TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Marge Käis

Ülevaade õppimisstrateegiatest ja õpetamis-viisidest programmeerimise algõppes

Bakalaureusetöö (9 EAP)

Juhendaja: PhD Reimo Palm

Tartu 2019

Ülevaade õppimisstrateegiatest ja õpetamisviisidest programmeerimise algõppes

Lühikokkuvõte:

Programmeerimist peetakse tänapäeval üheks olulisemaks, kuid ka keerulisemini omandatavaks oskuseks. Selle töö eesmärgiks on anda ülevaade programmeerimise algõppest nii õppimise kui ka õpetamise perspektiivist, tuvastada peamised õppimisel esinevad raskused ja edukust mõjutavad faktorid ning anda soovitusi sissejuhatavate programmeerimiskursuste parendamiseks.

Võtmesõnad:

Programmeerimise algõpe, programmeerimise didaktika

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika, S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

An Overview of the Learning and Teaching Strategies in Introductory Programming

Abstract:

Today, programming is regarded as one of the most valuable, yet hardest skills to acquire. The purpose of the thesis is to provide an overview of introductory programming both from the perspectives of learning and teaching, also to map the main difficulties during the learning process and factors most influential towards a student's success. The thesis aims to provide recommendations to improve introductory programming courses.

Keywords:

Introductory programming, programming didactics

CERCS: P175 Informatics, systems theory, S270 Pedagogy and didactics, S281 Computer-assisted education

Sisukord

1. Sissejuhatus

Tänapäeva maailmas, kus arvutite ja arvutiteaduse roll ühiskonnas iga päevaga üha kasvab, peetakse programmeerimisoskust üheks olulisimaks ja perspektiivikaimaks oskuseks ning vajadus pädevate programmeerijate järele on tööturul tõusvas tendentsis suur. Jeanette M. Wing on lausa nimetanud arvutuslikku mõtlemist (ingl *computational thinking*), mille üheks osaks on programmeerimisoskus, fundamentaalseks oskuseks, milles võiksid pädevad olla sarnaselt lugemisele ja kirjutamisele kõik inimesed, mitte ainult informaatikud [1]. Kahjuks on aga levinud eelarvamus, et programmeerimine on keskmisele inimesele üle jõu käiv ning heaks programmeerijaks on võimalik saada vaid vähestel andekatel. Mainitud spekulatiivset eelarvamust toetab kahjuks omamoodi fakt, et programmeerimise algõpe on ka tegelikkuses teatud segmendile õppuritest väga keeruline ning teistele oluliselt kergemini hoomatav, ja seda nii arvutiteaduse kui ka teiste erialade tudengite puhul [2].

Bakalaureusetöö eesmärgiks on teha ülevaade programmeerimisõppest õpilase pilgu läbi, vaadelda kasutusel olevaid programmeerimise õpetamise strateegiaid ning anda lõpuks analüüsi teel soovitusi sissejuhatavate programmeerimiskursuste läbiviimiseks. Töö keskendub eelkõige programmeerimise algõppele – õpilastele, kellel varasem kogemus programmeerimise õppimisega puudub või on vähene –, kuna algõppekursustes on rohkem osalejaid, osalejate hulk on mitmekesisem ning osalejate õppimisoskused on veel lõplikult välja kujunemata, mistõttu on õigete meetodite rakendamine eriti oluline. Arvutiteaduse tudengeid arvestades on oluline ka fakt, mille mainivad ära Porter ja Simon [3] – sissejuhataval kursusel ebaõnnestuvatel tudengitel on kõrge risk erialast loobumiseks. Töös käsitletud kirjandus hõlmab peamiselt programmeerimise algõppe-spetsiifilist kirjandust, kuid lisaks ka üleüldiselt sissejuhatavate informaatikakursuste kohta käivat kirjandust, kuna sissejuhatavatest informaatikakursustest suur hulk keskendub paljus programmeerimisõppele. Eristatud ei ole informaatikat peaerialana õppivaid tudengeid teiste erialade tudengitest – Winslow [4] toob välja, et algajatel on palju ühisomadusi ning nende vajadused on tihti väga sarnased.

Töö sissejuhatusele järgnev peatükk hõlmab kirjandusel põhinevat ülevaadet õpilaste käitumismustritest õppimisel, erinevatest õppijatüüpidest ning peamistest programmeerimise õppimisel esinevatest raskustest. Töö kolmas peatükk käsitleb levinumaid programmeerimise õpetamise strateegiaid – iga strateegiat iseloomustatakse vastava metoodika, eeliste ja puuduste ning konkreetsel strateegial põhinevate uurimuste tulemuste abil. Neljandas peatükis

analüüsitakse eelnevalt käsitletud teooriat – tehakse järeldusi ning antakse soovitusi sisse-juhatava programmeerimisõppe edendamiseks.

2. Programmeerimise õppimine

Selleks, et parandada õpetamisstrateegiat, on esmalt vaja kursis olla õppimisega õpilase perspektiivist. Tudengite õppimisharjumuste, õppimist mõjutavate faktorite ja levinuimate murekohtadega kursis olemine annab parema pinnase ka õpiraskuste tekke ennetamiseks või tudengite toetamiseks juba tekkinud raskuste korral. Robins jt [5] kirjutavad lausa, et olulisem olekski programmeerimisõppes keskenduda õpilaste õppimisele, mitte nii väga nende õpetamisele – eesmärgiks edendada õpitu paremat kinnistumist ja arendada õpilaste oskust reflekteerida ning iseseisvalt õppida ka edasises elus.

Alljärgnevalt on tehtud ülevaade õppijate seas levinud käitumismustritest, nende eelistest ja puudustest ning reaalteaduste tudengite seas eristatavatest õppijatüüpidest. Lisaks on käsitletud programmeerimise õppimise alustamisel sageli esinevad raskuseid.

2.1 Käitumismustrid õppimisel

Bjork jt [6] toovad välja iseseisva õppimise üha suureneva tähtsuse tänapäeva maailmas. Ka programmeerimisõppe saab valdavas osas liigitada iseseisva, juhendamata õppimise alla – akadeemilises kontekstis on tavaliselt kasutusel küll näiteks juhendajaga koos programmeerimist hõlmavad praktikumid ja ka loengud, kuid enamik materjali tuleb tudengeil omandada ja kinnistada siiski iseseisvalt. Iseseisva õppeprotsessi juhtimises ja enese hindamises esineb hiljutiste uuringute põhjal tihti suuri puudujääke, kuna toetutakse liigselt oma intuitsioonile, kuigi sellel konkreetsel juhul kipub intuitsioon olema ebausaldusväärne [6].

Näiteks nendivad Bjork jt [6], et tihti peavad õpilased ebaefektiivseid õppimisstrateegiaid efektiivseteks; kusjuures õpilaste üleüldine vähene informeeritus efektiivsete õppimistehnikate osas on üheks põhjuseks, miks õpilased ei kasuta tõestatult häid tehnikaid – eeldatakse, et õpilasi pole vaja õppima õpetada. Ka McNamara [7] sõnul on probleemne hariduses levinud arusaam, et oluline on õpilastele edastada materjali, kuid mitte keskenduda sealjuures sellele, kuidas saaksid õpilased seda materjali parimal viisil omandada – arvatakse et juhul, kui materjal on kursuses kaetud, peaksid õpilased sellest automaatselt ka aru saama. Eelneva toetuseks selgub Marissa K. Hartwigi ja John Dunlosky poolt 2011. aastal läbi viidud uuringust [8], et kõigest 36% õpilastest õpivad nii, nagu nad õpivad, seetõttu, et nad järgivad õppimisel õppejõult saadud suuniseid; Kornelli ja Bjorki 2007. aasta uurimuses [9] tõdes sama vaid 20% vastanutest, mis tõestab õpilaste vähest informeeritust erinevate õppemeetodite osas õppejõudude poolt.

