointia. Scratchista on myös tätä kirjoitettassa tulossa alle kouluikäisille suunnattu tablettiversio, ScratchJR.

Scratchin ajatus on, että oppilas ei kirjoita koodia. Sen sijaan asiat tehdään pääosin hiirellä raahaamalla: esimerkiksi ruudulle luodaan hahmoja ja liitetään niihin toiminnallisuuksia. Toiminnallisuuskia eli palikkamaisia työkaluja käyttämällä ja yhdistelemällä oppilas voi muodostaa esimerkiksi pelejä ja tarinoita. Samalla hän oppii ohjelmoinnillista ajattelua kuten ongelmien pilkkomista, algoritmeja ja toistettavuuden perusteita.

Scratchia käyttävät opettajat ympäri maailmaa, ja sen ympärille on syntynyt paljon harjoituksia. Scratchin yli neljän miljoonan (!) esimerkin joukosta löytyvät ohjelmointiohjeet esimerkiksi oman sähköisen pianon, joulukortin tai seikkailupelin luomiseen.

Opettajan on syytä tutustua Scratch-ympäristöön ainakin parin tunnin ajan etukäteen ja valita alkuun yhteinen harjoitus. Harjoituksia voi valita joko Scratchin omasta ympäristöstä tai lukea muiden opettajien suosituksia esimerkiksi suomalaisen Scratch Klubin sivuilta (sites.google.com/site/scratchklubi) tai kansainväliseltä Scratched-sivulta (scratched.media.mit.edu).

Oppilaita kannattaa myös kannustaa jakamaan ja selittämään omia ohjelmiaan.

Scratchin peruskonseptin ymmärtää parhaiten esimerkin avulla:

ESIMERKKI: INTERKATIIVISEN TERVEHDYSKORTIN TEKEMINEN SCRATCHILLA

Esimerkiksi valitussa Scratch-harjoituksessa tehdään interaktiivinen tervehdyskortti, esimerkiksi äitienpäiväkortti. Harjoitus kestää noin oppitunnin.

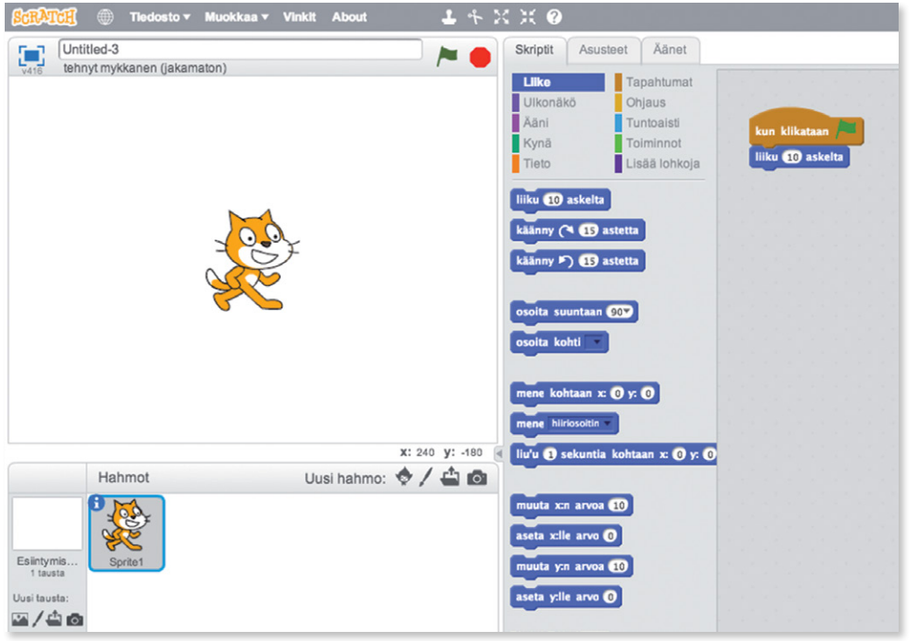
1. Ohjaa aluksi oppilaat osoitteeseen scratch.mit.edu/projects/editor/?tip_bar=hoc (Huom.! Kaikki harjoitusten osoitteet ja mitkä tahansa www-osoitteet voi lyhentää vain muutaman kirjaimen mittaiseksi dy.fi-osoitteesta löytyvän palvelun avulla. Lyhyet osoitteet on huomattavasti helpompi jakaa oppilaille.)

2. Selitä oppilaille, että oppitunnin tarkoituksena on harjoitella ohjelmointia luomalla jaettava sähköinen tervehdyskortti (esimerkiksi äitienpäiväkortti, joulukortti, pääsiäiskortti tms.). Kerro, että sähköisestä, nettiselaimessa toimivasta kortista tulee sellainen, jota klikkaamalla sen hahmot liikkuvat, kortista kuuluu ääniä tai muita jännittäviä asioita tapahtuu sen mukaan, millaisen kortin kukin oppilas haluaa tehdä.

3. Harjoituksessa on keskeistä nimenomaan lisätä korttiin erilaisia toiminnallisuuksia, jotta oppilaat pääsevät harjoittamaan ohjelmoinnillista ajatteluaan: mitä minun pitää käskeä tietokonetta tekemään, jotta ruudulla tapahtuu haluamani asia.

4. Aloittakaa vaihtamalla Scratchin kieli suomeksi klikkaamalla sivuston vasemmasta ylänurkasta maapallon kuvaa ja etsimällä valikosta sana "suomi". Jos oppilaat ymmärtävät englantia, voitte katsoa yhdessä oikeassa laidassa olevan videon, joka esittelee Scratchin perusideaa sekä joulukorttiharjoitusta.

5. Kortin tekeminen koostuu vaiheista: aluksi näytölle lisätään hahmo, sitten hahmoon liitetään toiminnallisuus. Tämän jälkeen kortille lisätään taustakuva ja lopuksi kortti jaetaan ystäville.



Scratchin perusnäkymä. Vasemman ylänurkan maapallokuvakkeesta vaihdetaan kieleksi suomi.

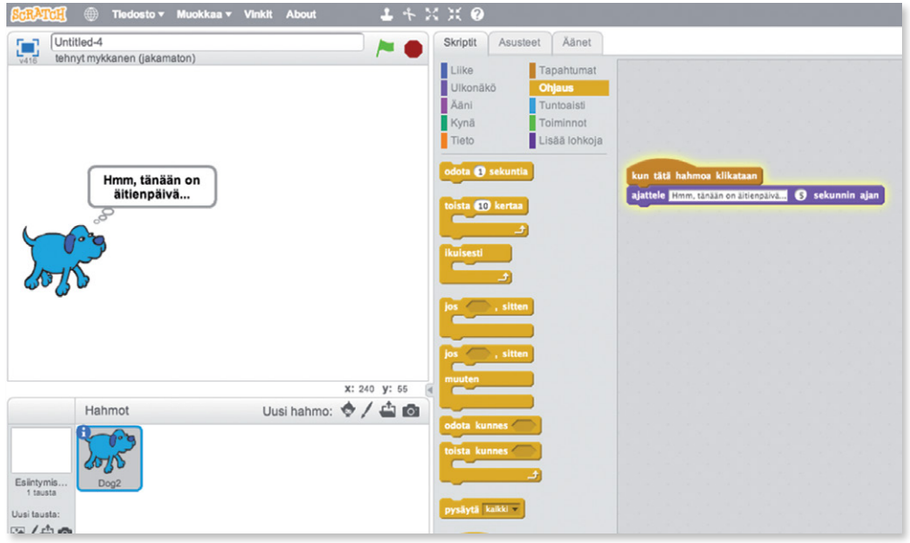
6. Hahmon lisääminen aloitetaan tuomalla näytölle jokin Scratchin tarjoamista valmiista kuvioista, jotka löytyvät Uusi hahmo -palkista. Scratchissa kaikki ohjelmat perustuvat hahmoihin ja niiden toimintaan. Hahmo voi olla valittu valmiista galleriasta, itse piirretty tai vaikkapa oppilaan oma valokuva itsestään. (Näytöllä valmiiksi olevan kissahahmon voi poistaa klikkaamalla ensin yläpalkin saksikuvaketta ja klikkaamalla sitten kissaa.)

7. Oheisessa kuvassa ruudulle tuotu hahmo on koira. Kun hahmo on tuotu ruudulle, siihen voidaan liittää erilaisia toiminnallisuuksia. Tämä on Scratchin koko konseptin ydin: oppilaan ei tarvitse kirjoittaa ohjelmakoodia, vaan hän valitsee valikosta erilaisia valmiita toiminnallisuuk-

sia ja yhdistelee niitä sekä toisiinsa että näytöllä oleviin objekteihin mielensä mukaan. Toiminnallisuuksia käytetään yksinkertaisesti raahaamalla ne hiirellä valikosta toiseen ikkunaan.

8. Tehkää aluksi esimerkin vuoksi yhteinen toiminnallisuus, jota oppilaat voivat muokata. Pyydä oppilaita valitsemaan Tapahtuma-välilehdeltä ”Kun tätä hahmoa klikataan”-toiminnallisuuden sisältävä palikka ja raahaamaan se näkymän oikeanpuoleiseen osaan.

9. Seuraavaksi pyydä oppilaita menemään Ulkonäkö-välilehdelle ja raahaamaan edellisen palikan alle uusi toiminnallisuus nimeltä ”Ajattele ‘TÄHÄN VALITSEMANNE TEKSTI’ 5 sekunnin ajan”.

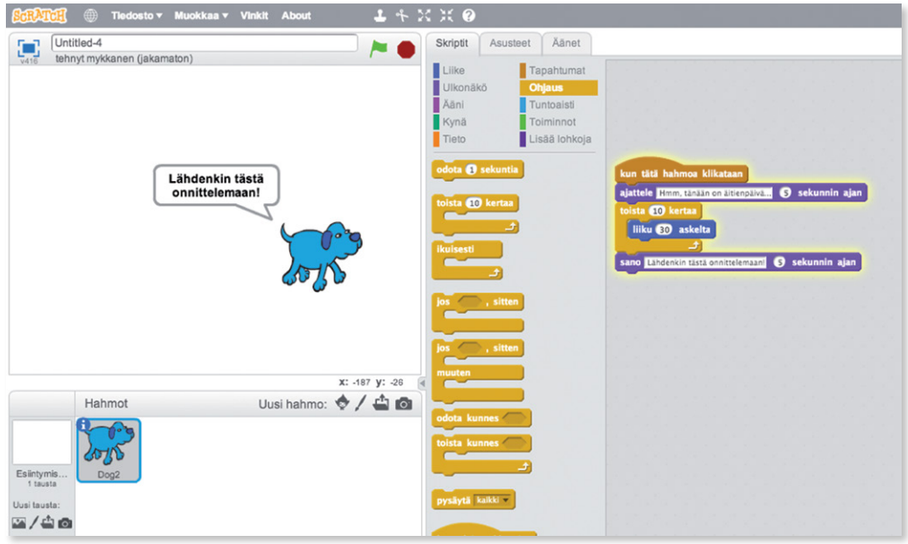


Eri värisiä toiminnallisuuspalikoita raahataan oikeanpuoleiseen ikkunaan, jolloin niistä syntyy toiminnallisuuksia vasemmassa ikkunassa näkyvälle koiralle. Kuvassa koiralle on lisätty toiminnallisuus, jonka mukaisesti koiraa klikkaamalla ruudulle ilmestyy puhekupla: ”Hmm, tänään on äitienpäivä...”

