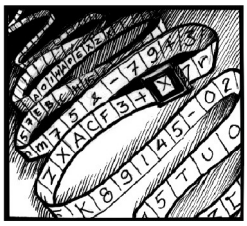

Peruskoulun ohjelmointiopetus

Juha Taina
Matemaattis-luonnontieteellinen tiedekunta, Helsingin yliopisto

Osaatko sanoa, mitä seuraava koodi tekee?

```
L1:     LDX #MSG
L2:     LDA #I
        BEQ L3
        BSR PRINT
        DECA
        STA #I
        JMP L2
L3:     RTS

MSG     FCC 'Hello, World!'
        FCB 0
```

Mahdollisesti. Osaatko kirjoittaa vastaavaa koodia? Mahdollisesti. Tarvitseeko sinun kirjoittaa vastaavaa koodia? Luultavasti ei.

Edellinen esimerkki on Motorolan 6809-prosessorin symbolista konekieltä. Minä opettelin 1980-luvulla ohjelmoinnin perusteet sillä kielellä, mutta en oppinut tekemään ohjelmistoja. 6809-assemblerin aikoina ohjelmistojen teon työkalut olivat vaikeita oppia ja käyttää, mutta laitteet olivat (melko) yksinkertaisia ohjelmoida ja käytössä olleet järjestelmät eivät juuri vaatineet erityisosaamista.

Tänään ohjelmistojen teon työkalut ovat selkeitä ja teoria hallittua, mutta laitteet ja järjestelmät ovat monin verroin 6809-aikaa monimutkaisempia, puhumattakaan kaikista tarpeellisista apuohjelmistoista, kirjastoista ja rajapinnoista. Vielä kun tarkastellaan modernia tapaa tehdä ohjelmistoja, saadaan mukaan ainakin ajanhallintaa, vaatimusten määrittelyä, muutostenhallintaa, eri tasoista suunnittelua ja suunnittelumalleja, useaa eri tyyppistä testausta, elinkaariajattelua, vihreitä arvoja ja henkilöhallintaa. Vähemmästäkin menee pyörälle päästään. Ohjelmistojen teko on nyt ja oli 6809-aikana vaikeaa, ja laadukkaiden ohjelmistojen teko on erityisen vaikeaa.

Ymmärrän, että peruskoulussa tavoitteena on enemmän esitellä ohjelmointia kuin tehdä oppilaista ohjelmistojen tekijöitä. Tämä on hyvä asia, sillä jokaisen tulisi ainakin jossain määrin ymmärtää, mitä tietojärjestelmän sisällä tapahtuu. Kuitenkin ilman selkeää teoreettista pohjaa ja pintaa syvemmälle menevää opetusta ohjelmistojen teosta ja toiminnasta käsiteltävä asia jää helposti irrallisten faktojen ulkoa opetteluksi ja valmiiden osien mekaaniseksi yhdistämiseksi. Ohjelmistojen teon opettamiseen tarvitaan enemmän kuin ohjelmoinnin opettamista.

Opettajalla on suurin rooli opetuksen onnistumisessa. Mitä tapahtuu, kun opettaja saa opetettavaksi uuden alueen, jota hän ei hallitse, joka ei kiinnosta häntä ja jossa osa oppilaista tietää selvästi häntä enemmän? Opettaja ahdistuu ja oppilaat turhautuvat. Ahdistunut opettaja ei jaksa innostua oppilaiden oppimisesta, ja turhautunut oppilas viettää mieluummin aikaa pelaamalla tai sosiaalisessa mediassa kuin seuraamassa hänelle triviaalia opetusta. Pahimmassa tapauksessa ohjelmoinnin opetuksesta tulee välttämätön paha, joka

ei hyödytä ketään ja vain vie tunteja muilta aineilta.

On kuitenkin alue, johon peruskoulussa annettava ohjelmointiopetus sopii hyvin: ryhmätyöskentely ja ryhmätyötaitojen opettelu. Nykyaikainen ohjelmistokehitys vaatii ryhmätyöskentelyä, ja tätä on mahdollista harjoitella ohjelmistojen teon ohessa. Yhdessä tekeminen ei synny itsestään vaan vaatii ohjausta ja harjoittelua. Yhteisen ohjelmiston teko on erinomainen tapa harjoitella ryhmätyöskentelyä ja vastuun ottamista.