Õpilaste õppimisharjumuste uurimisel on täheldatud mitmeid eriilmelisi strateegiaid. Prominentseimad ning erinevates käsitlustes sagedaimini esinevad neist on massharjutamine (ingl *massed practice/blocked practice*), hajutatud harjutamine (ingl *distributed practice/spaced practice*) ning praktilise testimise strateegia (ingl *practice testing/self testing*), mida vaadeldakse põhjalikumalt ka käesolevas töös. Kirjanduses käsitletakse levinutena ka näiteks ka vaheldatud harjutamise strateegiat (ingl *interleaved practice*), materjalide ülelugemist (ingl *rereading*) ja teksti märgistamist ja allajoonimist (ingl *highlighting and underlining*), kuid kuna näiteks viimased kaks on programmeerimisõppega töö autori hinnangul nõrgemas seoses kui eespool mainitud strateegiad, siis selles töös neile sügavamalt ei keskenduta.

2.1.1 Massharjutamine

Üheks levinuimaks, kuid võrdlemisi ebaefektiivseks õppimistaktikaks on massharjutamine. Massharjutamiseks nimetatakse strateegiat, mis koosneb pikkadest ja intensiivsetest õppesessioonidest [10].

Massharjutamise peamiseks ja levinuimaks näiteks on nn tuupimine (ingl *cramming*), mis hõlmab millegi (tihti esmakordset) intensiivset õppimist hinnatavale tööle eelnevate päevade või tundide jooksul [11]. Sellise lähenemise populaarsuse üheks põhjuseks on iseenesestmõistetav fakt, et paljud õpilased alustavad õppimisega alles viimasel õhtul – sellise teguviisi juures on kõik teised strateegiad välistatud isegi juhul, kui õpilane neist tegelikult teadlik on. Siinkohal tuleb samas välja tuua, et Brinthaupt jt [12] jõudsid järelduseni, et õppimise viimasele hetkele jätmist ja nn tuupimist tuleks siiski eristada, kuna tihti valivad õpilased tuupimise tee mitte vältimatu sunni tõttu, vaid vabatahtlikult, kuna naudivad viimasel hetkel pinge all õppimisest tulenevat väljakutset. Sama uuringu põhjal järeldus ka, et vabatahtlikult sellise strateegia valinud õpilaste tulemused olid sunniviisiliste tuupijate tulemustest oluliselt paremad, põhjuseks tõenäoliselt esimeste vilumus sellise õppimisviisi kasutamisel.

Massharjutamise teed mineku kasuks otsustamisel mängib rolli ka see, et massharjutamise puhul on materjali töötlemine oluliselt lihtsam ja soravam [6]. Nate Kornelli poolt läbi viidud uurimuse [11] käigus läbisid katsealused kolm eksperimenti, mis tõestasid, et vahetult peale esimest õppimissessiooni peeti tuupimist õpilaste poolt õppimistehnikana väga efektiivseks ning ennustati, et ka tulevase kokkuvõtliku testi tulemused võiksid just seda stra-

teegiat kasutades head olla. Kornelli arvates eeldasid uuringus osalejad tõenäoliselt, et suutlikkus õpitut lühiajaliselt peale õppimise lõpetamist meenutada on tugevalt seotud ka õpitu kinnistumisega pikas perspektiivis, mis tegelikult tõele ei vasta. Ka Bjork jt [6] tõid välja, et fakt, et õpiprotsessi käigus on õpitut lihtne meenutada, ei tähenda tingimata, et seda suudetakse meenutada ka teisel ajal ja kohas, kuna meenutamine on suuresti stiimulitest ja/või hiljutisusest (ingl *recency*) sõltuv. Eelnevast tulenevalt on tihti õpilaste õppetulemused sellist strateegia kasutades head, mis sellest, et materjali püsivalt ei omandatud [11]. Seega tekib olukord, kus lisaks õpilastele endile võivad õpilaste teadmisi ja tegelikku pädevust üle hinnata ka õppejõud.

2.1.2 Hajutatud harjutamine

Massharjutamisele vastandatakse üldiselt hajutatud harjutamist. Hajutatud harjutamine on defineeritud kui õppimisstrateegia, mille käigus õpitakse lühikeste eraldiseisvate sessioonidena pikema ajaperioodi vältel [13].

Hajutamise kasuteguri õppimisprotsessis tõestas Ebbinghaus eksperimentaal-psühholoogilises uurimuses [14], milles viis läbi eksperimendi õppimaks pähe valiku tähendusetuid täheühendeid. Nimetatud eksperimendi tulemusi analüüsides selgus, et hajutatud õppimise meetodit kasutades oli vaja täheühendeid õppimisel korrata samaväärse õpitulemuse nimel peaaegu poole vähem kordi kui massharjutamise korral. Hajutatud harjutamise kasulikkuse tõestas ka näiteks Nate Kornell [11].

Dunlosky [15] põhjal kasutatakse hajutatud harjutamist intuitiivselt nii mõnegi mitteakadeemilise oskuse omandamiseks – näiteks tantsimist harjutades või videomänge mängides on enamasti iseenesestmõistetav, et esinemise eelsel viimasel õhtul kogu õhtu tantsimisest või ühe pika sessiooni jooksul arvutimängude mängimisest kasulikum on pühenduda kummalegi tegevusele pika aja vältel lühemate sessioonidena. Akadeemilises kontekstis aga ei ole hajutatud harjutamine üldiselt õpilaste esimene valik [15]. Põhjustena võib ka siinkohal tuua välja eelnevalt nimetatud massharjutamise kasuks otsustamisel rolli mängivad asjaolud. Esiteks tõik, et viimasel hetkel õppimist alustades ei ole võimalik enam hajutatud harjutamist rakendada. Teiseks fakt, et hajutatud harjutamise puhul on kiired ja silmnähtavad tulemused visad tulema – hoolimata sellest, et pikas perspektiivis on tegu kasulikuma lähenemisega, on suutlikkus hajutatud harjutamisel aktiivse õppimisprotsessi jooksul nõrgem, mis võibki panna õpilasi selle efektiivsuses kahtlema [11].

2.1.3 Praktiline testimine

Praktilise testimise strateegiaks nimetatakse õppimisviisi, mille käigus õpilane testib ennast ise või teeb harjutusteste juba õppeprotsessi käigus – näiteks kattes äsja õpitud materjali kinni ning püüdes seda peast meenutada, selle asemel, et materjal lihtsalt uuesti üle lugeda; õppides küsimuskaartide (ingl *flash cards*) abil või püüdes eksamile või kontrolltööle sarnanevaid teste lahendada juba õppimise ajal [15]. Nii Hartwig jt [8], Dunlosky [15] kui ka Gurung [16] toovad praktilise testimise strateegia välja kui ühe efektiivseima õppemeetodi, kuid samas on jõutud ka järeldusele [16], et strateegia on pigem vähepopulaarne. See võib olla põhjustatud asjaolust, et enese testimist nähakse vaid vahendina enda teadmisi hinnata, mitte viisina õpitut kinnistada ja õppimist edendada [9] [17].

Praktiline testimine on õpistrateegiana efektiivne, kuna sunnib õpilase õpitut uuesti meelde tuletama ning tulenevalt inimmälu ülesehitusest muudab meelde tuletamine (ingl *retrieval*) informatsiooni tulevikus mälus lihtsamini ligipääsetavaks [6, 9]. Roediger ja Karpicke [18] viisid läbi eksperimendid, milles katsealused kasutasid õppimisel nii praktilise testimise strateegiat kui ka materjali mitmekordse läbilugemise strateegiat. Nimetatud eksperimentide tulemus oli analoogne massharjutamise ja hajutatud harjutamise vahelisele erinevusele – mitmekordne lugemine andis õppuritele eelise küll õpitu meenutamisel peale lühikese aja möödumist, tekitades sealjuures õpilastele ekslikku illusiooni meisterlikkusest, kuid pikemaajaliselt kinnistus materjal oluliselt paremini just praktilise testimise strateegiat kasutades. Korrapärasel ja pideval praktilisel testimisel rakendub kõrvalmõjuna iseeneslikult ka hajutatud harjutamise kasufaktor [18]. Lisaks on õppimisprotsessis kasulik ka enese testimisel saadav tagasiside, mis juhib tähelepanu aspektidele, milles teadmisi on veel vajaka [15].