10. Klikatkaa hahmoa. Hahmolle syntyy puhekupla, jossa valittu teksti lukee.

11. Tässä vaiheessa oppilas on saanut peruskäsityksen Scratchin toiminnallisuudesta. Seuraavaksi kukin voi kokeilla yhdistellä erilaisia toiminnallisuuksia.

12. Erilaisia vaihtoehtoja ovat esimerkiksi hahmon taustan muokkaus. Hahmon voi laittaa myös toistamaan komentoja, liikkumaan ja puhumaan. Oheisessa kuvassa hahmolle on lisätty seuraavaksi toiminnallisuus, jossa se liikkuu ruudun poikki ja toteaa äitienpäivän kunniaksi: ”Lähdenkin tästä onnittelemaan!” Lisää inspiraatiota löytyy myös 2 500 muusta kortista: scratch.mit.edu/studios/279432.



Koiralle on lisätty uusi toiminnallisuus, jonka mukaisesti koiraa klikkaamalla se liikkuu 10 kertaa peräkkäin 30 pienen pykälän verran oikealle. Liikkeen jälkeen koiralle ilmestyy puhekupla: ”Lähdenkin tästä onnittelemaan!”. Projektin voi tallentaa vasemman ylänurkan Tiedosto-nimisestä valikosta.

13. Kehota oppilaita jakamaan projektinsa ystävilleen tai vanhemmilleen ”Jaa”-painikkeesta näytön oikeasta ylänurkasta. Tämä avaa projektin omaan ikkunaansa, jonka linkin voi jakaa kavereille sähköpostina tai vaikkapa Facebook-viestinä.

14. Jos ja kun haluatte palata projekteihinne, muistakaa tallentaa ne klikkaamalla hiirellä vasemman ylänurkan valikosta: ”Tiedosto” > ”Tallenna nyt”.

Ohjeita opettajalle löytyy osoitteesta scratch.mit.edu/hoc/

MITEN PÄÄSEE ALKUUN: Scratch-ympäristö on tätä kirjoittaessa suomennettu jo suurelta osin. Scratchiin liittyy myös opettajille tarkoitettuja tukimateriaaleja suomeksi: avoinoppikirja.fi/tiedostot/muut/ohjelmointia_scratchin_kanssa.pdf

Tällä sivustolla on myös helposta vaikeaan etenevä lista harjoituksia: sites.google.com/site/scratchklubi/

Suomenkielisiä pelinrakennusohjeita löytyy myös osoitteesta: cs.helsinki.fi/group/linkki/materiaali/kerho/mustakivi/Scratch.html

Scratchin ympärillä pyörii kokonainen verkkosivusto englanniksi osoitteessa: scratched.media.mit.edu/resources

KENELLE: Alakouluikäisille, mutta myös vanhemmille.

VAATIMUKSET: Moderni selain sekä Adobe Flash Player. Scratch on ladattavissa myös Windows-, OS X- (eli Mac-) ja Linux-koneille.

KIELI: Englanti, osin suomennettu.

HINTA: Ilmainen.

OSOITE: scratch.mit.edu

”Kerro minulle, ja unohdan.
Opeta minulle, niin muistan.
Anna minun tehdä itse, ja opin.”

– Benjamin Franklin

MUITA 3–6-LUOKILLE SOPIVIA RESURSSEJA

	KODU	ALICE	APP INVENTOR
Lyhyesti	Microsoftin kehittämä visuaalinen ohjelmointiympäristö, jossa ohjelmoidaan pelejä.	3D-opetusympäristö, jossa opetellaan olio-ohjelmoinnin alkeita.	MIT:ssä kehitetty visuaalinen ohjelmointiympäristö, jolla voi tehdä Android-puhelimissa toimivia applikaatioita.
Luokka	7–9	3–6	4–9
Valmistauminen	Videotutoriaaleja esim. youtube.com/watch?v=KWUyPpwQllE	Tutoriaaleja, esim. alice.org/3.1/index.html	Vaatii valmistautumista – opettajalla tulisi olla hyvä käsitys materiaalista appinventor.mit.edu/explore/teach.html
Vaatimukset	Windows-käyttöjärjestelmä.	Windows-, OS X- (Mac) tai Linux-käyttöjärjestelmä.	Google-tili, Android-puhelin tai emulaattori.
Hinta	PC-versio maksuton, Xboxille viitisen euroa.	Ilmainen.	Ilmainen.
Kieli	Englanti.	Englanti.	Englanti.
Osoite	kodugamelab.com	alice.org/index.php	appinventor.mit.edu/explore

LOGO	TURTLE ROY	HACKETYHACK, KIDSRUBY, TRYRUBY.ORG
1960-luvulla kehitetty ohjelmointiympäristö, joka on suunniteltu opettamista varten.	Suomalainen Logoon perustuva ohjelmointiympäristö, jota on käytetty osana Koodikoulua.	Lapsille suunnattu versio Ruby-kielestä.
3–9	1–6	7–9
Vaatii valmistautumista – internetissä useita e-kirjoja.	Hyvä tutustua komentoihin etukäteen.	Vaatii valmistautumista.
Moderni verkoselain.	Moderni verkoselain.	Asennettava koneelle etukäteen, vaatii opettajalta valmistautumista.
Ilmainen.	Ilmainen.	Ilmainen.
Englanti.	Englanti.	Englanti.
el.media.mit.edu/logo-foundation/index.html	github.com/raimohanska/turtle-roy	hacketyhack.com tryruby.org kidsruby.com

7.–9. luokka:

OIKEA OHJELMOINTIKIELI!

Opetussuunnitelman luonnoksen mukaan yläkoulussa aletaan perehtyä johonkin oikeaan ohjelmointikieleen.

OPH:n suunnitelma ei ota kantaa käytettävään kieleen. Käytännössä mahdollisia vaihtoehtoja ovat esimerkiksi Python, Ruby tai JavaScript. Nämä ovat nykyaikaisia, eri käyttötarkoituksiin sopivia mutta ammattilaistenkin käyttämiä kieliä, joiden alkeet ovat ymmärrettävissä yläkoulutasolla.

Haasteena varsinaiseen ohjelmointikieleen hypättäessä on antaa oppilaalle hyvä ensikokemus ohjelmoinnista, ja saada hänet näkemään nopeasti työn tulokset, mikä motivoi jatkamaan kielen tutkimista ja oppimista. Avainasemassa on tässä tarjota tunne jatkuvasta edistymisestä ja ottaa niin pieniä edistysaskelia, ettei oppilas tipahda kyydistä.

Alla listattuja ohjelmointikieliä voi opettaa asentamalla kyseisen ohjelmointikielen omalle pöytäkoneelle tai läppärille. Toisaalta ohjatut, selainympäristössä toimivat tuotteet tarjoavat usein sekä valmiita neuvoja harjoitusten läpiviemiseksi että automaattisen tarkastuksen kirjoitetulle koodille. Suomenkielistä kirjallisuutta yläkouluikäisille on ikävä kyllä toistaiseksi tarjolla niukasti.

Tärkeintä on ottaa riittävän pieniä askelia ja tarjota oppilaalle tunne jatkuvasta edistymisestä.

KHAN ACADEMY

Mikä: Khan Academy on voittoa tavoittelematon sivusto, joka opettaa ohjelmoinnin lisäksi esimerkiksi matematiikan perusteita.

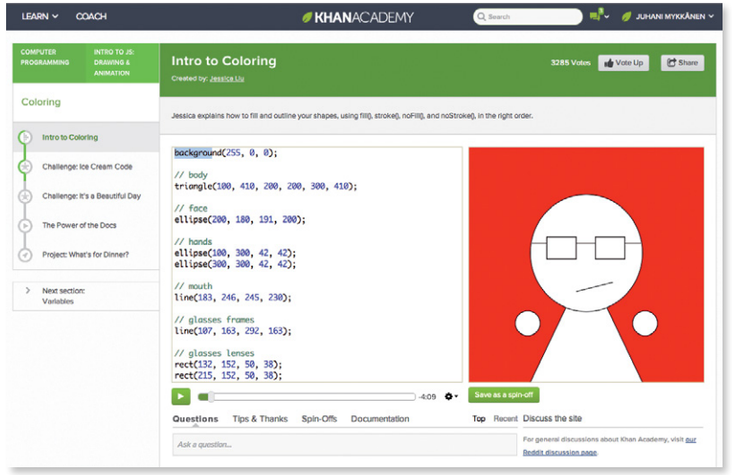
Khan Academyssä opetettava kieli on nimeltään JavaScript. JavaScriptiä käytetään lähes kaikilla verkkosivuilla, kun sivustolle halutaan ohjelmoida interaktiivisia toiminnallisuuksia, animaatioita, pelejä tai muuta vastaavaa. Khan Academy opettaa kielen perusteet interaktiivisten harjoitusten ja videoiden avulla.

Khan Academyssä oppilas katsoo ensin videon (*talk-through*), jossa seuraavaksi tehtävän harjoituksen sisältö selitetään. Kaikkiin teemoihin liittyy myös niin sanottu haaste (*challenge*), jossa interaktiivisessa ympäristössä kokeillaan käytännössä aiemmin selitettyä. Harjoituksiin liittyy myös niin sanottu projekti (*project*), jossa opittua sovelletaan.

Khan Academy toimii erityisen hyvin itseopiskelussa. Harjoituksissa opitaan JavaScript-kielen perusteita koodaamalla lautasmalleja ja muistipelejä, mutta tutustutaan myös ohjelmoijiin työssään.

Ohjelmointiin liittyviä harjoituksia ei löydy vielä tätä kirjoittaessa suomeksi, mutta muuta sivuston sisältöä on jo vapaaehtoisvoimin käännetty suomeksi.

ESIMERKKI: VÄRITYSHARJOITUS JAVASCRIPTILLÄ



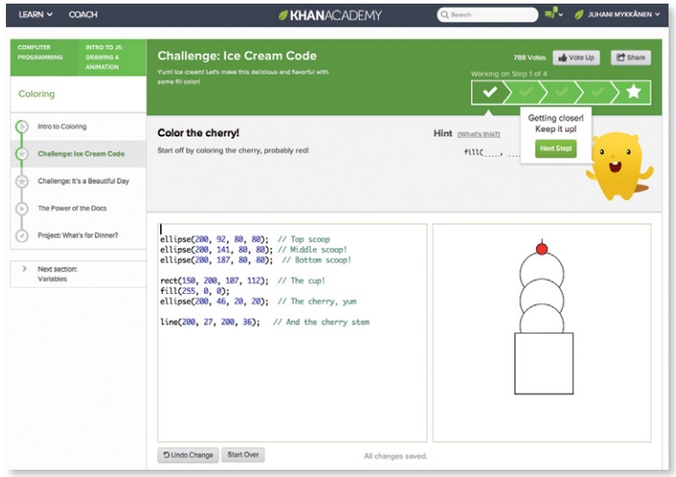
Khan Academyn väritysharjoitus JavaScript-kielellä.

