TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kristen Kotkas

Programmeerimise algkursuste eksamid

Bakalaureusetöö (9 EAP)

Juhendajad: Marina Lepp, PhD
Eno Tõnisson, PhD

Tartu 2018

Programmeerimise algkursuste eksamid

Lühikokkuvõte:

Antud bakalaureusetöös uuritakse, millised on erinevate programmeerimise algkursuste eksamid. Selleks antakse teaduskirjanduse põhjal ülevaade teemadest ja programmeerimiskeeltest, mida õpetatakse programmeerimise algkursustel. Seejärel tuuakse kirjandusest välja erinevad eksamites kasutatavad küsimuste liigid ning võrreldakse erinevaid eksami läbiviimise vorme. Bakalaureusetöö lõpus antakse ülevaade Tartu Ülikooli e-kursuse „Programmeerimise alused II“ lõpueksami teinud osalejate tagasisidest sooritatud eksami kohta.

Võtmesõnad:

Programmeerimise algkursus, eksam, eksamiküsimused

CERCS: P175 Informaatika, süsteemiteooria; S270 Pedagoogika ja didaktika

Final Examinations in Introductory Programming Courses

Abstract:

The purpose of this bachelor thesis is to research the final examinations of different introductory programming courses. Based on the academic literature, this thesis reports the concepts and programming languages taught in these courses. Furthermore, this thesis presents the different types of questions used in examinations and compares different forms of examinations. At the end of the thesis, the participants' feedback about the examination in the University of Tartu online course "Introduction to Programming II" is presented.

Keywords:

Introductory programming, CS1, examination, examination questions

CERCS: P175 Informatics, systems theory; S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Programmeerimiskursuste populaarsus on viimastel aastatel järsult tõusnud. Seda nii Eestis, kui ka välismaal [1, 2]. Tartu Ülikooli statistika põhjal [3] on päevasesse õppesse informaatika bakalaureuseõppekavale vastuvõetavate üliõpilaste arv võrreldes 2005. aastaga rohkem kui kolmekordistunud. 2005. aastal võeti informaatika bakalaureuseõppekavale riigieelarvelistele õppekohtadele õppima 61 üliõpilast [4]. 2017. aastal võeti samale õppekavale õppima 188 üliõpilast [5]. Lisaks üliõpilastele mõeldud kursustele on suurenenud huvi ka üldhariduskoolide õpilastele ja teistele huvilistele mõeldud kursuste vastu. Ühe sellise näitena on MOOCide (inglise keeles *Massive Open Online Course*) populaarsuse kasv [6].

Programmeerimiskursuseid on erinevaid ning nendes kasutatakse erinevaid hindamissüsteeme. Seoses programmeerimiskursuste suureneva populaarsusega on programmeerimise õppes suurenenud tähelepanu nendel kursustel osalenud üliõpilaste soorituse hindamisele. Tew ja Guzdial [7] toovad välja, et kuigi paljudel STEM (*Science, Technology, Engineering and Mathematics*) erialadel on ühesed vahendid hindamaks antud erialadel õpitut, siis programmeerimiskursustel need puuduvad.

Hindamine on oluline osa nii õppimisel kui ka õpetamisel, seega tuleb seda läbimõeldult korraldada. Sageli kasutatakse kursusel omandatu hindamiseks eksamit. Antud bakalaureusetöös käsitletakse eksamit, kui kontrollitud oludes tööd, mis viiakse läbi kursuse lõpus. Selleks, et anda õppija omandatu kohta õiglane hinnang, peab eksami koostamisel arvestama erinevate asjaoludega. Programmeerimiskursuse eksam ei pruugi olla ainult paberil kirjutatav töö – see võib sisaldada ka arvutis programmeerimist. Lisaks koosneb eksam erinevatest küsimuste liikidest ning eksami lahendamisel võib lubada kasutada abistavaid materjale.

Eelnevast tulenevalt on bakalaureusetöö peamiseks eesmärgiks anda ülevaade erinevate programmeerimise algkursuste eksamitest. Programmeerimise algkursusena mõeldakse kursust, kus õppija omandab oma esimesed teadmised ja oskused programmeerimises. Algkursus võib olla jagatud mitmeks erinevaks osaks. Antud bakalaureusetöös käsitletakse algkursusena Tartu Ülikooli kursust „Programmeerimise alused II“, mis on „Programmeerimise alused“ jätkukursus. Täpsemalt antakse ülevaade kursusest „Programmeerimise alused II“ bakalaureusetöö viimases peatükis.