2.2 Õppeedukust mõjutavad faktorid programmeerimises

Byrne ja Lyons toovad välja, et tihti eeldatakse, et üldiselt andekad õpilased peaksid probleemideta saama hakkama ka programmeerimisega, kuid kahjuks näitab kogemus vastupidist [19]. Seega tõstatub küsimus – mis täpsemalt mõjutab tudengi edukust sissejuhatavas programmeerimisõppes?

Järgnevalt antakse ülevaade nii isikuomadustest kui ka õpilase varasemat tausta ja kogemust puudutavatest aspektidest, mille mõju edukusele programmeerimise algõppes ning üldistes sissejuhatavates informaatikakursustes on kirjanduses uuritud.

2.2.1. Õppijatüübid

Järgnev lõik tugineb Richard M. Felderi artiklile [20]. Erinevad inimesed võtavad vastu ja töötlevad informatsiooni kõige paremini erinevatel viisidel – mõned õpilased eelistavad fakte, teised teooriaid; mõned visuaalset informatsiooni, teised verbaalset; mõned individuaalselt õppimist, teised koostööna. Eelistatud õppimisviisi järgi eristatakse erinevaid õppijatüüpe. Õppijatüüpidega arvestamine on õpetamisviisi valikul oluline, kuna ebasobivate ja/või läbimõtlemata õpetamisviiside kasutamine võib omada negatiivset mõju õpilaste suhtumisele ka eriala sisu suhtes. Samas tuleb märkida, et õppimisviisidele liiga tugevas vastavuses olevad õpetamisviisid vähendavad õpilaste osavust edasises elus erialasel elukutsel hakkama saada. Reaalses elus tuleb osata omandada informatsiooni erinevatel, sh mitte-eelistatud õppimisviisidel, kuna rutiinselt töötlemist vajav informatsioon võib tulla mistahes kujul. Ideaalis tuleks õpetamisel jõuda kõigi õppijatüüpideni.

Maia jt [21] koostasid 2017. aastal süstemaatilise ülevaate programmeerimise õpetamise alasest kirjandusest aastatel 2006 – 2017, mille tulemusena selgus, et enim kasutatakse tänapäeval programmeerimise õpetamisel Felder-Silvermani 1988. aastal välja pakutud inseneriala õppijatüüpide mudelit [22]. Nimetatud mudeli reliaabluse (ingl *reliability)* ja valiidsuse (ingl *validity)* on tõestanud oma uuringus näiteks Litzinger jt [23]. Gomes jt [2] tõid selle mudeli valiku kasuks otsustamisel välja ka järgnevad tegurid: hindamisküsimustiku [24] kättesaadavus internetis, tulemuste automaatne genereerimine õpilaste vastuste põhjal ning tulemuste otsene interpretatsioon. Tulenevalt eelnevalt väljatoodud hüvedest ning faktist, et sellel mudelil põhinevatele õppijatüüpidele on vastavuses ka mudel sobilikest õpetamisviisidest [22], antakse ka käesolevas töös põhjalikum ülevaade õppijatüüpidest just Felder-Silvermani mudeli põhjal, kuigi lisaks tõid Maia jt [21] programmeerimise õpetamisel tänapäeval kasutusel olevatena välja veel ka näiteks VARKi modaalsuste mudeli, Perkinsi mudeli ning Gregorc Style Delineatori (GSD) mudeli. Lisaks on insenerihariduses kasutatud õppijastiilide eristamiseks ka näiteks Myers-Briggsi tüübiindikaatoril (MBTI) põhinevat mudelit, Kolbi mudelit ning Herrmann Brain Dominance Instrumenti [20].

Järgnev lõik ning sellele järgnev mudeli kirjeldus on koostatud Felderi ja Silvermani artikli [22] põhjal. Struktureeritud hariduslikus kontekstis võib õppimist käsitleda kui kaheastmelist protsessi, mis hõlmab informatsiooni vastuvõtmist ning töötlemist. Sõltuvalt individuaalselt eelistatud lähenemisest nimetatud vastuvõtmis- ja töötlemisprotsessidele jaotuvad inseneriõppe õpilased Felder-Silvermani mudeli põhjal alljärgneval viisil nelja dimensiooni

alusel, moodustades kokku 16 (2^4) erinevat õppimisstiili (üks stiil oleks näiteks sensoorne, verbaalne, aktiivne, globaalne):

1. **taju** (ingl *perception)*:
    a. **sensoorne** (ingl *sensory*) – tudeng eelistab fakte, andmeid ja eksperimenteerimist ning ülesannete lahendamist standardsel viisil; on detailidele orienteeritud ja põhjalik, kuid võib olla seetõttu aeglane;
    b. **intuitiivne** (ingl *intuitive*) – tudeng eelistab printsiipe, teooriaid ja uuendusi ning ei salli kordumist; tüdineb liigsest detailsusest ja mõistab uusi kontseptsioone kiiresti, kuid võib olla hooletu;
2. **sisend** (ingl *input*):
    a. **visuaalne** (ingl *visual*) – tudeng omandab kõige paremini informatsiooni, mis on edastatud kas piltide, diagrammide, jooniste vm näol ning unustab tõenäoliselt näiteks kuulmise teel vastu võetud informatsiooni;
    b. **verbaalne** (ingl *verbal*) – tudeng omandab kõige paremini uut informatsiooni kuulmise, arutamise, selgitamise ning lugemise teel; visuaalne informatsioon kipub kergemini ununema;
3. **töötlus** (ingl *processing*):
    a. **aktiivne** (ingl *active*) – tudengil on vaja informatsiooni edukaks töötlemiseks seda välises maailmas kasutada – kas kaaslastega arutada või praktiliselt testida; eelistab töötada grupis ja eksperimenteerida;
    b. **reflektiivne** (ingl *reflective*) – tudeng eelistab informatsiooni töödelda introspektiivselt reflekteerides näiteks selle üle mõtisklemise teel; meeldib pigem omaette või maksimaalselt ühe kaaslasega koos teoretiseerida;
4. **mõistmine** (ingl *understanding*):
    a. **järjestikune** (ingl *sequential*) – tudeng eelistab õppida materjali etteantud järjestuses ja ühtlaste, sarnase raskusastmega osade kaupa lineaarse protsessina; oskab materjali kasutada ka siis, kui mõistab seda veel vaid osaliselt ning on osavam koonduvas mõtlemises ja analüüsis;
    b. **globaalne** (ingl *global*) – tudeng õpib sööstudena ja eelistab vahel hoopis kohe keerulisema materjali õppimisest alustada, teeb intuitiivseid järeldusi ning ei pruugi osata oma mõttekäike selgitada; tal on raske kasutada materjali, mida ta läbinisti ei mõista; on parem hajuvas mõtteviisis ning sünteesis.

Järgnev tugineb Felder jt artiklile [22]. Algselt koosnes Felder-Silvermani mudel viiest kategooriast – lisaks hetkel kasutusel olevatele dimensioonidele eristati õpilasi ka induktiivse (üksikult üldisele) või deduktiivse (üldiselt üksikule) õpistiili eelistuse järgi, kuid selle kategooria kaotas Richard M. Felder mudelist 2002. aastal, sest jõudis arusaamale, et induktsiooni näol on vähemalt bakalaureuseõppes (ingl *undergraduate level)* tegemist ainuõige lähenemisega. Induktsioon on inimese loomulik õppimisstiil – vastsündinud ei oma sündides teadmisi printsiipidest, mida maailmas rakendama hakata, vaid teevad järeldusi läbi kogemuste. Samas on deduktsioon tehniliste ainete õpetamisel loomulik õpetamisviis, kuna lihtsam on alustada kehtivate põhimõtete selgitamisest ning seejärel tutvustada asjakohaseid näiteid. Deduktsiooni kahjuks räägib muuhulgas näiteks fakt, et selline õpetamisviis jätab õpilastele heidutava ja liigselt aukartustäratava mulje, et õppematerjali autor/juhendaja jõudiski omal käel esimese katsega perfektse teooriani, kui tegelikkuses eelnes sellele pikk eksimuste ja paranduste protsess.