Tältä näyttää harjoitus, joka liittyy JavaScriptillä värittämiseen Khan Academyssä. Vasemmalla näkyy koodia ja oikealla se, mitä koodi tekee (piirtää näytölle palloista, suorakaiteista ja viivoista koostuvan ukkelihahmon).

Play-nappia painamalla voi seurata videota, jossa naisääni selittää koodin toiminnallisuutta ja tekee muutoksia koodiriveihin. Selostus on englanniksi, mutta hammasratasta klikkaamalla saa näkyville tekstitykset. Tekstityksiä ei vielä tätä kirjoittaessa ole suomennettu.

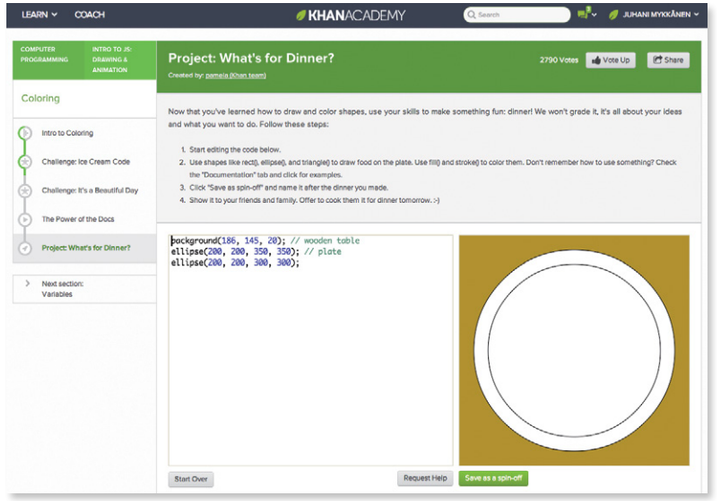
1. Harjoitus kannattaa aloittaa kuuntelemalla ohjeistus. Selostuksen voi pysäyttää milloin tahansa. Näkymän alareunasta voi myös tarkistaa, minkälaisia kysymyksiä muilla on ollut aiheeseen liittyen, minkälaisia versioita samasta ohjelmasta on tehty, ja mitä dokumentaatiota on tarjolla.

2. Selostuksen jälkeen on haasteen vuoro. Tässä kohden oppilas pääsee kokeilemaan taitojaan. Oheisessa kuvassa esitetyssä haasteessa muutetaan ensin koodin avulla kirsikka punaiseksi ja sen jälkeen askel askeleelta koko jäätelö. Kaikki komennot löytyvät näkymän alareunasta, ja tarvittaessa haasteisiin löytyy myös vinkkejä.



Toinen väritysharjoitus JavaScript-kielellä. Vasemmalla olevaa koodia muuttamalla väritetään oikealla olevan kakun osien värejä.

3. Niin kutsutun projektin ideana on soveltaa aiemmin opittua ja tarjota ympäristö kokeilemiseen. Projektin voi tallentaa. Seuraavan sivun kuvassa esitetyn projektin tarkoituksena on piirtää koodia käyttämällä lautaselle erilaisia yksinkertaisista geometrisista muodoista koostuvia ruokia.



Kolmas väritysharjoitus JavaScript-kielellä. Vasemmalla olevaa koodia muuttamalla luodaan oikealla olevalle lautaselle yksinkertaisia geometrisia muotoja, jotka edustavat ruokaa.

KENELLE: Khan Academy soveltuu yläkouluikäisille ja sitä vanhemmille.

VAATIMUKSET: Toimii lähes kaikilla tietokoneilla ilman asennusta.

MITEN PÄÄSEE ALKUUN: Khan Academy on hyvin ohjattu ympäristö, ja motivoitunut oppija käy ohjelmoinnin perusteet läpi ajatuksella muutamassa päivässä. Opettajan on hyvä kokeilla ensimmäiset harjoitukset läpi itse.

KIELI: Englanti.

HINTA: Ilmainen. Takana on voittoa tavoittelematon yhteisö.

OSOITE:
khanacademy.org/computing/cs

MIKSI TÄMÄ ON TÄRKEÄÄ?

KOODI2016:N TUKIJAT KERTOVAT

”HALUAMME, ETTÄ SUOMEEN KASVAA KOODAAJASUKUPOLVI”

Anni Vepsäläinen, *toimitusjohtaja, Diacor*

”Terveydenhuollossa ohjelmistot näkyvät jo nyt jokaisessa asiakaskohtaamisessa. Oli kyseessä sitten ajanvaraus netin välityksellä, puhelinajavaraus puhelinkeskuksen kautta, lääkärissä käynti, laboratoriotutkimusten tekeminen… Softa on joka paikassa läsnä. Se paitsi tukee toimintaa on myös ihan sen ytimessä.

Kun softa toimii hyvin, se parantaa terveydenhuollon asiakkaan kokemusta ja hänen saamansa palvelun laatua. Asiakkaalle hyvin tehty ohjelmisto näkyy esimerkiksi niin, että hän voi hoitaa asioitaan entistä enemmän sähköisesti. Palveluja voidaan viedä verkkoon. Tälläkin alalla asiakkaat ja potilaat arvostavat sitä.

Aina se henkilökohtainen kohtaaminen lääkärin kanssa jää mieleen, ja sitä sähköiset palvelut eivät korvaa. Mutta ne täydentävät sitä.

Kun sähköiset palvelut toimivat sujuvasti, niin asiakas voi hoitaa asioitaan ajasta ja paikasta riippumatta ja pääsee myös vastaanotolle nopeasti ja joustavasti.

Ohjelmiston laadulla on myös meille terveydenhuollon tarjoajille valtava merkitys. Jos meille tulee päivitys potilastietojärjestelmään tai muuhun ydinjärjestelmään, niin sillä, kuinka laadukkaasti se on rakennettu on välitön vaikutus meidän toimintamme sujuvuuteen. Ihan jo lähtien siitä, että henkilökohtaisessa kontaktissa lääkäri voi paremmin ja rauhassa keskittyä potilaan asioihin, jos hänen ei tarvitse tuskailla tietotekniikan kanssa.

Esimerkiksi näistä syistä haluamme, että Suomeen on kasvamassa koodaajasukupolvi. Haluamme vaikuttaa sekä tehdyn työn laatuun että osaajien määrään.”

MUITA 7–9-LUOKILLE SOPIVIA RESURSSEJA

	W3 SCHOOLS	UDACITY	CODECADEMY
Lyhyesti	Verkon sanakirja. Sisältää HTML-, CSS-, JavaScript-, XML-, PHP- ja SQL-termejä ja niihin liittyviä interaktiivisia koodipätkiä.	Useita amerikkalaisten yliopiston ohjelmointikursseja, joita seurataan videoiden ja harjoitusten kautta.	Interaktiivisia selaimessa suoritettavia harjoituksia. Kielivaihtoehtoina HTML, CSS, JavaScript, Python, Ruby ja PHP. Sopii itseohjautuvalle.
Luokka	6–9+	9+	6–9+
Valmistauminen	Toimii parhaiten lunttilappuna.	Soveltuu erityisen edistyneille itseopiskeluun.	Hyvää opiskelumateriaalia myös opettajalle.
Vaatimukset	Tietokone, internetselain.	Tietokone, internetselain.	Tietokone, internetselain.
Hinta	Ilmainen.	Ilmainen, mutta sertifikaatti maksaa.	Ilmainen.
Kieli	Englanti.	Englanti.	Englanti.
Osoite	w3schools.com	udacity.com	codecademy.com

CODE SCHOOL	EDX	HELSINGIN YLIOPISTON MOOC	MIT OPENCOURSEWARE
Pidemmälle edistyneille syventäviä kursseja JavaScriptin, HTML:n, CSS:n Ruby:n ja iOS:n maailmasta. Oppilaat oppivat videoiden ja selaimessa tehtävien harjoitusten avulla.	Videoita Python-, Ruby-, C++-kielistä sekä koneälystä. Yhdistelmä pdf-tiedostoja, tutoriaaleja ja kysymyssarjoja.	Helsingin yliopisto järjestää avoimia verkkokursseja ohjelmoinnista.	MIT:n ohjelmointikursseja – sisältää erityisesti kehuttuja teoriakursseja, lukumateriaalia, harjoituskysymyksiä sekä luentovideoita.
9+	9+	9+	9+
Soveltuu erityisen edistyneille itseopiskeluun.	Kurssit käynnissä vain tietyn aikaa.	Kurssit käynnissä vain tietyn aikaa.	Soveltuu itseohjautuvalle.
Tietokone, internetselain.	Tietokone, internetselain.	Tietokone, internetselain.	Tietokone, internetselain.
Ilmainen.	Ilmainen.	Ilmainen.	Ilmainen.
Englanti.	Englanti.	Englanti, osassa suomi.	Englanti.
codeschool.com	edx.org	mooc.cs.helsinki.fi	http://ocw.mit.edu/courses/intro-programming/

MUITA 7–9-LUOKILLE SOPIVIA RESURSSEJA

	PROCESSING	COURSERA	CODE AVENGERS
Lyhyesti	Kiinnostava vaihtoehto lapselle, joka on jo kokeillut JavaScriptin perusteet ja kaipaa visuaalisempaa oppimisympäristöä.	JavaScriptin, Pythonin ja tietokantojen perusteet. Perustuu yliopistokursseihin.	Selaimessa pelattava peli, jossa opitaan JavaScriptin, HTML:n ja CSS:n perusteita.
Luokka	6–9+	9+	3–9+
Valmistauminen	Ei etene lineaarisesti, vaatii opetajalta perehtymistä.	Soveltuu itseohjautuvalle.	Opettajan on hyvä pelata itse läpi ensin.
Vaatimukset	Tietokone, internetselain.	Tietokone, internetselain.	Tietokone, internetselain.
Hinta	Ilmainen.	Ilmainen.	Ilmainen.
Kieli	Englanti.	Englanti.	Englanti.
Osoite	processing.org	coursera.org/course/cs101	codeavengers.com

CODE COMBAT	RUBY WARRIOR	DASH	KINESTETHIC LEARNING ACTIVITIES
Selaimessa pelattava seikkailupeli, jossa opitaan JavaScriptin perusteita.	Selaimessa pelattava seikkailupeli, jossa opitaan Rubyn perusteita.	HTML:n ja JavaScriptin perusteet interaktiivisten harjoitusten perusteella.	Leikkejä, jotka opettavat ohjelmoinnin peruskäsitteitä. Soveltuvat myös vanhemmille oppilaille.
6–9	6–9	6–9+	7–9
–	–	Soveltuu myös opettajien itseoppimis-materiaaliksi.	Vaatii valmistautumista – opettajan on hyvä lukea koko harjoituksen kuvaus.
Tietokone, internetselain.	Tietokone, internetselain.	Tietokone, internetselain.	-
Ilmainen.	Ilmainen.	Ilmainen.	Ilmainen.
Englanti.	Englanti.	Englanti.	Ohjeet englanniksi.
codecombat.com	bloc.io/ruby-warrior	dash.generalassemb.ly	cs.ubc.ca/~kla

PÄÄMÄÄRÄ

Kuvaaja Maija Tammi: ”Leikkivä lapsi päättää, mihin juna menee ja millaista rataa pitkin. Samoin tietokoneelle on kerrottava, mitä sen haluaa tekevän. Kuvassa on leikkijunan veturi ja junarataa.”