Eksamitel peavad õppijad vastama küsimustele ning lahendama ülesandeid. Programmeerimiskursustel võivad nendeks olla programmeerimisülesanded, kus lahenduseks tuleb kirjutada konkreetne programm. Läbi erinevate ülesannete demonstreerivad õppijad oma teadmiseid ja oskuseid. Edaspidi nimetatakse antud bakalaureusetöös küsimusteks nii konkreetseid eksamitel kasutatavaid küsimusi, kui ka programmeerimisülesandeid. Eelnevast tulenevalt on teiseks eesmärgiks teaduskirjandusest koondada erinevad küsimuste liigid ja eksami läbiviimise vormid, mida kasutatakse programmeerimise algkursuste eksamites.

Kolmandaks eesmärgiks on teada saada Tartu Ülikooli kursuse „Programmeerimise alused II“ osalejate arvamus antud aines läbiviidava eksami kohta.

Täitmaks eespool nimetatud eesmärke, otsitakse vastuseid järgmistele küsimustele:

1. Millised on programmeerimise algkursused?
2. Millised on programmeerimise algkursuste eksamid?
3. Kuidas toetavad erinevad eksami läbiviimise vormid eksami sooritamist?
4. Millised on erinevate küsimuste liikide eelised ja puudused?
5. Mida arvavad osalejad „Programmeerimise alused II“ kursusel toimunud eksamist?

Esimesele neljale uurimisküsimusele antakse vastus teaduskirjanduse põhjal. Viimasele uurimisküsimusele antakse vastus antud eksami sooritanute seas läbiviidud küsitluse põhjal.

Käesoleva bakalaureusetöö esimeses peatükis vastatakse küsimusele „Millised on programmeerimise algkursused?“. Selleks tuuakse teaduskirjanduse põhjal välja teemad, mida neil kursustel õpetatakse. Lisaks antakse ülevaade, milliste programmeerimiskeeltega neid teemasid õpetatakse ning millised keeled domineerivad programmeerimise algõppes. Esimese peatüki lõpus tehakse kokkuvõte programmeerimise algkursuste eesmärkidest ning antakse põgus ülevaade programmeerimise algkursustest Eesti kõrgkoolides.

Teises peatükis vastatakse küsimusele „Millised on programmeerimise algkursuste eksamid?“. Selleks antakse kõigepealt ülevaade kujundavast ja kokkuvõtvast hindamisest ning sellest, kuidas eksam täidab kokkuvõtva hindamise rolli. Seejärel kirjeldatakse, kuidas valmivad programmeerimise algkursuste eksamid ning mis teemasid seal käsitletakse.

Kolmandas peatükis saadakse vastus küsimusele „Kuidas toetavad erinevad eksami läbiviimise vormid eksami sooritamist?“. Selleks analüüsitakse artikleid, mis kirjeldavad avatud ja suletud materjalidega eksamit. Selle põhjal võrreldakse neid ja tuuakse välja kummagi eelised ja puudused. Lisaks antakse ülevaade õppija enda koostatud abilehe kasutamisest eksamil, millega üritatakse leevendada avatud materjalidega eksamil esinevaid puuduseid.

Neljandas peatükis vastatakse küsimusele „Millised on erinevate küsimuste liikide eelised ja puudused?“. Selleks analüüsitakse teaduslikke artikleid, mis kirjeldavad eksamites kasutatavaid küsimuste liike, näiteks valikvastustega küsimusi ja kirjalikku lühivastust nõudvaid küsimusi. Seal tuuakse välja võrdlus erinevate küsimuste liikide vahel programmeerimise algkursuste eksamites. Iga liigi kohta esitatakse ka seda liiki illustreeriv näidisküsimus.

Bakalaureusetöö viimase peatükis otsitakse vastust küsimusele „Mida arvavad osalejad „Programmeerimise alused II“ kursusel toimunud eksamist?“. Selleks analüüsitakse Tartu Ülikooli e-kursuse „Programmeerimise alused II“ lõpueksami teinud osalejate tagasisidet tehtud eksami kohta. Antud vastustest selgitatakse, mis eksami sooritanutele meeldis ning mis kõige rohkem raskust tekitas.