Felder-Silvermani mudelil põhineva õppijatüübi määramiseks on võimalik kasutada internetis kättesaadavat hindamisküsimustikku [24], mis arendati välja Richard M. Felderi ning Barbara A. Solomani koostöös. Oluliseks peetakse siinkohal märkida, et õpilase õppimisstiil viitab vaid *võimalikele* tugevustele või raskusi tekitavatele kalduvustele ning ei väljenda mingil viisil õpilase sobivust või mittesobivust konkreetsele erialale või kursusele [25].

Kirjanduses on uuritud ka õppijatüüpide jaotust programmeerimise ja informaatika tudengite hulgas. Kuigi siinkohal pole hetkel võimalik anda ülevaadet tulemustest Eesti tudengite põhjal, on näiteks Zualkernan jt [26] leidnud, et kultuuriliste ja/või geograafiliste erinevuste roll antud juhul on marginaalne ning seega on töö autori hinnangul asjakohane ja informatiivne ka ülevaade rahvusvahelistest tulemustest. Sagedasti on järeldunud uurimustest visuaalsete õppurite tugev arvuline ülekaal verbaalsete õppurite suhtes [2, 26–30]. Nii Gomes jt [2], Zualkernan jt [26], Zualkernan [27], Carmo jt [28] kui ka Kumar [29] on täheldanud ka sensoorsete õppijate rohkust intuitiivsete õppijatega võrreldes. Nenditud on ka järjestikuse õppimisstiili sagedamat eelistamist globaalsega võrreldes [2, 26–29]. Info töötlemise dimensiooni osas on leitud, et reflektiivseid ja aktiivseid õppijaid on suhteliselt võrdselt [2, 26–28, 30]; vaid Kumar [29] märkis ära aktiivsete õppurite kergelt suurema osakaalu.

Mitmetes uuringutes on püütud tuvastada korrelatsiooni õppijatüüpide ja üldise edukuse vahel programmeerimise algõppe kursustes. Tulemused on vastukäivad: Gomes jt (kasutades Felder-Silvermani mudelit) [2] ning Byrne ja Lyons (kasutades Kolbi mudelit) [19] oma

uurimustes vastavat seost ei täheldanud, samas kui Thomas jt (Felder-Silvermani mudeli põhjal) [31] leidsid kinnitust, et inseneriõppes on tõepoolest teatud tüüpi (reflektiivse, intuitiivse, verbaalse ja järjestikuse õppimisstiiliga) õppurid teistest oluliselt edukamad, mida nentis oma artiklis ka Felder [20]. Zualkernan jt (Felder-Silvermani mudelil põhinevalt) [26], Kumar (Felder-Silvermani mudeli põhjal) [29], Chamillard ja Karolick (kasutades Group Embedded Figures Testi, Felder-Silvermani mudelit, Kolbi mudelit ning Keirsey Temperament Sorterit) [32] ning Allert (Felder-Silvermani mudeli põhjal) [30] jõudsid oma uurimustes järeldusele, et õppimisstiilil ja edukusel arvutiteaduslikus hariduses on tugev seos. Kõigis seoseid täheldanud mainitud uurimustes [26, 29–32] toodi üksmeelselt välja reflektiivset õppimisstiili kasutavate tudengite eelis aktiivset õppimisstiili kasutavate tudengite ees, kahes uurimuses [30, 31] toodi välja ka verbaalse õpistiili paremus visuaalse ees. Maia jt [21] märgivad samas ära, et kuigi õppimisstiil mõjutab õpilase õppimisvõimekust, ei saa seda võtta ühese edukuse või ebaõnnestumise faktorina.

2.2.2. Muud mõjutegurid

Mitmed uuringud [2, 30, 33] on suurimate indikaatoritena õpilase edukusele sissejuhatavates programmeerimis- ja informaatikakursustes toonud välja õpilase enesetõhususe, enesekindluse ja oma teadmiste tajumise. Wilson [34] toob oma uuringus välja naissoost tudengite oluliselt madalama enesehinnangu enesetõhususele, kuid samas nendib, et meessoost tudengite kõrgem hinnang ei olnud korrelatsioonis nende tegelike teadmistega. Üldiselt tudengi sugu tema edukust programmeerimise õppimisel märkimisväärselt mõjutavat ei paista [34]. Ka Byrne ja Lyons [19] oma uuringus seost soo ja tulemuste vahel ei leidnud ning tõdesid, et seose puudumine viitab faktile, et naissoost tudengid on hoolimata ühiskonnas levinud eelarvamusest võimelised programmeerimises häid tulemusi saavutama. Kumar [29] küll märkas oma uuringu tulemustes soo mõju edukusele – meessoost tudengite tulemused olid naissoost tudengite tulemustest oluliselt paremad –, kuid tõi välja, et selle põhjuseks võis olla ka õpilaste erinev taust ja varasem ettevalmistatus.

Palju on uuritud varasema programmeerimiskogemuse mõju tudengite tulemustele sissejuhatavates programmeerimis- või informaatikakursustes. Vastava seose olemasolu on järeldanud oma uuringutes näiteks Gomes ja Mendes [35], kusjuures edukust ei mõjutanud fakt, kas varasemalt õpitud programmeerimiskeel (või programmeerimiskeeled) kattus kursuses käsitletuga. Ka Hagan ja Markham [36] jõudsid järeldusele, et varasem programmeerimiskogemus tuleb kasuks sõltumata õpitud keelest, küll aga mõjutab edukust varasemalt õpitud

keelte arv – mida rohkem, seda parem. Varasema programmeerimiskogemuse kasutegur järeldus ka Byrne'i ja Lyonsi [19] ja Wilsoni [34] uurimustest, kuigi Wilson märkis ära, et eelnev programmeerimiskogemus omas mõju vaid semestri keskel toimunud vaheeksamile ning kogu kursuse lõikes märkimisväärset seost ei olnud. Allert [30] ja Gomes jt [2] samas eelneva programmeerimiskogemuse mõju ei täheldanud.

Uuritud on ka varasemate matemaatikateadmiste seost edukusega programmeerimisõppes. Gomes jt [2] tõid välja, et matemaatikaoskused võiksid viidata probleemilahendusoskusele, mis on programmeerimisõppes fundamentaalne. Hiina tudengite põhjal seost keskkooliastme matemaatikatulemuste ja programmeerimiskursuse hinde vahel aga ei leitud [2]. Wilson [34] aga tõi tugevama keskkoolimatemaatika tausta välja aga kui ühe olulisima potentsiaalse edu indikaatori. Matemaatiliste oskuste positiivne mõju järeldus ka Byrne'i ja Lyonsi [19] ning Bergini ja Reilly [33] uurimustest.

Huvitava tulemusena selgus kahest uuringust [30, 34] arvutimängude mängimise märkimisväärne negatiivne mõju, mida Allert [30] seostas faktiga, et arvutimängude mängijad võisid olla tulnud informaatikat õppima tuginedes eksiarvamusele, et erialal tegeletakse arvutimängude loomisega. Samuti pakkus ta välja põhjenduse, et nendel tudengitel jääb lihtsalt arvutimängude kõrvalt õppimiseks teistest tudengitest vähem aega.

2.3 Peamised raskused programmeerimise õppimisel

Kirjanduse põhjal on arvutiteaduse tudengite seas laialdaselt märgata programmeerimisevastast suhtumist ning tihti on programmeerimise algõppe kursused neile niivõrd vastumeelsed, et viivad halvemal juhul isegi erialastest õpingutest loobumiseni [31]. Alustavatele programmeerijatele raskusi valmistavate aspektide kindlaks määramiseks on tehtud mitmeid uuringuid.