Sopivaa luokasta riippumatta:

MOBIILISOVELLUKSET

Älypuhelimet ja tabletit ovat tulleet kiinteäksi osaksi arkeamme, ja mobiililaitteille on alkanut löytyä työkaluja myös ohjelmoinnin opettamiseen. Puhelimilla ja tableteilla voi monipuolistaa harjoituksia, mutta yksinään ne eivät vielä opeta ohjelmointia. Monet mobiilisovellukset ovat pulmapelejä, jotka muuttuvat haastavammiksi kierros kierrokselta.

Mobiilisovellukset ovat tuotteina uusia, niiden hinnat vaihtelevat, eikä saatavilla ole yhtä ehdotonta vaihtoehtoa. Siksi olemme nostaneet esille kaksi erilaista sovellusta. Tätä kirjoitettaessa myös aiemmin esitellystä Scratchista on tulossa ScratchJR-niminen tablettiversio.

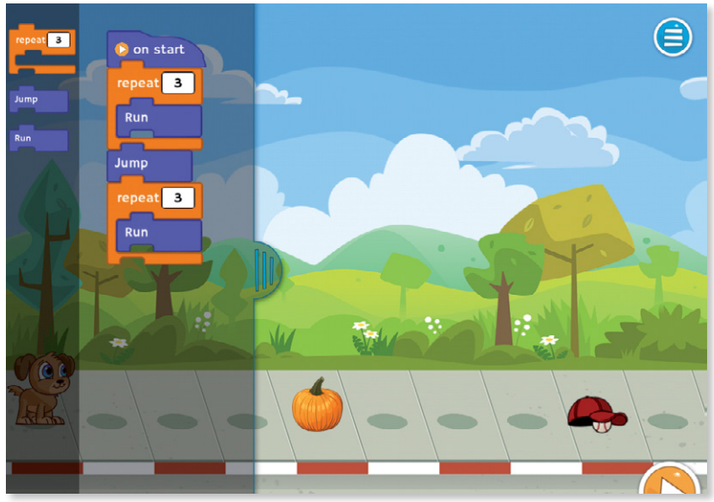
Puhelimilla ja tableteilla voi monipuolistaa harjoituksia. Monet mobiilisovellukset ovat pulmapelejä.

TYNKER

Tynker on Scratchin inspiroima iPad-sovellus, joka opettaa lapselle koodauksen perusteita ongelmapelien avulla. Tynkerin harjoitukset vaikeutuvat kenttä kentältä. Harjoituksissa seikkailevat esimerkiksi koira, jonka pitää ohjelmoida tiensä takaisin kotiin sekä robotti, joka ohjaa avaruuskonetta.

Pelaaminen Tynkerillä on hyvin ohjattua. Käyttöliittymä ja komennot ovat englanniksi, mutta ne ovat hyvin yksinkertaisia.

Tynker sisältää paljon työkaluja opettajia varten. Oppilaita voi jakaa ryhmiin, ja tekemisen tahtia voi seurata oppilastasolla. Erilaisia opintokokonaisuuksia on kuusi, joissa jokaisessa on noin 45 minuuttia sisältöä. Lisää materiaalia saa maksua vastaan. Tynkeristä löytyy myös maksullinen verkkoversio, jota voi pelata iPadin sijaan pöytäkoneen tai kannettavan tietokoneen selaimessa.



Tynker-iPad-sovellus. Koiran on selvitettävä tiensä esteiden yli oppilaan antamilla komennoilla.

Kuvassa esitetyssä tehtävässä lapsen tehtävänä on ohjata koira lakin luo. Vasemmalla olevaan ikkunaan raahataan Scratchista tuttuja komentopalikoita. Matkalla olevia esteitä kuuluu väistää.

Komentopalikoista muodostuva koodi vaikeutuu taso tasolta. Tässä esimerkissä harjoitellaan niin kutsutun silmukan käsitettä eli saman asian toistamista yhä uudelleen. Oranssin ”repeat 3” -palikan sisälle raahataan violetti ”run”-komento. Näiden yhteisvaikutuksesta koira ottaa aina kolme askelta, hyppää esineen yli ja toistaa taas saman kolmen askeleen ja hypyn toiminnon. Kuten edellisen sivun kuvasta näkee, koira päätyy tällöin hyppäämään sekä kurpitsan että lippalakin yli.

Play-nappia painamalla oppilas pääsee kokeilemaan palikoista luomaansa koodia. Ohjelma myös antaa vinkkejä, jos oppilas jumiutuu. Tasoja voi ratkaista eri tavoin.

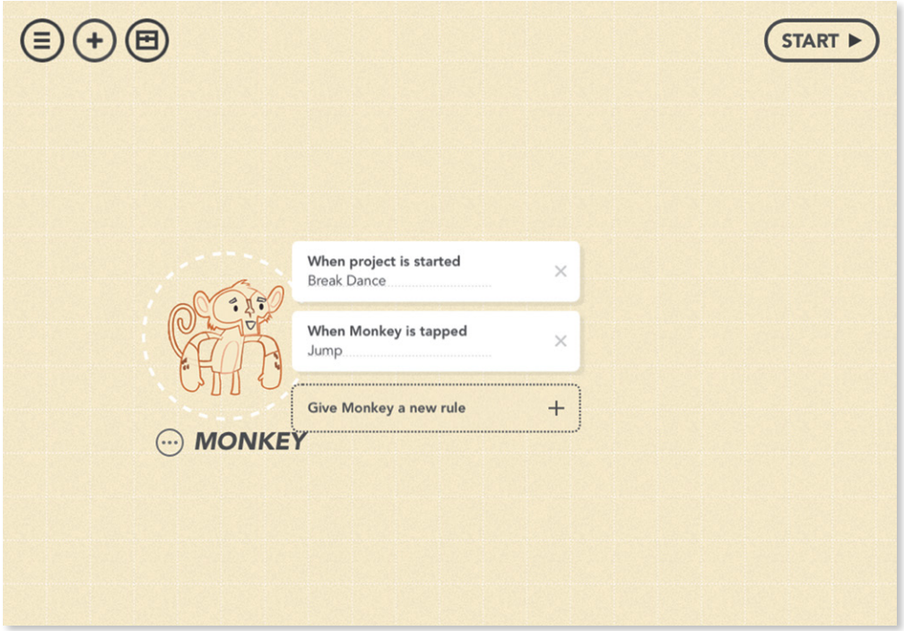
Kun taso on päästy läpi, oppilas näkee muun muassa, kuinka kauan ratkaisemiseen meni, ja olisiko tason voinut ratkaista muilla tavoin.

HOPSCOTCH

Hopscotch on Tynkeriä monipuolisempi iPad-ympäristö, jossa lapsi voi kokeilla visuaalista koodaamista. Hopscotch muistuttaa myös edellä esiteltyä Scratchia.

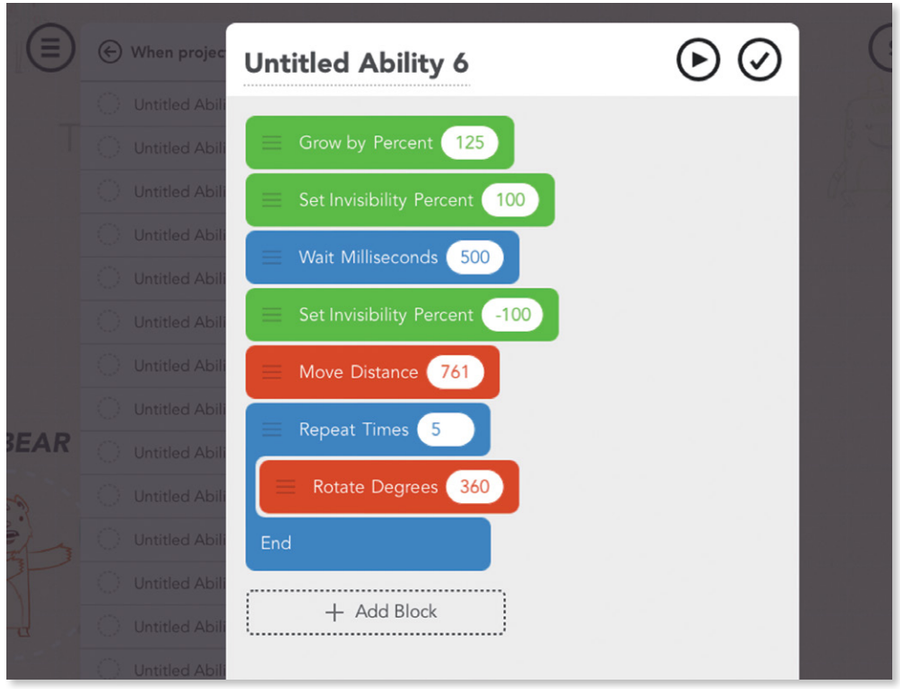
Koodipalikoista muodostuvat projektit voivat olla pelejä, tarinoita tai vaikka taidetta. Muiden rakentamia projekteja voi tutkia ja katsoa, miten ne on rakennettu. Tällä hetkellä Community-osiosta löytyy esimerkiksi peli, jossa eläimet juoksevat kilpaa, kun oppilas tökkää ruutua.

Hopscotch on ilmainen. Sitä varten on myös kirjoitettu opettajan opas, jonka saadakseen on jätettävä sähköpostiosoitteensa. Opettajan oppaaseen sisältyy kalvoja, joissa selitetään Hopscotchin avulla esimerkiksi mitä on ohjelmointi, mitä ovat silmukat, mitä on satunnaisuus ja mitä ovat ehtolauseet.



Hopscotchissa aloitetaan valitsemalla hahmo.

Hopscotchissa jokaiseen hahmoon liittyy sääntöjä. Kuvassa yllä apinaan liitetään sääntö, jonka perusteella se tanssii, kun projekti aloitetaan ja hyppää, kun apinaa tökätään. Lisäksi kaikilla hahmoilla on joukko kyvykkyyksiä, joka ovat ennalta ohjelmoituja koodipätkiä, joita voi käyttää sellaisenaan. Apina voi esimerkiksi tanssia, kasvaa tai leijua. Omia kyvykkyyksiä voi myös tallentaa.



Koodia kirjoitetaan raahamalla Scratchista tuttuja liikkumiseen, piirtämiseen, ulkonäköön ja ehtoihin liittyviä palikoita ohjelmointialueelle. Erilaisia ohjelmapätkiä voi kirjoittaa useampia. Esimerkiksi tässä määritellään mielikuvituksellisesti, että ohjelman alkaessa karhuhahmo (ks. seuraavan sivun kuva) kasvaa 125 prosenttia, katoaa hetkeksi, liikkuu näytöllä 761 kuvapisteen verran ja pyörii 360 asteen ympyrää viisi kertaa.

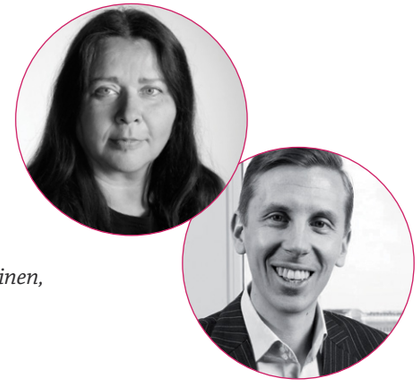
Kun painaa edellisen kuvan oikean ylänurkan Play-nappia, ohjelmakoodin toiminnallisuutta pääsee kokeilemaan käytännössä. Tässä tarinan alku karhuineen.