Kokku on bakalaureusetöös kaks lisa. Lisa 1 on tagasisideküsitlus „Programmeerimise alused II“ eksami sooritanutele. Lisas 2 on välja toodud näidisülesanded eksamil kasutatavate erinevate küsimuste liikide kohta.

1. Programmeerimise algkursused

Antud peatükis antakse ülevaade programmeerimise algkursustest. Esmalt analüüsitakse teaduskirjandust, et saada teada, kuidas viiakse läbi programmeerimise algkursuseid – mis teemasid antud kursused katavad ja mis keeli seal õpetatakse. Teiseks antakse ülevaade Eesti kõrgkoolides toimuvast – tuuakse välja, kui paljudes kõrgkoolides õpetatakse programmeerimist algtasemel ja missuguseid programmeerimiskeeli selleks kasutatakse.

1.1 Programmeerimise algkursuse mõiste

Programmeerimise algkursusteks nimetatakse kursuseid, kus õppija omandab oma esimesed teadmised ja oskused programmeerimises. Inglise keeles nimetatakse neid kursuseid mitmeti. Peamiselt kasutatakse kahte terminit: *Introductory Programming Course* ja CS1 (*Computer Science 1*).

Termin „CS1“ pärineb 1978. aastal avaldatud ACMi (*Association for Computing Machinery*) arvutiteaduse õppekavast [8]. Seal toodi välja kaheksa järjestikust arvutiteaduse kursust, CS1 – CS8, kus CS1 ja CS2 tähistavad programmeerimise algkursuseid. Hertzil ilmus 2010. aastal artikkel „*What Do "CS1" and "CS2" Mean? Investigating Differences In the Early Courses*“ [9]. Sealt selgus, et kuigi programmeerimise algkursuste õppejõud kasutavad endiselt neid termineid nende originaaltähenduses, siis kursuste sisu on ajaga muutunud. Ta toob välja, et termini „CS1“ all mõeldakse peamiselt programmeerimise algkursust ja „CS2“ all andmestruktuuride algkursust. Uurides õppejõududelt programmeerimisega seotud teemade õpetamist CS1 ja CS2 kursustel selgus, et arusaamad nendest nimetustest ja antud kursustel käsitletavatest teemadest erinevad suuresti ja ühest definitsiooni antud nimetustele ei ole.

Peedoski 2014. aastal kaitstud bakalaureusetöös [10] viidi läbi uuring Eesti kõrgkoolides õpetatavatest programmeerimise algkursustest. Seal välja toodud kursuste nimedest saab teha üldise kokkuvõtte eesti keeles toimuvate kursuste nimede kohta. Enamik kursustest kandsid nime „Programmeerimine“, „Programmeerimise alused“ või „Programmeerimise algkursus“. Kui kursus oli jaotatud mitmesse osasse, lisandusid nimedele juurde ka järjekorranumbrid „I“ ja „II“.

1.2 Kursustel käsitletavad teemad

Vastavalt Murphy jt väitele [11:2] võib öelda, et programmeerimise algkursuseid, mida ülikoolides õpetatakse, ei ole väga palju uuritud. Nad toovad küll välja 2017. aastal avaldatud artiklis, et kuigi Austraalias ja Uus-Meremaal on pikka aega uuritud ülikoolides õpetatavaid programmeerimise algkursuseid, siis mujal samasuguseid uuringuid sellises mahus tehtud ei ole. Murphy jt teostasid sarnase uuringu Austraalias ja Uus-Meremaal läbiviiduga, et teada saada programmeerimise algkursuste õpetamisest Suurbritannia ülikoolides. Järgnevalt kasutatakse veel Austraalia ja Uus-Meremaa näitel 2016. aastal läbiviidud Masoni ja Simoni samateemalist uuringut [12]. Suurbritannia uuringus saadi lõplikud tulemused 80 ning Austraalias ja Uus-Meremaal 48 kursuse kohta.

Programmeerimise paradigmad jagunevad nelja kategooriasse: protseduuriline, objektorienteeritud, funktsionaalne ja loogiline programmeerimine. Kaks peamist paradigmat, millega õpetatakse programmeerimise algteadmiseid, on protseduuriline ja objektorienteeritud.