Milne'i ja Rowe'i uuringu [37] põhjal selgus, et õpilastel on laias laastus peamiselt raskusi teemadega, mis on kas otseselt või kaudselt seotud programmi jooksutamise ajal arvuti mälus toimuva mõistmisega – näiteks andmestruktuurid ja viidad, konkreetsemalt objektorienteeritud programmeerimise kontekstis ka näiteks polümorfism, virtuaalfunktsioonid ja dünaamiline mälujaotus. Olulisena märgiti siinkohal ära ka fakt, et selliste teemade puhul polegi õpilaste jaoks keeruline nende kontseptuaalne mõistmine, vaid just nimelt nende praktiline implementeerimine.

Lisaks eelmainitud teemadele ilmnes Milne'i ja Rowe'i uuringust [37] ka rekursiooni õppimise keerukus, mida nad põhjendasid faktiga, et õpilastel on raske enda jaoks visualiseerida rekursiivse programmi jooksutamise mittelineaarset protsessi. Raskusi rekursiooni õppimisel on uurinud ka näiteks Raja Sooriamurthi [38], kes tõi välja, et tihti keskenduvad õpilased imperatiivse programmeerimise lähenemise kohaselt sellele, *kuidas* programm toimib ja vähem sellele, *mida* programm tegema peaks, kuid rekursiooni käsitlemisel on kasulikum kasutada just viimast, deklaratiivset lähenemist. Deklaratiivse lähenemise eelis rekursiooni käsitlemisel järeldus ka Ginat' jt uurimusest [39]. Suur osa tänapäeva programmeerimise algõppe kursuseid, sealhulgas ka selles töös käsitletavad, põhinevad aga just imperatiivse programmeerimise paradigmal.

Järgnev lõik põhineb Leon E. Winslow' artiklil [4], kui pole viidatud teisiti. Matemaatikud peavad keerulisimateks ülesanneteks neid, milles ülesandepüstitus on ette antud sõnalisel kujul, mitte matemaatiliste sümbolite notatsioonis. Programmeerimises on aga kõik ülesanded sõnalised ning sageli on õpilaste esimeseks raskuseks üleüldse etteantud probleemist aru saamine. Kui probleemi olemus on endale selgeks tehtud ning ka lahendusideeni jõutud, on paljudele algajatele järgmiseks suureks katsumuseks lahenduse arvutisobilikule kujule teisendamine. Winslow' arvates on alustavatel programmeerijatel tihti raskusi ka probleemilahenduse tehnikatega – kasutatakse üldisi probleemilahenduse strateegiaid (näiteks sarnase lahenduse kopeerimine) selle asemel, et kasutada probleemispetsiifilisi strateegiaid, mis on oluliselt tulemusrikkamad. Ta märgib ära, et kogenud programmeerijaid ei pruugigi eristada algajaist laiemad teadmised programmeerimiskeele süntaksi ja semantika osas, küll aga suurem teadmiste pagas probleemi domeeni ja lahendusstrateegiate osas. Adelson ja Soloway [40] hinnangul on domeeni tundmine tarkvaraarenduses suure tähtsusega ning programmeerija oskused mingi ülesande lahendamisel sõltuvad suurel määral tema kogemusest ülesande domeenis. Puudujääke tudengite probleemilahendusoskustes konstateerivad ka Gomes ja Mendes [41], kes toovad probleemsena välja ka järjekindluse puudumise – tudengid kipuvad probleemi lahendamise osas kiiresti alla andma, kuid programmeerimisprobleemide lahendamine nõuab pühendumust ja püsivust.

Gomes ja Mendes [41] nendivad, et õpilased kasutavad valesid õppimismeetodeid – ollakse harjunud valemite või protseduuride pähe õppimisega, omamata sealjuures täielikku arusaama aluseks olevast kontseptsioonist. Lisaks ei teadvustata endile tihti, et täielikult teooriapõhine lähenemine ei ole programmeerimise õppimisel mõistlik ning selle asemel tuleks

läheneda asjale praktilisest küljest ja keskenduda õppimisprotsessis praktiliste ülesannete lahendamisele [41].

Raskusi tekitab programmeerimise õppimisel ka motivatsiooni puudumine [41].

3. Programmeerimise õpetamise meetodid

Vihavainen jt [42] toovad välja, et programmeerima õppimise keerukuse põhjuseid seostatakse muu hulgas programmeerimise kursustel kasutatavate õppemetoodikatega. Selleks, et uurida õpetamisviisi muutmise mõju õpilaste tulemustele, tegid nad kvantitatiivse süstemaatilise ülevaate aastatel 1980 – 2014 ilmunud sissejuhatava programmeerimise kursuste õpetamisel korraldatud sekkumistest (ingl *interventions*). Selgus, et strateegiast sõltumatult tõstis õpetamisstrateegia muutmine tavapärase loengutel ja praktikumidel põhineva lähenemisega võrreldes kursuse edukalt läbinud tudengite määra keskmiselt kolmandiku võrra, mis tõestab läbimõeldud muudatuste tegemise kasulikkust ja põhjendatust. Kuigi uuringus statistiliselt strateegiate vahel eriti suuri erinevusi ei leitud, paistsid siiski enim positiivselt silma strateegiad, millega õpilased said samastuda (ingl *relatable content*) – näiteks mängude kasutamine – ning mis rakendasid sealjuures ka mõnda koostöö tegemise vormi – näiteks paarisprogrammeerimist.

Järgnevalt antakse ülevaade töö autori hinnangul asjakohaseimatest kirjanduses levinud õpetamismeetoditest, nende kasutamisest ning kasutamisel saavutatud tulemustest tehtud uuringutel põhinevalt.

3.1 Paarisprogrammeerimine

Nagappan jt [43] ning Williams jt [44] kirjutavad, et akadeemilises keskkonnas tuleb õpilastel programmeerima õppida sageli üksinda ning koostöö tegemisse suhtutakse kui petturlusse, mis läheb aga vastuollu sellega, et erialasel tööl nõutakse just nimelt koostöö tegemise oskust. Ka Porter jt [45] konstateerivad, et kahe inimese koostöö kasulikkus erialase töötamise kontekstis on tõestatud. Üheks koostööd hõlmavaks õpetamismeetodiks on näiteks paarisprogrammeerimine (ingl *pair programming*).

Paarisprogrammeerimise korral on ühe arvuti taga korraga kaks inimest – üks, juht (ingl *driver*), kes vastutab koodi trükkimise või disaini dokumenteerimise eest ning teine, navigaator (ingl *navigaator*), kelle ülesandeks on juhi töö vaatlemine, eksimuste otsimine ning valmidus igal hetkel juhti ajurünnakuga (ingl *brainstorming*) abistada [43].

Paarisprogrammeerimise kasutuselevõtuga paranenud õpitulemused ja vähenenud väljalangevus sissejuhatava programmeerimise kursusel on järeldunud mitmest uuringust [43, 44, 46]. Salge'i ja Berente'i 2016. aastal avaldatud uuringus [47] viidi läbi metaanalüüs 15 aasta

vältel avaldatud uuringute põhjal, mille tulemusena selgus, et paarisprogrammeerimine parandab tarkvarakvaliteeti, õppetulemusi ja vähendab ülesande lahendamisele kuluvat aega. Selle meetodi üheks kasuteguriks on veel näiteks elavnenud diskussioon praktikumides, kusjuures suhtlus praktikumijuhendajaga toimus oluliselt lühemate intervallidena kui sooloprogrammeerimisel, kuna õpilased arutasid tekkinud küsimusi pigem omavahel paarides ning küsisid juhendaja abi vaid siis, kui miski jäi ka kahe peale ebaselgeks [44]. Seega vähenes aeg, mis kulus muidu tööseisakuks sel ajal, kui õpilane sattus ülesande lahendamisel kimbatusse, kuid juhendaja tegeles veel eelmise õpilase (või õpilaste) küsimustega [44]. Paarisprogrammeerimine omab positiivset mõju ka õpilaste enesekindlusele ja rahuolule [46, 48]. Vihavainen jt [42] toovad samas teiste õpetamisstiilidega võrreldes paarisprogrammeerimise kasutuselevõtu välja kui ühe vähima mõjuga muudatuse, kuid mõju oli nende sõnul siiski positiivne ja märgatav.