MUITA MOBIILIRESURSSEJA

	CARGOBOT	KODABLE	HAKITZU
Lyhyesti	Peli. Pelaajat oppivat logiikkaa siirtelemällä kontteja robottikädellä. Tasojen muuttuessa vaikeammiksi pelaajan pitää muodostaa funktioita ja käyttää jäljellä olevat siirrot tehokkaasti.	Kodable on ilmainen iPad-peli, joka opettaa yksinkertaisten logiikkaharjoitusten kautta ohjelmointia. Sopii parhaiten pienille.	Pelaajat oppivat JavaScriptin perusteet ohjaamalla taistelevia robotteja.
Luokka	3–6	1–2	5–9
Valmistauminen	Ohjelman asennus.	Ohjelman asennus.	Opettajan materaali: dropbox.com/s/doj9fug23eymsgf/Hakitzu%20Teachers%20Guide.pdf
Vaatimukset	iPad, iPhone.	iPad, iPhone.	iPad, iPhone, Android-laite.
Hinta	Ilmainen.	Ilmainen.	Ilmainen.
Kieli	Englanti.	Mm. englanti, suomi, ruotsi.	Englanti.
Osoite	twolivesleft.com/CargoBot	kodable.com	kuatostudios.com

MIKSI TÄMÄ ON TÄRKEÄÄ? | **KOODI2016:N TUKIJAT KERTOVAT**

”JOS ET YMMÄRRÄ, MITEN OHJELMISTO TOIMII, OLET AINA HUKASSA”



Mervi Karikorpi, *johtaja, innovaatioympäristö ja uudistuminen, Teknologiateollisuus ry*
Rasmus Roiha*, toimitusjohtaja, Ohjelmistoyrittäjät ry*

MERVI KARIKORPI: ”EU-tasolla 2020:een mennessä nopean kasvun skenaariossa ict-osaamista tarvitaan nettona lähes 1 000 000 uudessa työpaikassa. Meillä tulee siis olemaan hyvin paljon tarvetta perusohjelmisto-osaamiselle.

Mietitään esimerkiksi älykkäitä, keskenään keskustelevia laitteita tai yhä useampien asioiden yhteyksiä verkkomaaillmaan: kaikki nämä asiat luovat uusia työpaikkoja. Ohjelmointitaito on hyvää osaamista maasta ja alasta riippumatta.

Jos Suomessa haluaa yliopistoon opiskelemaan esimerkiksi historiaa tai taideaineita, takana on koko peruskoulu ja tavallisesti lukio aiheesta. Tietojenkäsittelyn ja ohjelmistotekniikan opinnot joutuu aloittamaan nollasta. Se ei ole hyvä asia osaamiseen perustuvassa yhteiskunnassa.”

RASMUS ROIHA: “Ai miksi koodauksen opettaminen peruskoulussa on tärkeää? Koska ohjelmistoja tulee olemaan kaikessa.

Puhutaan älyvaatteista, puhutaan Ponssen älykoneista, peleistä, jääkaapeista, autoista, veneistä, veneen moottoreista. Ohjelmistot ovat melkein kaikessa jo nyt. Jos et ymmärrä, miten ohjelmisto toimii, olet aina hukassa. Oli toimiala sitten jääkaappi- ja pakastinala tai metsäala.

Lisäksi meidän vientiteollisuus koostuu enemmän ja enemmän palveluista ja tuotteista, jotka ovat osittain ohjelmistoa. Se voi olla puhdasta softaa tai se voi olla älytekstiilejä, joissa on pieni ohjelmisto. Suomi elää viennistä, ja se mitä me viemme, se sisältää softaa. Trendi ei tule pienenemään.”

OPEN TARKISTUSLISTA ENSIMMÄISELLE TUNNILLE

Oli kyseessä sitten leikki tai verkkoresurssi, opettajan kannattaa käydä oheinen lista läpi ennen ensimmäistä ohjelmointiharjoitusta.

1. VALITSE SOPIVA HARJOITUS

Valitse sopiva harjoitus tai tutoriaali hyödyntäen esimerkiksi jotain aiemmin tässä osiossa esiteltyä verkkoresurssia tai leikkiä.

Käy harjoitus läpi itse, jotta tunnet sen kulun.

2. VARAA TARVITTAVAT RESURSSIT

Tarkista, että harjoitus / sivusto / tutoriaali toimii oppilaan käytössä olevalla laitteella (mukaan lukien ääni ja kuva).

Jos harjoitus vaatii ääntä, eikä koululla ole tarjota kuulokkeita, pohdi, onko harjoitusta mahdollista käyttää tilassanne, vai tuleeko hälyä liikaa. Oppilaita voi myös pyytää tuomaan omia kuulokkeitaan oppitunnille.

Jos laitteita ei ole riittävästi, ohjaa kaksi tai useampia oppilaita saman laitteen ääreen. Kun oppilaat työskentelevät yhdessä, he auttavat toisiaan ja nojaavat vähemmän opettajaan. He oppivat myös, että ohjelmointi on ihmisten välistä yhteispeliä.

3. SELITÄ OPPILAILLE, MIKSI OHJELMOINTIA OPETELLAAN

Selitä ohjelmoinnin ajatus ja tarkoitus yksinkertaisella tavalla: ihminen käskee tietokonetta, koska kone osaa tehdä monia asioita ihmistä nopeammin ja varmemmin. Käytä esimerkkejä, joista sekä pojat että tytöt välittävät (ihmisten auttaminen ja elämien pelastaminen, ihmisten yhdistäminen toisiinsa, oppilaan käyttämän puhelimen toiminta ja sen sovellukset, pelit, elokuvat, musiikki).

Sano esimerkiksi: ”Ajatelkaa asioita, joita näette joke päivä: Älypuhelin. Mikroaaltouuni. Tabletti. Liikennevalo. Facebook. WhatsApp-viestiohjelma. Kaikki ne voivat toimia vain siksi, että joku on osannut ohjelmoida ne.”

Tai: ”Ohjelmointi auttaa yhdistämään ihmisen luovat ideat ja tietokoneen tarjoamat voimavarat. Ohjelmoijia tarvitaan kehittämään uusia lääkkeitä, tekemään tietokonepelejä, luomaan animaatioelokuvia, rakentamaan robotteja, tutkimaan vieraita planeettoja tai kehittämään itseään ajavia autoja.”

Voit myös käydä läpi sivuilla 26–31 esiteltyjä esimerkkejä siitä, mitä kaikkea ohjelmoija voi luoda maailmaan.

4. OHJAA OPPILAAT HARJOITUKSEN PARIIN

Kirjoita harjoituksen aloittamisen ohjeet näkyville.

Varmista, että kaikki saavat harjoituksen auki.

Pyydä oppilasta kysymään ensin kolmelta kaverilta ja vasta sitten opettajalta.

5. KUN OPPILAS KOHTAA ONGELMAN

Kun oppilaalla on ongelma, voit sanoa: ”Kysy ensin kolmelta luokkakaverilta. Jos heillä ei ole vastausta, kysy sitten minulta.”

Tavalliset pedagogiset keinot toimivat myös ohjelmointiharjoituksissa. Rohkaise opiskelijoita onnistumisista: ”Hienoa Pörri, sinähän onnistuit! Kokeile saatko seuraavankin sujumaan!”

On täysin sallittua vastata oppilaalle: ”Enpä osaa vastata tuohon suoralta kädeltä. Katsotaan ongelmaa yhdessä.”

Voit myös muistuttaa oppilaita: ”Ohjelmoimaan oppiminen on kuin uuden kielen oppimista: kukaan ei osaa kaikkea sujuvasti saman tien.”

6. MITÄ TEHDÄ OPPILAILLE, JOTKA OVAT SELVÄSTI MUITA EDELLÄ?

Oppilaat voivat kokeilla muita harjoituksia joko saman sovelluksen parissa tai muualla. Esimerkkejä sopivista harjoituksista löytyy tämän oppaan sivuilta 98–99 ja 106–109.

Ensimmäisinä harjoituksista selvinneet oppilaat voivat myös auttaa muita.

Lista mukailee Code.org-sivuston opettajille suunnattua Quick Tips -listaa:
code.org/learn *(sivun alalaita)*

MILLOIN OPITAAN TEKEMÄÄN VERKKOSIVUJA?

Oman verkkosivuston tekeminen on monille lapsille innostavaa.

Monet vasta-alkajat luovat verkkosivustoja nykyään valmiita sivustopohjia muokkaamalla. Jos sivut haluaa kuitenkin rakentaa itse tai oppia muokkaamaan valmiita pohjia monipuolisesti, HTML- ja CSS-kielistä on hyötyä.

HTML eli HyperText Markup Language ja CSS eli Cascading Style Sheets eivät ole kirjaimellisesti ohjelmointikieliä, vaan niin kutsuttuja kuvauskieliä.

HTML on merkkauskieli, jota käytetään internetissä julkaistavan verkkosivuston luomiseen. Perusmuodossaan HTML:llä merkitään esimerkiksi, mikä osa verkkosivuston sisällöstä on tekstiä, mikä on otsikkoa, ja mihin kohden tulee kuva tai taulukko.

CSS:ää puolestaan käytetään verkkosivuston ulkoasun muokkaamiseen. CSS:llä voi muuttaa tekstin kirjasintyyppiä, otsikon väriä, sivuston elementtien sijoittelua ja niin edelleen.

Näiden lisäksi hyvin yleinen sivustojen kehittämisessä tarvittava työkalu on JavaScript-kieli, jota käytetään esimerkiksi vuorovaikutusta sivustolle tuovien toiminnallisuuksien luomiseen.

Monille lapsille oman verkkosivuston tekeminen on innostavaa ja helposti ymmärrettävää, koska visuaalista sivustoa luodessa sille tehdyt muutokset näkee heti ruudulla. Siksi verkkotyökalujenkin opettamisen voi aloittaa jo alakoulussa, jos vain aikaa ja intoa riittää.

HTML:ää, CSS:ää ja JavaScriptiä voi hyvin opetella vaikkapa pienoisverkkokursseilla. Näitä tarjoavat esimerkiksi Codecademy (codecademy.com) ja Khan Academy (khanacademy.org).

”OHJELMOINNIN OSAAJALLA ON VALTA”

Kati Korhonen-Yrjänheikki, *yksikönjohtaja, Tekniikan akateemiset TEK*

”Länsimaissa kukaan ei enää kysy, miksi pitää osata lukea ja kirjoittaa. Uskon, että 10 vuoden kuluttua kukaan ei kysy, miksi jo lapsille ja nuorille pitää opettaa ohjelmoinnin perusteet. Sen sijaan kysytään, miksi ohjelmointia tuotiin peruskouluun vasta 2010-luvulla.

Ohjelmoinnin osaajalla on valta. Esimerkiksi avoimen lähdekoodin tapauksessa koodari pystyy tutkailemaan sovelluksen konepellin alle, ottamaan ohjelmiston haltuun ja halutessaan kehittämään sitä edelleen.