Ryhmätyöskentelyn voi ottaa mukaan ohjelmistojen teon opetukseen alusta alkaen. Opiskelijoiden motivaatio paranee, kun he saavat pohtia ohjelmistojen tekoon liittyvää teoriaa, ongelmia ja ratkaisuja yhdessä. Sopivilla aktivoivilla pienillä harjoituksilla opiskelijoiden mielenkiinto saadaan pysymään yllä, ja samalla saadaan opetettua ohjelmoinnin lisäksi ainakin suunnittelua ja testausta. Näin oppilaat saavat kokonaiskuvan ohjelmistojen teosta sekä ennen kaikkea kokemusta yhdessä oppimisesta, yhteistyöstä ja ryhmätyöskentelystä.

Oikean ohjelmointikielen käyttö ohjelmistojen teossa ei ole välttämätön, vaan hyvin suunnittelemalla ohjelmistojen tekoa on mahdollista opetella ja harjoitella valmiiden ohjelman osien avulla. Opettajan ei tarvitse olla ohjelmoinnin ammattilainen, vaan ohjelmointitaitoa tärkeämpiä ominaisuuksia hänellä ovat ryhmätyötaidot, ryhmädynamiikan hallinta ja suurten kokonaisuuksien ymmärtäminen.

Eniten oppilaat hyötyvät opetuksesta, jos he saavat käyttää mielikuvitustaan, tehdä konkreettisia asioita ja nähdä töidensä tulokset. Niinpä heti teorian alusta asti on tärkeää, että oppilaat pohtivat ja tekevät asioita yhdessä. Esimerkiksi kumpi seuraavista tehtävistä luultavasti motivoisi oppilaita enemmän: ”Minkälaisia tietokoneita on sinun elämässäsi ja mitä kaikkea niillä tehdään?” vai ”Minkälainen tietokoneohjelma auttaisi sinua selviämään matematiikan läksyistä? Mitä kaikkea ohjelma voisi osata ja mitä sinun pitäisi tehdä itse?” Entä ”Miksi tietokoneohjelmat toimivat joskus väärin? Minkälaisia kokemuksia sinulla on väärin toimivista ohjelmista?” vai ”Pekan mopossa on tietokoneohjelma, joka huolehtii kaasusta, jarrusta ja vilkuista. Mistä tiedät, että Pekan mopon tietokoneohjelma toimii oikein ja miten varmistat asian?”

Kaiken kaikkiaan ohjelmistojen teon opetus peruskoulussa olisi oikein toteutettuna erittäin hyvä asia. Opetuksesta ja oppimisesta on mahdollista tehdä erittäin palkitsevaa sekä oppilaille että opettajille, mutta ei ilman työtä ja huolellista suunnittelua. Toivotaan, että kaikki tulee menemään parhain päin.

Tehtäviä Euroopan tyttöjen matematiikkaolympialaisista 14.–20.4.2015

1. Olkoon $\triangle ABC$ teräväkulmainen kolmio, ja olkoon sen pisteestä C piirretyn korkeusjanan kantapiste D. Kulman $\angle ABC$ puolittaja leikkaa suoraa CD pisteessä E ja kolmion $\triangle ADE$ ympäripiirrettyä ympyrää ω pisteessä F. Jos $\angle ADF = 45^\circ$, niin osoita, että CF sivuaa ympyrää ω.

2. Domino on 2×1 tai 1×2 -laatta. Selvitä kuinka monella eri tavalla n^2 dominoa voi asettaa $2n \times 2n$ -shakkilaudalle ilman päällekkäisyyksiä niin, että jokainen 2×2 -neliö sisältää ainakin kaksi peittämätöntä ruutua, jotka ovat samalla rivillä tai samalla sarakkeella.

3. Olkoot n ja m kokonaislukuja ja suurempia kuin 1, ja olkoot $a_1, a_2, \ldots, a_m$ positiivisia kokonaislukuja, jotka eivät ole isompia kuin n^m. Osoita, että on olemassa positiiviset kokonaisluvut $b_1, b_2, \ldots, b_m$, jotka eivät ole isompia kuin n, ja joille
$$\text{syt}(a_1 + b_1, a_2 + b_2, \ldots, a_m + b_m) < n,$$
missä $\text{syt}(x_1, x_2, \ldots, x_m)$ tarkoittaa lukujen $x_1, x_2, \ldots, x_m$ suurinta yhteistä tekijää.