Oluline on siinkohal märkida, et paarisprogrammeerimise rakendamisel tuleb pöörata tähelepanu sellele, et paariliste oskuste tasemed oleksid sarnased, kuna sellel on suur roll meetodi efektiivsusel [49]. Lisaks on oluline paariliste iseloomude sobivus [43].

3.2 Mängupõhine programmeerimine

Leutenegger ja Edgington [50] avaldavad arvamust, et õppeedukus ja tudengite väljalangevuse vältimine on tugevalt seotud õpitava materjali kaasahaaravusega tudengite silmis ning toovad tudengite paelumise võimalusena välja mängude kasutamise programmeerimise õpetamise protsessis. Ka Becker [51] väidab, et meelelahutusliku aspekti väärtust ei tohiks õpetamisel alahinnata, kuna just õpitavast innustatus motiveerib õpilasi rasketel aegadel.

Mängupõhine programmeerimine (ingl *game-themed programming*) integreerib sissejuhatava programmeerimise kursustesse lihtsad interaktiivsed graafikaprogrammid, kusjuures eesmärgiks ei ole õpetada tudengeid mänge programmeerima, vaid õpetada tudengitele abstraktseid programmeerimise kontseptsioone läbi mängude tööpõhimõtete tutvustamise [52]. Mängupõhise programmeerimise implementeerimise ühe konkreetse võimalusena toovad Sung jt [53] välja näiteks ülesanded, kus tudengid peavad rakendama arvutiteaduslikke kontseptsioone (andmestruktuure, täisarvude jagamist, juhusliku arvu genereerimist, klassihierarhiat vm) selleks, et täiendada etteantud graafika ja kasutusfunktsionaalsustega mängulaadset rakendust.

Tõestatud on mängupõhise programmeerimise kasutuselevõtust tulenenud vähenenud

tudengite väljalangevus, uute tudengite suurenenud huvi kursuste vastu ning, sõltumata kriitikute hüpoteesist, naiste ja meeste võrdne huvi mängupõhise lähenemise vastu [50]. Sung jt [53] tõid võrreldes konsoolipõhiste ülesannetega märkimisväärsetena välja kursuse edukalt lõpetanute arvu kasvu ja paremad õpitulemused, kusjuures fakt, et õpilased ülesande lahendamisel oma mänge ise mängisid, aitas tuvastada programmis esinenud vigu ning parendas seeläbi lahendust. Välja toodi ka tähelepanek, et interaktiivsete graafiliste rakenduste kasutamine toetas eksperimenteerimist ja visualiseerimist, mis parandas õpilaste tulemusi ja enesekindlust [53]. Becker [51] rõhutas õpilaste märgatavalt suurenenud entusiasmi õpitava suhtes peale mängude kasutuselevõttu. Sarnaselt paarisprogrammeerimisele aga tõid Vihavainen jt [42] välja mängupõhise lähenemise kui ühe vähim positiivset mõju avaldanud õpetamisstrateegia.

3.3 Kaaslase Juhendamine

Harvardi Ülikooli füüsika õppejõud Eric Mazur jõudis oma karjääri ühes punktis tõdemuseni, et tudengid küll memoreerivad loengutes käsitletut ja oskavad seda vajadusel standardseid ülesandeid lahendades ka rakendada, kuid sügavamast kontseptsioonide mõistmisest jääb vajaka [54]. Simon ja Cutts [55] tõstatasid Mazuri poolt välja toodud probleemi potentsiaalse olemasolu ka arvutiteaduse kontekstis – võimalik on, et tudengid programmeerivad küll toimivaid lahendusi, kuid tegelikult aluseks olevat kontseptsiooni ei mõista –, ning toovad lahendusena välja lähenemise, mille arendas välja ja võttis kasutusele ka Mazur – Kaaslase Juhendamine (ingl *Peer Instruction*).

Kaaslase Juhendamine (KJ) on aktiivsel õppimisel põhinev meetod, mis kasutab loenguformaati ära selleks, et rakendada õpilasi probleemi lahendamisse ja rühmana arutlemisse [3]. Oluline on siinkohal eristada fikseeritud komponentidest koosnevat struktureeritud Kaaslase Juhendamise meetodit eraldiseisvast samanimelisest (ingl *peer instruction/peer tutoring*) õppemeetodist, mis seisneb bakalaureusetudengi õppeassistendina õppeprotsessi kaasamises [56].

Järgnev meetodi kirjeldus põhineb Simoni ja Cuttsi artiklil [55]. KJ puhul valmistuvad õpilased loenguks juba kodus ette (loevad läbi loengus käsitleva materjali) ja teevad läbi valikvastustega testi eesmärgiga stimuleerida õppeprotsessi ja anda tagasisidet valmiduse osas loengus õppida. Loengusse põimitakse seejärel valikvastustega küsimusi ja arutelu. Selleks, et suunata õpilasi sügavamate kontseptuaalsete probleemide ja levinud eksiarvamuste üle mõtisklema, kasutatakse järgmisi samme (iga samm on olulise tähtsusega):

1. õpilased mõtisklevad küsimuse üle individuaalselt ning valivad vastuse (selleks kasutatakse tavaliselt klikkereid);
2. õpilased arutlevad omavahel ettemääratud gruppides;
3. õpilased valivad samale küsimusele uuesti vastuse (gruppides tuleb jõuda konsensuseni);
4. küsimuse üle arutletakse klassiüleselt (iga grupp selgitab oma vastust) ning juhendaja selgitab lahendust enda poolt.

Juhul, kui klikkerite abil vastamine ja arutelu ei ole edukad, saab juhendaja korraga reageerida ja vajadusel teemat kohe väiksemahulise loengu näol edasi selgitada – sellisel hetkel on eksperdi seletus eriti väärtuslik, kuna õpilaste ajud on valmis põhjendust oma isikliku arusaamaga siduma [55].

Kaaslase Juhendamise tehnika on teadaolevalt võrreldes standardse lähenemisega (loengud) vähendanud läbikukkujate arvu sissejuhatavates informaatikakursustes olulisel määral, kusjuures KJ halvim läbikukkujate määr oli endiselt väiksem kui standardsetel loengutel põhinevate kursuste keskmine läbikukkujate määr [3]. Suur hulk tudengeid andsid KJ-le peale kursust positiivset tagasisidet – üle 60% õpilastest soovitaksid samasugust lähenemist ka teistele kursustele ning umbes 80% õpilastest avaldasid arvamust, et selline strateegia oli nende õppimisprotsessis kasulik; lisaks tõid tehnika kasutamise väärtuslikkuse õpetamise poole pealt välja ka juhendajad – lihtsam oli tuvastada õpilaste arusaamises tekkivaid probleeme [57].

Puuduste ja arenemiskohtadena võib välja tuua, et teatud hulk õpilasi aktiivselt siiski arutelus ei osalenud ning vaid vähemus klassist luges enne loengut materjalid läbi, mistõttu tehnika kasu nende jaoks ei avaldanud [57].

4. Järeldused ja soovitused

Töö käigus ilmnesid nii mõnedki kitsaskohad ja arenemisvõimalused praeguses sissejuhatavas programmeerimisõppes.