Suomessa on noin 2 600 peruskoulua, joissa opiskelee 550 000 peruskoululaista. Suomessa pidetään tärkeänä, että laadukas opetus on tasa-arvoisesti kaikkien saatavilla. Tietotekniikan opetusratkaisut ovat kuitenkin nykyisellään kunta- ja koulukohtaisia ja perustuvat pääosin oppilaan omaan harrastuneisuuteen. Ainoa tapa varmistaa laatu ja tasa-arvo on se, että ohjelmointia aletaan opettaa joka ikisessä peruskoulussa.

Samalla myös opettajia pitää innostaa, ja meidän on kehitettävä opettajien perus- ja täydennyskoulusta. Jokainen luokanopettaja tarvitsee ohjelmointi- ja tietoturvaosaamista. Lisäksi tarvitaan tietotekniikan aineenopettajia, jotka osaavat tietotekniikkaa laaja-alaisesti ja syvällisesti.

Toivon, että kun ohjelmointia opetetaan kouluissa, samalla hyödynnetään tiimityöskentelyä ja vahvistetaan lasten vuorovaikutustaitoja.

Käytännön elämässä yhteistyötä tehdään sekä kasvokkain että virtuaalisesti ja kokoonpanoissa, joissa on hyvin erilaisia ihmisiä. Ohjelmointiryhmissäkin voisi olla eri-ikäisiä oppilaita, ja miksei osa voisi olla myös toisesta koulusta tai maasta.”

OHJELMOINTI JA MUUT OPPIAINEET

Uudessa opetussuunnitelman luonnoksessa ohjelmoinnin opetteluun otetaan aikaa lähtökohtaisesti matematiikan tuntijaosta. Samaan aikaan on ilmeistä, että koska ohjelmoinnilla on sovelluksia sekä tieteessä, taiteessa että esimerkiksi urheilussa, niin sitä voi myös sivuta lähes kaikissa kuviteltavissa olevissa oppiaineissa.

Alla on muutama satunnaisesti valittuihin oppiaineisiin liittyvä esimerkki, jotka toivottavasti herättävät ajatuksia siitä, miten ohjelmointiin ja ohjelmien kehittämiseen liittyviä asioita voidaan ottaa esille innostavalla, arkielämän konkretiaa painottavalla tavalla.

YHTEISKUNTAOPPI

- Yläkoulun yhteiskuntaopissa voidaan perehtyä erilaisiin yhteiskunnallisesti merkittäviin asioihin verkosta löytyvien informaation visualisointien avulla. Esimerkiksi Apps4Finland-kisassa palkitut työt ovat hyvä alku: apps4finland.fi/voittajat-2013. Vaikka oppilaat eivät koodaisi itse, he näkevät konkreettisesti, miten ohjelmointitaitoja ja avointa dataa eli julkisesti saatavilla olevia tietovarantoja yhdistelemällä voi esittää kiinnostavia asioita maailmasta.

- Yhteiskuntaopissa voisi harkita opetussuunnitelman puitteissa teettää koe-esseitä myös tieto- ja viestintätekniikkaan sekä ohjelmointiin liittyvistä asioista:
  - Miten ihmisen rooli työelämässä on muuttunut, kun tietokoneista on tullut yhä ”älykkäämpiä”?
  - Miten internet syntyi? Keitä ihmisiä ja tahoja kehitykseen liittyi?
  - Miten ensimmäiset tietokoneet syntyivät ja kuka niitä käytti?
  - Miten ilmainen tietosanakirja Wikipedia tai verkon musiikin ja elokuvien laittomat latauspalvelut ovat muuttaneet ihmisten elämää?

Matematiikassa voisi keskittyä laskemisen sijasta itse ongelmaan.

MATEMATIIKKA

- Matematiikkaa voisi – uudessa maailmanjärjestyksessä – opiskella tietokoneiden ja ohjelmointitaitojen avulla aivan nykyisestä poikkeavalla tavalla. Siinä keskityttäisiin varsinaisen laskemisen sijasta ongelman ymmärtämiseen ja muotoiluun – ja laskutoimitusosan ratkaisemiseen käytettäisiin tietokonetta. Kun tietokone hoitaa laskut, itse ongelmat voisivat olla laskutoimituksellisesti huomattavasti vaikeampia, ja tämän ansiosta niillä voisi olla yhä useammin selvä, ymmärrettävä ja motivoiva yhteys tosielämään, esimerkiksi biologiaan, maantietoon tai fyysisen maailman ilmiöihin.
  - Matemaatikko **Conrad Wolfram** on pitänyt aiheesta loistavan, huikealla tavalla silmiä avaavan puheen: youtube.com/watch?v=60OVlfAUPJg
  - Wolframin ja kumppaneiden ehdotukset uudenlaiselle, tietokonepohjaiselle matematiikan opetussuunnitelmalla löytyvät osoitteesta computerbased-math.org

LIIKUNTA

- Esimerkiksi alakoulun liikuntatunnilla voidaan leikkiä mitä tahansa yksikäsitteisten ohjeiden antamista vaativaa leikkiä. Tällainen voisi olla vaikkapa esterataleikki, jossa urheilukentälle kootaan renkaista ja tötsistä koostuva reitti. Oppilaiden on ohjeistettava tarkoin komennoin toisensa liikkumaan sen läpi sokkona, ja nopeimman parin aika voittaa.

- Erilaisia leikkejä on esitetty esimerkiksi CS Unplugged -sivustolla (csunplugged.org).

KUVATAIDE

- Kuvataiteessa voidaan kokeilla suunnitella käyttöliittymä omalle mobiilisovellukselle tai piirtää luonnoksia omiksi kotisivuiksi.

- Toinen kuvataiteen tunnilla mahdollinen harjoitus on pyytää oppilasta antamaan toiselle oppilaalle ohjeita tietynlaisen kuvan piirtämiseksi. CS Unplugged -sivustolla on esimerkiksi harjoitus, jossa oppilaan tulee saada toinen oppilas piirtämään suorakaiteen ja rastin sekä oppilaan oman nimen sisältämä kuvio antamalla vain tarkkoja suullisia ohjeita ja näkemättä, mitä toinen piirtää. Toinen ei myöskään saa esittää kysymyksiä.

    - Tämä ja muita harjoituksia löytyy osoitteesta: csunplugged.org/sites/default/files/activity_pdfs_full/unplugged-12-programming_languages.pdf

- Yläkouluikäiset voivat opettaa tietokoneen luomaan taideteoksia Processing-ohjelmointiympäristön avulla: openprocessing.org/classroom/855

MUSIIKKI

- Musiikkitunnilla voidaan perehtyä musiikin tekemiseen tietokoneella – tai ihmetellä esimerkiksi EMI-nimisen tietokoneohjelmiston säveltämiä **Frédéric Chopinia** matkivia masurkoita: artsites.ucsc.edu/faculty/cope/mp3page.htm

KÄSITYÖ

- Käsityötunnilla voidaan puhua siitä, millä tavoin neulominen on itse asiassa algoritmista toimintaa, joka on suhteellisen helppoa opettaa myös tietokoneelle. Tietoja aiheesta: codecademy.com/blog/70-how-knitters-are-human-computers

VIERAAT KIELET

- Vieraissa kielissä voidaan pohtia esimerkiksi sitä, mitä tietokoneelle pitäisi sanoa, jotta se oppisi muodostamaan ruotsin tai ranskan säännöllisen verbin perusmuodosta eri persoonamuotoja. (Vinkki: koneen on opittva lisäämään perusmuotoon sopivia päätteitä.)

MIKSI TÄMÄ ON TÄRKEÄÄ? | **KOODI2016:N TUKIJAT KERTOVAT**

”LAPSET OVAT PORTTI NYKYAIKAISTEN TAITOJEN VIEMISESSÄ KOTEIHIN”

Yrjö Kari-Koskinen, *seniorikonsultti, Futurice*

”Yksi asia, jonka takia yhä useamman olisi hyvä ymmärtää edes perusasioita ohjelmoinnista ja ohjelmistojen kehittämisestä, on yritysten ja julkisen puolen päätöksenteko ja ostot. Nyt niistä budjeteista päättävät usein ihmiset, jotka eivät tunne ohjelmointia ja ohjelmistokehityksen haasteita. Se taas johtaa isoihin ongelmiin, jos ostaja ei oikein tiedä mitä on ostamassa.

Kaikkien ei missään nimessä tarvitse osata koodata, mutta kaikilla olisi hyvä olla perusymmärrys siitä, miten nämä jutut syntyvät. Parhaimmillaan tietotekniikka tukee muuta aineosaamista. Esimerkiksi taideaineissa tietotekniikalla voi saada aikaan päräyttävää kuvataidetta ja musiikkia. Tai sitten koodaamisen opettaminen voi kannustaa muiden aineiden oppimiseen. Oppilas voi tajuta, että hänen pitää opiskella matikkaa, jotta voi luoda siistiä 3D-grafiikkaa.

Lapset ovat myös portti nykyaikaisten taitojen viemisessä koteihin. He laittavat digiboksin asetukset kuntoon jo nyt. Jatkossa lapset voivat olla yhä merkittävämmässä roolissa sinä, miten tietotekninen osaaminen menee koteihin.

Futuricen liiketoiminta on lyhytjänteistä verrattuna peruskoulun opetussuunnitelman tekemiseen. Mutta haluamme rakentaa parempia digipalveluita ja toimivampaa tietoyhteiskuntaa. Ohjelmoinnin opettamisen tukeminen luo yhteiskuntaa, jossa ei tule eriarvoistumista ja lokeroitumista niihin, jotka osaavat digitaitoja ja niihin jotka eivät. Se on sekä Futuricen etu että kaikkien etu.”

Ohjelmoinnin opettaminen ei ole vain peruskoulun asia.
Viimeinen osio paneutuu siihen, mitä seuraavaksi tulisi tehdä.
Talkoisiin tarvitaan vanhemmat, yritykset ja julkkikset.

4 | *Mitä seuraavaksi?*

HAASTEITA SINULLE JA MINULLE

Tätä opasta kirjoittaessamme kaksi johtopäätöstä on vahvistunut mielissämme entisestään:

1) Jokaisella lapsella on oikeus oppia, mitä ohjelmoimalla voi luoda ja rakentaa.

2) Niille, jotka innostuvat aiheesta, on annettava mahdollisuus kehittyä eteenpäin.

Lapsille tulee tarjota arkipäivän elämään, tieteeseen, taiteeseen, vapaa-aikaan ja työhön liittyvä innostava kokonaiskuva siitä, miten teknologia vaikuttaa elämäämme. Tuohon kokonaiskuvaan kuuluu opetus ohjelmoinnin perusteista, mutta myös – tämä oppaan ulkopuolelta – peruskäsitys esimerkiksi siitä, miten toimivat vaikkapa tietokone itse tai maailmanlaajuinen tietoverkko nimeltä internet.

Jotta lapset ja nuoret saisivat parhaat mahdolliset lähtökohdat oppia, muidenkin kuin opettajien ja peruskoulun, Opetus- ja kulttuuriministeriön tai Opetushallituksen on toimittava.

Mukaan tarvitaan rehtorit, kuntien päättäjät, yritysjohtajat, ohjelmoijat, vanhemmat, poliitikot, kirjastonhoitajat ja toimittajat. Sinut ja minut.