Suurbritannias läbiviidud uuringust selgus, et objektorienteeritud paradigmat õpetab 50% kursustest (40 kursust) ja protseduurilist 33,8% kursustest (27 kursust). Nimetatud jaotused ei pruugi edasi anda täpset olukorda, sest vastajatel paluti nimetada ainult üks paradigma, kuigi neid võib konkreetsel kursustel olla rohkem.

Austraalias ja Uus-Meremaal läbiviidud uuringus, kust saadi vastused 48 kursuse kohta, olid need näitajad vastupidised. Objektorienteeritud paradigma kasutab 33% kursustest (16 kursust) ja protseduurilist 50% kursustest (24 kursust). Kahe kursuse kohta vastati, et seal kasutatakse nii protseduurilist kui ka objektorienteeritud paradigma.

Mõlema uuringu järgi moodustavad kursused, kus õpetatakse ainult protseduurlist või objektorienteeritud paradigma, kogu kursuste arvust rohkem kui 80%. Neile järgnes funktsionaalne paradigma, mida Suurbritannias õpetab 8,8% kursustest (7 kursust) ning Austraalias ja Uus-Meremaal 8,3% kursustest (4 kursust). Funktsionaalprogrammeerimise õpetamiseks kasutati enamasti programmeerimiskeeli Haskell (Suurbritannias 2 kursust ning Austraalias ja Uus-Meremaal 1 kursus) ja Python (Suurbritannias 1 kursus ning Austraalias ja Uus-Meremaal 2 kursust). Loogilist paradigmat ei õpeta mitte ükski uuringutes osalenud programmeerimise algkursus.

Programmeerimise teemad, mida õpetatakse programmeerimise algkursustel on enamjaolt paradigmade ülesed. Need on teemad, mida otseselt või kaudselt saab kasutada mitmes programmeerimise paradigmas. Tew ja Guzdial analüüsisid 12 programmeerimise algkursuse õpikut, et selgitada välja teemad, mida käsitleda programmeerimise algkursustel [7]. Nimekiri programmeerimise algkursuste õpikutes käsitlevatest teemadest [7:99]:

- muutuja,
- sisend/väljund,
- rekursioon,
- aritmeetilised tehted (`+`, `-`, `*`, `/`),
- võrdlustehted (`>`, `<`, `>=`, `<=`, `==`, `!=`),
- loogikatehted (`and`, `or`),
- muutuja väärtustamine,
- tingimusavaldis,
- korduslause (`for`, `while`),
- mitmekordne korduslause,
- funktsioon,
- funktsiooni parameetrid,
- funktsiooni tagastusväärtus,
- andmetüübid (täisarv, ujukomaarv, tõeväärtus, sõne, massiiv, puu),

- objekt ja klass,
- konstruktor,
- isend ja lokaalne muutuja,
- get- ja set-meetodid,
- ülekatmine,
- polümorfism,
- pärilus.

Kursusel kasutatavate teemade valik sõltub kursusest endast. Kuigi eespool olev nimekiri on enamjaolt paradigmade ülene, siis viimased seitse teemat nimekirjast on iseloomulikud ainult objektorienteeritud programmeerimisele.

1.3 Kursustel kasutatavad programmeerimiskeeled

Eelpool nimetatud Austraalia ja Uus-Meremaa [12] ning Suurbritannia [11] artiklites uuriti ka, milliseid programmeerimiskeeli programmeerimise algkursustel õpetatakse. Lisaks nendele töödele kasutatakse järgnevates lõikudes Ameerika Ühendriikide näites Guo 2014. aastal läbiviidud uuringut, mis käsitles ainult programmeerimiskeelte populaarsust [13]. Seal uuriti 39 parima USA ülikooli 84 algkursust. Gou mainib ära, et nimekiri ülikoolidest pärineb U.S. News 2014. aasta järjestusel. Kokku nimetati Austraalia ja Uus-Meremaa uuringus 16 keelt, millega õpetatakse programmeerimise algteadmiseid, Suurbritannias oli see näitaja 13 ning USAs toodi välja vaid 7 kõige populaarsemat. Nelja riigi peale ühiselt nimetatud keeli oli neli: Python, Java, C ja MATLAB.