Esmalt järeldus üldhariduslik, kuid ka programmeerimise konteksti rakenduv fakt, et õpilastel on raskusi oma õpiprotsessi teadliku struktureerimisega. Teadmine, et õppimise viimasele hetkele jätmine (ja seeläbi sund omandada kogu materjal tuupimise teel) ei ole mõistlik, on küll enamasti intuitiivselt mõistetav, kuid samas ei ole kirjandusel põhinevalt tudengid eriti hästi kursis ka alternatiivsete tehnikate ning nende kasutamisega kaasneva tuluga, mistõttu ei oska nad neid ka kasutada. Näiteks selgus uuringute põhjal, et üht suurima tõestatud kasuga õppimisstrateegiat – enese praktilist testimist – nähakse vaid võimalusena enese teadmisi kontrollida, kuid mitte viisina aktiivselt õppida. Lisaks ei pruugi olla intuitiivne näiteks fakt, et ka varem, kui viimasel hetkel massharjutamine ei ole niivõrd kasulik, kui õppimise ajaliselt ühtlane jaotamine. Töö autori arvates võiks olla märkimisväärselt abi õpilaste hulgas erinevate õppimisstrateegiate eeliste ja puuduste osas teadlikkuse tõstmisest, kuna see aitaks langetada õppimistrateegia valikul teadlikumaid otsuseid. Kuna programmeerimise ainetes välja pakutud (kodu-)ülesanded on tihti reaalelulised, siis ühe võimalusena võiks töö autori arvates käsitleda õppimisstrateegiaid näiteks sissejuhatava programmeerimiskursuse esimeste ülesannete koostamisel (kasvõi tingimuslausete teemas). Õppimisstrateegiate osa ülesandes ei peaks sealjuures olema tingimata mastaapne, vaid piisaks ka lihtsalt õpilastes huvitekitavast pealiskaudsest käsitlusest, millele võiks lisada viite materjali põhjalikuma käsitluse juurde. Selline ülesanne oleks samal ajal nii õppimisstrateegiate osas informatiivne kui ka programmeerimise seisukohalt arendav.

Mitme uuringu tulemuste põhjal järeldus ka muster, et Felder-Silvermani mudelil põhinevas klassifikatsioonis leidub selliste iseloomuomadustega õppureid, kelle tulemusi praegune programmeerimise algõppe ülesehitus ei soodusta. Tugevaimalt eristus korrapära, kus aktiivselt infot töötlevate tudengite tulemused on reflektiivselt infot töötlevate tudengite omadest nõrgemad. Seega võiks olla kasulik rakendada programmeerimisõppes õpetamismeetodeid, mis soodustavad rohkem ka aktiivset õppimisstiili kasutavaid tudengeid. Paslikuks meetodiks võiks olla näiteks paarisprogrammeerimine, mille sobilikkuse aktiivsete õppurite suhtes on välja toonud ka Zualkernan [27]. Paarisprogrammeerimise õpetamismeetodina kasutuselevõtu kasulikkus seisneb lisaks faktis, et selgus, et see tõstab tõestatult õpilaste

enesekindlust ning töö käigus järeldus, et enesekindlus on sissejuhatavas programmeerimiskursuses üks suuremaid indikaatoreid õpilase edukusele. Lisaks paarisprogrammeerimisele rakendaks aktiivset õppimist rohkem ka Kaaslase Juhendamise õpetamisstrateegia, mis hõlmab sealjuures ka reflektiivset õppimist (materjali eelnev kodus läbi töötamine). Kaaslase Juhendamise strateegia aitaks kaasa ka ühe programmeerimise õppimisel esineva raskusena välja toodud probleemi – õpilaste vähese kontseptsioonide süvitsi mõistmise – lahendamisel.

Raskusitekitavana selgus töös näiteks veel motivatsiooni puudumine. Võimaliku lahendusena võiks kaaluda töös käsitletud mängupõhise programmeerimise õpetamismeetodit, mis on teadaolevalt õpilaste motivatsiooni tõstnud.

Standardse õpetamismeetodi teadlik asendamine uuenduslikumaga toodi välja kui statistiliselt positiivseid tulemusi toonud muudatus. Siinkohal on veel märkimisväärne, et õpetamismeetodeid võib omavahel vastavalt vajadusele ka kombineerida nagu mainiti töös – häid tulemusi saavutati, kui seoti omavahel õpilastele samastatavat materjali kasutav strateegia ja koostööd hõlmav strateegia. Lisaks on mitme strateegia koos kasutamise positiivset mõju täheldanud ka Porter ja Simon [58].

5. Kokkuvõte

Bakalaureusetöös keskenduti programmeerimise algõppele nii selle õppimise kui ka õpetamise perspektiivist asjakohasel kirjandusel põhinevalt. Esmalt anti ülevaade mõnedest levinuimatest õppimisstrateegiatest – massharjutamisest, hajutatud harjutamisest ning praktiliste testide kasutamisest, uurides nende kasutamise põhjuseid ning eeliseid ja puuduseid. Sealjuures selgus fakt, et suur hulk õpilasi on erinevatest õppimisstrateegiatest ja nende kasutamise eelistest teadmatuses.

Seejärel käsitleti sissejuhatavas programmeerimisõppes edu saavutamisel mõju omavaid faktoreid. Vaadeldi programmeerimishariduses eristatavaid õppijatüüpe, mis iseloomustavad viise, kuidas erinevad õpilased informatsiooni omandavad. Peamiselt keskenduti töös Felder-Silvermani inseneerihariduse tarbeks välja arendatud mudelile, kuna see on statistiliselt tänapäeva programmeerimishariduses enim kasutusel ja kirjanduses sagedaimini uuritud. Peale õppijatüüpide kirjeldati ka teisi faktoreid, mis õpilase edukust sissejuhatavas programmeerimiskursuses mõjutada võivad. Käsitletud faktoritest suurimas seoses heade õpitulemustega oli õpilase enesekindlus.

Töös anti ka ülevaade programmeerimisõppes sageli esinevatest raskustest. Ilmnes, et raskusi tekitavad näiteks teemad, mille mõistmine eeldab arusaama arvuti mälus toimuvast, näiteks rekursioon. Lisaks toodi välja näiteks puudulikud probleemilahendusoskused ja vähene motivatsioon.

Tehti ka kokkuvõte töö autori silmis asjakohasemaimatest õpetamismeetoditest – paarisprogrammeerimisest, mängupõhisest programmeerimisest ning Kaaslase Juhendamise meetodist. Kõik eelmainitud meetodid on saavutanud kirjanduse põhjal häid tulemusi ning sobituvad ka töös välja toodud probleemide konteksti.

Lõpuks anti soovitusi sissejuhatavate programmeerimiskursuste parendamiseks töö teoreetilisel osal põhinevalt, keskendudes näiteks õppimisstrateegiate osas teadlikkuse tõstmisele ning faktile, et programmeerimishariduses kipub Felder-Silvermani mudeli järgi klassifitseeritud õpilaste hulgas reflektiivsetel õppijatel aktiivsete õppijate ees eelis olema.

Töö kirjutamine oli autori arvates hariv, huvitav ning silmaringi avardav. Autori kui sissejuhatava programmeerimiskursuse läbinu jaoks selgus töö koostamise käigus väga palju uut informatsiooni, mida oleks olnud kasulik teada juba enne ise õpilasena kursuse läbimist.

6. Viidatud kirjandus

[1] Wing J. Computational Thinking. *Commun ACM* 2006; 49: 33–35.

[2] Gomes A, Mendes AJ, Marcelino MJ, et al. Student's characteristics and programming learning — A Macanese perspective. In: *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. 2017, pp. 346–353.

[3] Porter L, Bailey Lee C, Simon B. Halving fail rates using peer instruction: a study of four computer science courses. 2013, pp. 177–182.

[4] Winslow LE. Programming Pedagogy—a Psychological Overview. *SIGCSE Bull* 1996; 28: 17–22.

[5] Robins A, Rountree J, Rountree N. Learning and Teaching Programming: A Review and Discussion. *Comput Sci Educ* 2003; 13: 137–172.

[6] Bjork R a. (1), Dunlosky J(2), Kornell N(3). *Self-regulated learning: Beliefs, techniques, and illusions*. Annual Reviews Inc., 2013. Epub ahead of print 01 2013. DOI: 10.1146/annurev-psych-113011-143823.