Vaikka oppaassa on ylipäänsä ollut ohjelmointia kohtaan myönteinen sävy, tässä viimeisessä osiossa heittäydymme entistäkin puolueellisemmiksi.

Seuraavilla sivuilla esitämme käytännönläheisiä haasteita meille kaikille.

VAIKUTTAJAT, PEREHTYKÄÄ OHJELMOINTIIN

KENELLE: Eduskunta, ammattiliittopomot jne.
HAASTE: Ohjelmointi on ala, jossa Suomella on hyvät edellytykset pärjätä riippumatta luonnonvaroistamme tai sijainnistamme. Hyvää koodia on ilmaista monistaa ja levittää, joten kaikki on sen tekijöistä kiinni.

Poliitikot ja ay-väki voivat puhua ohjelmoinnin ja teknologian tärkeydestä kansantaloudellemme. He voivat vaikuttaa siihen, minkälaiset työt jäävät Suomeen ja myös minkälaisia osaajia tänne muuttaa. Unohdetaan hetkeksi valitus siitä, että kehittyvä automatisaatio vie työpaikkoja ja muistetaan, että ohjelmistoja tuntevan työvoiman kysyntä kasvaa toimialalla kuin toimialalla.

Tätä kaikkea varten on tärkeää, että päättäjät ymmärtävät, mistä puhuvat. Ei ole häpeä myöntää, ettei ymmärrä ohjelmointia, jos ei ole koskaan ohjelmoinut.

Niinpä esimerkiksi kansanedustajat: lukekaa tämä opas, tilatkaa eduskuntaan ihminen pitämään päivän kestävä pikajohdatus ohjelmointiin – mitä tahansa, mikä saa teidät sinuiksi asian kanssa.

INFORMAATION VISUALISOIJAT, LYÖKÄÄ FAKTAT TISKIIN

KENELLE: Tiedonkäsittelyn ja visualisoinnin ammattilaiset ja harrastajat.
HAASTE: Hacks and hackers -porukka, informaation visualisointivelhot, graafikot, data-analyytikot, yhteiskuntatieteilijät ja Excel-gurut: kerätkää yhteen kansantalouden tasolla olennaiset avainluvut ja esittäkää ne vastaan sanomattomassa, helppolukuisessa ja visuaalisesti päräyttävässä infografiikassa:

Ei ole häpeä myöntää, ettei ymmärrä ohjelmointia, jos ei ole ohjelmoinut.

Miten paljon ohjelmistoalan työpaikkoja syntyy sekä vanhoihin, että uusiin yrityksiin? Kuinka nopeasti tarve kasvaa?

Kuinka paljon tietojenkäsittelytieteiden opiskelijoita koulutetaan? Entä tietojenkäsittelytieteen opettajia? Vastaammeko kysyntään?

Miten hyvän ohjelmoijan palkkakehitys muuttuu – ja kannattaisiko ihmisten innostua alasta enemmän?

Miten paljon naisia on koodareina verrattuna miehiin? Miten monissa kouluissa tarjotaan haastavia ohjelmoinnin valinnaiskokonaisuuksia?

Yksi esimerkki tavoitteesta lopputulemaksi löytyy osoitteesta code.org/stats.

VIESTIMET, ESITELKÄÄ ROOLIMALLEJA

KENELLE: Media, vanhemmat, opinto-ohjaajat.
HAASTE: Jostain syystä huomattavasti useampi

meistä tietää, kuka suunnitteli Aalto-maljakon kuin sen, kenen vetämä tiimi on koodannut miljoonien käyttämän Dropbox-palvelun (**Drew Houston**) tai suunnitellut kymmenien miljoonien pelaaman Clash of Clans -pelin (**Lassi Leppinen**).

Olisiko pahitteeksi, jos suomalaiset lapset ja nuoret saisivat useammin kasvot ihmisille, jotka rakentavat arkipäivämme sähköisiä tuotteita ja palveluita ja vaikuttavat samalla miljoonien elämään?

Koodarista on mahdollista kirjoittaa yhtä eläväisesti ja kiinnostavasti kuin muusikosta, näyttelijästä tai urheilijasta (sen näkee esimerkiksi Wired-nimisestä lehdestä).

Sinä toimittaja: myy esimiehellesi juttuidea paikkakuntasi guruohjelmoijasta – tai vaikka kokonainen juttusarja suomalaisen koodin sankareista. (Yksi heistä on seuraavalla sivulla.)

JULKKIKSET, PUHUKAA KOODAUKSESTA

KENELLE: Julkisuuden henkilöt.
HAASTE:
”Haluaisin, että ohjelmointi olisi peruskouluissa aivan yhtä tärkeä aine kuin biologia, kemia, fysiikka ja vastaavat. Jos haluamme kiihdyttää työllisyyden kasvua, meidän täytyy ymmärtää missä töitä on ja missä talouskasvu on mahdollista.”
– Ashton Kutcher, näyttelijä

”Käytäntömme Facebookilla on palkata kirjaimellisesti niin paljon lahjakkaita koodareita kuin löydämme. Markkinoilla ei yksinkertaisesti ole riittävästi tarvittavilla taidoilla varustettuja ihmisiä.”
– Mark Zuckerberg, Facebookin toimitusjohtaja ja perustaja, maailman nuorin miljardööri

”Ohjelmointi on hyvin tärkeää, kun ajattelee tulevaisuutta ja sitä, mihin asiat ovat menossa. Kun älypuhelimia ja tabletteja on yhä enemmän, ja yhä useammat pääsevät käsiksi kaikkeen olemassa olevaan tietoon, mielestäni on hyvin tärkeää oppia ohjelmoinnin kieltä.”
– Chris Bosh, All-Star-tason NBA-pelaaja, Miami Heat

Yhdysvalloissa lukuisat koko maan tuntemat nimet puhuvat ohjelmointiosaamisesta. Sama jeesaisi Suomessakin.

Cheek, **Eva Wahlström**, **Tomi Björck**, **Lenita Airisto**, Robin, **Jari Tervo** – tarttukaa toimeen!

Cheek, Lenita Airisto ja Tomi Björck, tarttukaa toimeen!

Linus Torvalds. Kuva: Linuxmag.com.

PERUSTAKAA LIPUTUSPÄIVÄ LINUSILLE

KENELLE: Kuka ikinä päättääkään liputuspäivistä, media.
HAASTE: Briteissä yhä useampi tietää, keitä ovat tai olivat uraa uurtava tietojenkäsittelytieteilijä **Alan Turing** tai world wide webin keksijä **Sir Tim Berners-Lee**.

Britit kertovat eurooppalaisena kansakuntana tarkoituksella ja tietoisesti tarinaa omasta it-osaamisestaan näiden nimien kautta, koska nimet merkitsevät.

Suomessa meillä olisi oman häntämme nostamisen tarkoitukseen tarjolla loistava nimi: **Linus Torvalds**.

Torvalds on yhden maailman suosituimmista käyttöjärjestelmistä, Linuxin, isä ja kehittäjä.

Hyvin konkreettinen tapa muistaa Torvaldsia olisi perustaa hänelle oma liputuspäivä. Esimerkiksi hänen syntymäpäivänsä on joulukuun 28. päivä.

Suomalaiset tarvitsevat muutenkin oman ohjelmointihistoriansa selittämistä suurelle yleisölle.

Harva it-asioista tietävien piirien ulkopuolinen esimerkiksi hahmottaa, kuinka paljon Suomessa on luotu maailman mittakaavassa aivan keskeisiä avoimen lähdekoodin ohjelmistoja kuten MySQL-tietokanta, SSH-tiedonsalausprotokolla, IRC-verkkokeskusteluympäristö tai versionhallintajärjestelmä Git (jonka isä on muuten myös Linus Torvalds).

KOODARIT, OPETTAKAA OPETTAJIA

KENELLE: Ohjelmointia osaavat opettajat, yritysten koodarit, luokan osaavat oppilaat.
HAASTE: Opettajia voi kouluttaa muutenkin kuin veso-päivinä tai OPH:n koulutuksissa. Harkitkaa esimerkiksi seuraavia:

SOVELTAMINEN

Kuvaaja Maija Tammi:
”Kuvassa on kana. Aiemmista kuvista opitun kaavan soveltaminen nopeuttaa kuvan esittämän asian hahmottamista.”

- Pitäkää tehokoulutuspäivä, jonka aikana koulun ohjelmoinnista ymmärtävät opettajat opettavat muita.
- Luokan parhaat oppilaat voi nimittää ohjelmointitukihenkilöiksi, jotka neuvovat muita ja jopa opettajaa vaikeissa paikoissa.
- Opettaja voi oppia toiselta opettajalta myös samanaikaisopetuksen avulla. Tällöin koulutettava opettaja kuuntelee ja oppii, kun toinen opettaja opettaa luokkaa.
- Jos peruskoulun yhteydessä on lukio, osaavat lukiolaiset voivat opettaa peruskoulun opettajia mahdollisesti kurssipisteitä vastaan tai vapaaehtoistyönä.
- Tiedustelkaa it-alan yrityksiltä tai it-alan firmoissa työskenteleviltä oppilaiden vanhemmilta, olisiko heillä kiinnostusta tulla pitämään koulunne opettajille tehokoulutussessio.
- Opinto-ohjaajilla on iso rooli peruskoululaisten ammatin ja jatkokoulutuksen valinnassa. Varmistakaa, että opoilla on ajantasaista, kiinnostavaa ja monipuolista materiaalia siitä, miten ohjelmointi liittyy nykyaikaisiin töihin.

KOODARIT, JÄRJESTÄKÄÄ ILTAPÄIVÄKERHOJA

KENELLE: Ohjelmointia osaavat opettajat, vanhemmat ja muut aikuiset.

HAASTE: Kaiken oppimisen ei tarvitse tapahtua luokkahuoneessa ja kouluajalla. Vapaaehtoisvoimin järjestettävät iltapäiväkerhot ja suomalaisten yritysten väen järjestämien ilmaisten Koodikoulujen (koodikoulu.fi) tapainen toiminta ovat aivan ratkaisevassa roolissa viimeistään siinä vaiheessa, kun ohjelmoinnista innostuvat oppilaat haluavat kehittyä pidemmälle, kuin mihin oppitunnin aikana on mahdollisuus.

Esimerkiksi ohjelmoija Marko Klemetti (ks. sivu 139) on käynyt pitämässä Espoon Leppävaarassa alakoululaisille koodauskerhoa loistavin ja innostavin tuloksin. Tällaista tarvitaan lisää. Eikä tarvitse edes odottaa syksyyn 2016.

REHTORIT, KOKEILKAA KOODAUSPÄIVÄÄ

KENELLE: Rehtorit ja opettajat,

HAASTE: Jos koulunne haluaa aloittaa ohjelmoinnista innostumisen selkeällä startilla, voitte järjestää ohjelmointiteemapäivän kullekin luokalle. Tämän voi tehdä jo hyvissä ajoin ennen syksyä 2016 ja siten pilotoida ohjelmoinnin opettamista. Koodauspäivän aikana saadaan kerralla ohjelmointikylpy, joka saattaa jättää paremman ja mieleenpainuvamman jäljen kuin yksittäiset opetussessiot.

Teemapäivään voidaan kutsua vieraileviksi opettajiksi myös ohjelmointia osaavia vanhempia.