[7] McNamara DS. Strategies to read and learn: overcoming learning by consumption. *Med Educ* 2010; 44: 340–346.

[8] Hartwig MK, Dunlosky J. Study strategies of college students: Are self-testing and scheduling related to achievement? *Psychon Bull Rev* 2012; 19: 126–134.

[9] Kornell N, Bjork R. The promise and perils of self-regulated study. *Psychon Bull Rev* 2007; 14: 219–24.

[10] Massed Practice definition | Psychology Glossary | alleydog.com, https://www.alleydog.com/glossary/definition.php?term=Massed+Practice (accessed 11 March 2019).

[11] Kornell N. Optimising learning using flashcards: Spacing is more effective than cramming. *Appl Cogn Psychol* 2009; 23: 1297–1317.

[12] Brinthaupt TM, Shin CM. The Relationship of Academic Cramming to Flow Experience. *COLLEGE STUDENT JOURNAL*, 2001, p. 457.

[13] Distributed Practice definition | Psychology Glossary | alleydog.com, https://www.alleydog.com/glossary/definition.php?term=Distributed+Practice (accessed 11 March 2019).

[14] Ebbinghaus H. *Memory; a contribution to experimental psychology*. New York city, Teachers college, Columbia university, http://archive.org/details/memorycontributi00ebbiuoft (1913, accessed 9 March 2019).

[15] Strengthening the Student Toolbox: Study Strategies to Boost Learning, https://www.aft.org/sites/default/files/periodicals/dunlosky.pdf (accessed 7 March 2019).

[16] Gurung R. How Do Students Really Study (and Does It Matter)? *Commun Educ Commun Educ J Personal Teach Psychol Commun Q Commun Educ* 2002; 2: 149–155.

[17] Karpicke JD. Retrieval-Based Learning : A Decade of Progress 3. 2017. Epub ahead of print 2017. DOI: 10.1111/j.1745-6916.2006.00012.x.

[18] Roediger HL, Karpicke JD. Test-Enhanced Learning: Taking Memory Tests Improves Long-Term Retention. *Psychol Sci* 2006; 17: 249–255.

[19] Byrne P, Lyons G. The Effect of Student Attributes on Success in Programming. 2001, pp. 49–52.

[20] Felder RM. Matters of style. *ASEE Prism* 1996; 6: 18–23.

[21] Maia MCO, Guerrero DS, Figueiredo J. Learning styles in programming education: A systematic mapping study. *2017 IEEE Front Educ Conf FIE* 2017; 1–7.

[22] Felder R. Learning and Teaching Styles in Engineering Education. *J Eng Educ - Wash-* 1988; 78: 674–681.

[23] Litzinger T, Ha Lee S, C. Wise J, et al. A Psychometric Study of the Index of Learning Styles©. *J Eng Educ* 2007; 96: 309–319.

[24] Index of Learning Styles Questionnaire, https://www.webtools.ncsu.edu/learningstyles/ (accessed 6 May 2019).

[25] Index of Learning Styles (ILS) | College of Engineering | NC State University, https://www.engr.ncsu.edu/ils/ (accessed 6 May 2019).

[26] Zualkernan IA, Allert J, Qadah GZ. Learning Styles of Computer Programming Students: A Middle Eastern and American Comparison. *IEEE Trans Educ* 2006; 49: 443–450.

[27] Zualkernan IA. Using Soloman-Felder Learning Style Index to Evaluate Pedagogical Resources for Introductory Programming Classes. In: *29th International Conference on Software Engineering (ICSE'07)*. 2007, pp. 723–726.

[28] Carmo L, Pereira F, Gomes A, et al. Learning styles and problem solving strategies.

[29] Kumar AN. Learning styles of computer science I students. 2017, pp. 1–6.

[30] Allert J. Learning style and factors contributing to success in an introductory computer science course. 2004, pp. 385–389.

[31] Thomas L, Ratcliffe M, Woodbury J, et al. Learning Styles and Performance in the Introductory Programming Sequence. In: *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, pp. 33–37.

[32] T. Chamillard A, Karolick D. Using learning style data in an introductory computer science course. 1999, pp. 291–295.

[33] Bergin S, Reilly R. Programming: Factors that Influence Success. 2005, pp. 411–415.

[34] Wilson BC. A Study of Factors Promoting Success in Computer Science Including Gender Differences. *Comput Sci Educ* 2002; 12: 141–164.

[35] Gomes A, Mendes A. A study on student's characteristics and programming learning. 2008.

[36] Hagan D, Markham S. Does it help to have some programming experience before beginning a computing degree program? 2000, pp. 25–28.

[37] Milne I, Rowe G. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Educ Inf Technol* 2002; 7: 55–66.

[38] Sooriamurthi R. Problems in Comprehending Recursion and Suggested Solutions. In: *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, pp. 25–28.

[39] Ginat D, Shifroni E, College T-H. Teaching Recursion in a Procedural Environment - How much should we emphasize the Computing Model? 5.

[40] Adelson B, Soloway E. The Role of Domain Expenence in Software Design. *IEEE Trans Softw Eng* 1985; SE-11: 1351–1360.

[41] Gomes A, Mendes A. Learning to program - difficulties and solutions. 2007, pp. 283–287.

[42] Vihavainen A, Airaksinen J, Watson C. A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*. New York, NY, USA: ACM, pp. 19–26.

[43] Nagappan N, A. Williams L, Ferzli M, et al. Improving the CS1 experience with pair programming. 2003, pp. 359–362.

[44] Williams LA, Wiebe EN, Yang K, et al. In Support of Pair Programming in the Introductory Computer Science Course. *Comput Sci Educ* 2002; 12: 197–212.

[45] Porter L, Guzdial M, Mcdowell C, et al. Success in Introductory Programming: What Works? *Commun ACM* 2013; 56: 34–36.

[46] Mcdowell C, Werner L, Bullock H, et al. Pair programming improves student retention, confidence, and program quality. *Commun ACM* 2006; 49: 90–95.

[47] Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research? *2016 49th Hawaii Int Conf Syst Sci HICSS Syst Sci HICSS 2016 49th Hawaii Int Conf Syst Sci HICSS 2014 47th Hawaii Int Conf On* 2016; 5398.

[48] Hanks B, McDowell C, Draper D, et al. Program Quality with Pair Programming in CS. 5.

[49] Chaparro EA, Yuksel A, Romero PE, et al. Factors Affecting the Perceived Effectiveness of Pair Programming in Higher Education. In: *PPIG*. 2005.

[50] Leutenegger S, Edgington J. A Games First Approach to Teaching Introductory Programming. In: *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, pp. 115–118.

[51] Becker K. Teaching with games: the Minesweeper and Asteroids experience. *J Comput Sci Coll - JCSC*; 17.

[52] Mohorovičić S, Strčić V. An Overview of Computer Programming Teaching Methods.

[53] Sung K, Hillyard C, Angotti R, et al. Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses. *Educ IEEE Trans On* 2011; 54: 416–427.

[54] Education. Farewell, lecture? - PubMed - NCBI, https://www.ncbi.nlm.nih.gov/pubmed/19119207 (accessed 10 May 2019).

[55] Simon B, Cutts Q. Peer Instruction: A Teaching Method to Foster Deep Understanding. *Commun ACM* 2012; 55: 27–29.

[56] Chase JD, Okie EG. Combining Cooperative Learning and Peer Instruction in Introductory Computer Science. In: *Proceedings of the Thirty-first SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, pp. 372–376.

[57] Simon B, Kohanfars M, Lee J, et al. Experience report: peer instruction in introductory computing. In: *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*. Milwaukee, Wisconsin, USA: ACM Press, p. 341.

[58] Porter L, Simon B. Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1. 2013, pp. 165–170.

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Marge Käis,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „**Ülevaade õppimisstrateegiatest ja õpetamisviisidest programmeerimise algõppes**“, mille juhendaja on Reimo Palm, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Marge Käis
10.05.2019