Rehtorit, järjestäkää koulussanne koodauspäivä jo ennen syksyä 2016!

Resursseja ja ohjeita suomeksi löytyy esimerkiksi osoitteesta koodaustunti.fi sekä tämän oppaan sivuilta osoitteesta koodi2016.fi/koodauspaiva.

OPPILAAT, OPETTAKAA TOISIANNE

KENELLE: Oppilaille (vaatii opettajien aloitteen).

HAASTE: Jos ette ole vielä niin tehneet, lanseeratkaa alakouluunne kummioppilaiden parit: yksi nelosluokkalainen kutakin ekaluokkalaista kohden. Kummiparit ovat yhteydessä kolmen vuoden ajan, kunnes kolmoselta neloselle siirtyvästä tulee opettajakummi uudelle ekaluokkalaisella ja kuudesluokkalainen siirtyy yläkouluun.

Vanhempi kummioppilas auttaa nuorempaa alkuun ohjelmoinnin kanssa (muun muassa).

ENGLANNIN OSAAJAT, KÄÄNTÄKÄÄ OPPIMATERIAALEJA SUOMEKSI

KENELLE: Sinulle, joka osaat englantia ja suomea.

HAASTE: Kuten tämän oppaan edellisestä luvusta käy ilmi, internetissä on valtava määrä ohjelmoinnin opettamiseen tarkoitettuja valmiita ja ilmaisia oppimateriaaleja. Suuri osa näistä resursseista on kuitenkin olemassa toistaiseksi vain englanniksi.

Tästä huolimatta palvelun tarjoaja on usein luonut halukkaille työkalut kääntää materiaaleja helposti omalle kielelleen siten, että ne toimivat yhä suoraan sivustolla. Eräs tällaisen mahdollisuuden tarjoaja on esimerkiksi JavaScript-ohjelmointikieltä opettava verkkokoulu Khan Academy.

Siispä: kerätkää porukka ja kääntäkää talkoovoimin vaikkapa yhden viikonlopun aikana kasa Khan Academyn harjoituksia suomeksi. Lisätietoja: khanacademy.org.

OPPIMATERIAALIN TUOTTAJAT, TEHKÄÄ TYÖNNE

KENELLE: Oppimateriaalien tuottajat ja kustantajat.

HAASTE: Kiinnostus lasten ja nuorten ohjelmointiopetusta kohtaan kasvaa jatkuvasti, mikä näkyy niin mediassa kuin päättäjien ja yrityspomojen puheissa.

Ohjelmointi tulee Suomessa peruskoulun opetussuunnitelmaan syksyllä 2016.

Kysyntä lapsille ja nuorille suunnatuille suomenkielisille ohjelmointioppikirjoille ja muille materiaa-

leille tulee kasvamaan vääjäämättä.

Maksullisten oppimateriaalien tuottajat: kai teitä raha kiinnostaa?

(Ja opettaja: Onko sinulla pöytälaatikossa hyväksi havaitsemaasi oppimateriaalia esimerkiksi siitä, miten algoritmin käsite kannattaa opettaa? Hyvä – testatulle materiaalille on varmasti tilausta.)

Hei oppimateriaalien tuottajat: kai teitä raha kiinnostaa?

VANHEMMAT, VAATIKAA KOULUILTA OHJELMOINNIN OPETUSTA

KENELLE: kaikki vanhemmat, joilla on lapsia peruskoulussa.
HAASTE: Ohjelmointi tulee opetussuunnitelmaan, mutta sen opettaminen tulee olemaan kiinni yksittäisistä kouluista, rehtoreista ja opettajista.

Suomen järjestelmässä ei tulla mitenkään erityisesti tarkkailemaan sitä, kuinka vakavasti rehtorit ja opettajat ottavat tämän uuden vastuun. Luotto on siis kova.

Jos kuitenkin näyttää siltä, että juuri oman lapsen koulussa ei tartutakaan tuumasta toimeen ohjelmointiharjoitusten kanssa, vanhempien on hyvä ottaa kohteliaasti asia esille koulun kanssa. (Aivan äärimmäisessä tapauksessa vanhemmilla on oikeus valittaa opetuksen puutteesta aluehallintoviranomaisille, mutta lähtekäämme siitä, että ystävällinen vuoropuhelu on paras keino.)

Mitä enemmän koko Suomen vanhemmat esittävät toiveensa siitä, että haluamme lastemme oppivan ohjelmoinnin perusteet, sitä helpommin muutoksesta tulee onnistunut ja innostunut.

”OLEN ITSE PITÄNYT LAPSILLE KOODAUSKERHOA”

Marko Klemetti, *DevOps-yksikön johtaja, Eficode*

”Eficode haluaa palkata ohjelmoijiksi nuoria ja motivoituneita, lupaavia opiskelijoita. Tilanne suosii nyt niitä, joille vaikkapa faija on jo lapsena opettanut koodamista. Kun ohjelmointia aletaan opettaa peruskoulussa, se tasa-arvoistaa lapsia ja antaa mahdollisuuden monipuoliselle kehittymiselle koodauksen kautta.

Olen itse pitänyt tänä keväänä Espoossa Perkkaanpuiston koulussa vapaaehtoista koodauskerhoa kolmos–kutosluokkalaisille. Kerhoon piti alun perin ottaa 10 oppilasta, mutta sinne ilmoittautui 15. Lopulta vielä kolme lasta pyysi ekana päivänä itku silmässä kerhon oven takana, että päästäisikö mekin mukaan.

Kaksi ekaa tuntia keskityttiin yksinkertaisessa ohjelmointiympäristössä ihan vain siihen, miten kilpikonnan saa liikkumaan ruudulla. Sitä kautta käytiin läpi ohjelmointiin liittyviä asioita kuten komentoja, silmukoita ja funktioita.

Esimerkiksi funktiolle toimivin selitys oli, että sille asialle, mitä se kilppari tekee ruudulla annetaan jokin nimi, jotta voit pyytää sitä tekemään saman asian helposti myöhemmin uudestaan. Kun olet saanut kerran kilpparin liikkumaan neliökuviossa, niin kun kirjoitat ruudulle uudestaan sanan ”neliö”, niin kilppari tekee taas neliön.

Kerhossa on nähnyt, että nelos–vitosilla on jo matematiikan, englannin kielen ja tietokonetaitojen takia nuorempia paremmat valmiudet. Niiden kanssa päästiin neljännellä kerralla niin pitkälle, että kirjoitettiin jo tasohyppelypelin alkua.

Kolme varttia on aika maksimikeskittymisaika alakoululaiselle! Mutta siinä ajassa lapset sisäistävät valtavan paljon enemmän tietoa kuin aikuiset.”

LOPUKSI

Ohjelmointi tulee peruskouluihin, ja ensimmäiseksi pitää sanoa, että siitä selvitään ihan varmasti.

Toiseksi pitää sanoa, että tavallisen rehtorin tai opettajan näkökulmasta myös haasteet ovat ilmeisiä: ohjelmointia osaavia kollegoja ei ole kiusaksi saakka, testattua suomenkielistä materiaalia on vähän, ja tarjolla on vain vähän parhaita käytäntöjä.

Kokonaisen ikäluokan opettamisesta ei ole kokemuksia, joten kaikkeen liittyy arvailua.

Pohdittavaa on myös siinä, miten koulukohtaisesta opetussuunnitelmasta saa yhtenäisen, järjestelmällisesti etenevän kokonaisuuden, josta on iloa ja hyötyä kaikentasoisille opiskelijoille.

Tämän oppaan tarkoitus on antaa peruskäsitys ohjelmoinnista, sen merkityksestä ja mahdollisuuksista. Olemme hahmotelleet myös, miten ohjelmointia voi opettaa eri luokka-asteilla ilman valtaisia opettajien koulutusohjelmia.

Opas ei ole opetussuunnitelma, mutta toivottavasti se toimii inspiraationa koulujen omien suunnitelmien laatimiseen.

Jokaista ongelmaa ei voi ratkaista etukäteen, ja jostain on aloitettava.

Nyt pallo on päättäjillä, rehtoreilla ja opettajilla – sekä vanhemmilla ja oppilailla.

Jokaista ongelmaa ei voi ratkaista etukäteen, ja jostain on aloitettava.

Kokemus ja tätä kirjaa varten tekemämme koodareiden haastattelut kertovat, että suurin osa taitavista ohjelmoijista on innostunut asiasta nuorena. Ihminen on ilostunut ensin jostain pienestä, ja sitten koodaus on vienyt mennessään.

Siksikin keskeistä on selkeä tahtotila: lapsilla on oikeus ymmärtää, miten tietokoneen avulla rakennetaan asioita ja luodaan uutta maailmaa.

Jos olemme siitä samaa mieltä, seuraavat askeleet on helppo ottaa.

Suunta on yhteinen.

KOKEILE KOODAUSPÄIVÄÄ KOULUSSASI JO NYT!

Haluatko päästä kokeilemaan ohjelmoinnin opettamista jo nyt? Hienoa, sillä ei ole mitään syytä odottaa syksyyn 2016 saakka.

Osoitteessa Koodi2016.fi/koodauspaiva on esitetty valmis ohjelma ja harjoitukset mahdollisiksi koodauspäiviksi 1–2-, 3–6- ja 7–9-luokille. A4-paperille helposti tulostettavien ohjeiden avulla voi järjestää koodauspäivän, vaikka ei olisi koskaan itse koodannut riviäkään.

Päivien tavoitteena ei ole antaa kaiken kattavaa kuvaa ohjelmoinnista vaan kannustaa tutkimiseen ja oppimiseen. Saa kokeilla!

» Koodi2016.fi/koodauspaiva

OHJELMOINTI TULEE peruskoulun opetussuunnitelmaan syksyllä 2016.
Koodi2016 on opettajille ja opetusalan päättäjille suunnattu opas.
Se puhuu tiiviisti ja selkokielisesti kolmesta asiasta:

- **Mitä ohjelmointi on, ja miksi se on tärkeää?**
- **Miten opetussuunnitelma muuttuu ja miksi?**
- **Miten ohjelmointia voi opettaa peruskoulussa?**

Koodi2016 ei ole oppimateriaali vaan kiertoajelu.
Oppaan luettuaan opettajalla on käsitys siitä, miten hän voi lähestyä ohjelmoinnin opettamista – ja olo, että tästä kyllä selvitään.

ILMAINEN MATERIAALI on kokonaan myös osoitteessa Koodi2016.fi.

KOODI2016:N KIRJOITTAJISTA

JUHANI MYKKÄNEN *(s. 1984) on it-alan diplomi-insinööri, luovan alan yrittäjä ja vapaa toimittaja. Hän on työskennellyt muun muassa Helsingin Sanomien Nyt-liitteen sekä Radio Helsingin esimiehenä ja HS:n internetin ja uuden tekniikan kolumnistina.*

LINDA LIUKAS *(s. 1986) on perustanut ohjelmointia naisille opettavan Rails Girls -liikkeen, joka on levinnyt yli 200 kaupunkiin ympäri maailmaa. Hän on työskennellyt ohjelmoinnin verkkokursseja tarjoavassa Codecademyssä ja kirjoittaa ohjelmointia lapsille opettavaa* Hello Ruby *-kirjaa.*