Programmeerimise algkursuste kõige populaarsemateks keelteks osutusid Python ja Java, samas erinevates riikides nende populaarsus erines. Tabelis 1 on ära toodud Pythoni ja Java osamäärad riigiti kõikide uuritud kursuste lõikes.

Tabel 1. Pythoni ja Java protsentuaalne jaotus riigiti kõikides uuritud kursustes.

Riik	**Python**	**Java**
Suurbritannia	17,5%	61,3%
Austraalia ja Uus-Meremaa	31%	31%
Ameerika Ühendriigid	41,7%	30,1%

Tabelist 1 ilmneb, et Suurbritannias domineerib programmeerimise algkursustel Java. Vaadates samu näitajaid teiste riikide kohta selgub, et USAs domineerib programmeerimise algkursustel Python ning Austraalias ja Uus-Meremaal on nimetatud keelte populaarsus võrdne.

USAs läbiviidud uuringus uuriti ainult, milliseid programmeerimiskeeli kasutatakse algkursustel. Suurbritannias ning Austraalias ja Uus-Meremaal uuriti ka põhjuseid konkreetse keele valiku kohta. Selleks paluti vastajatel nimekirjast potentsiaalsetest põhjustest hinnata igaühe olulisust. Antud tulemustest saadi kokku üldine hinnang erinevatele programmeerimiskeele valiku põhjendustele.

Peamised põhjendused keele valikuks olid:

- pedagoogiline kasulikkus,
- olulisus tööturul,
- kättesaadavus ja maksumus,
- tegemist on objektorienteeritud keelega.

Kaks kõige populaarsemat keelt õpetamaks programmeerimise algkursust olid Java ja Python. Võttes kokku Suurbritannias ning Austraalias ja Uus-Meremaal läbiviidud uuringutes nimetatud põhjendused konkreetse keele kasuks selgus, et Java ja Pythoni valiku põhjendused erinesid. Tabel 2 näitab mõlemast uuringust kombineeritud põhjendusi konkreetse keele kohta. Eraldi protsendina oli Austraalia ja Uus-Meremaa näites välja toomata Pythoni kohta käiv põhjendus „Abi online kogukonnast“ ning Java puhul „Kättesaadavus ja maksumus“. Käesolevas tabelis kajastatakse nende juures ainult Suurbritannias välja tulnud osakaalud.

Tabel 2. Keele valiku põhjendus konkreetse keele kohta. Sulgudes on märgitud kahe uuringu peale keskmine osakaal kõikidest vastanutest.

Python	**Java**
Pedagoogiline kasulikkus (69,9%)	Objektorienteeritud keel (86,6%)
Kättesaadavus ja maksumus (58,3%)	Nõudlus tööturul (75,8%)
Abi online kogukonnas (45,5%)	Kättesaadavus ja maksumus (55,3%)
Platvormist sõltumatus (42,7%)	Platvormist sõltumatus (50,2 %)

Mõlema keele puhul saab esile tuua domineerivamad põhjused selle valikuks. Java on peamine objektorienteeritud keel, mida õpetatakse, ning selle järele on tööturul ka suur nõudlus. Python on süntaktiliselt suhteliselt lihtne keel, seega on sellega lihtsam õpetada programmeerimise üldiseid teemasid, sest sellisel juhul ei valmista õppijale nende omandamisel nii suurt raskust keel ise.

Keelte populaarsus programmeerimise algkursustel ei ole stabiilne, vaid on aja jooksul muutunud. Guo toob välja 2014. aastal ilmunud uuringus [13], kus uuriti Ameerika Ühendriikide ülikoolide programmeerimise algkursuseid, et pikalt domineerinud Javat vahetakse järjest enam välja Pythoniga. Sarnane trend selgub ka Austraalias ja Uus-Meremaal, kus ülikoolides läbiviidavaid programmeerimise algkursuseid on uuritud juba aastast 2001. Kõige uuemast, 2017. aastal avaldatud Masoni ja Simoni uuringust [12] ning 2014. aastal avaldatud Masoni ja Cooperi uuringust [14] ilmneb, et Java on oma populaarsust kaotanud ning Python on Javale järgi jõudnud. Tabelis 3 on näha Pythoni ja Java populaarsuste muutust Austraalia ja Uus-Meremaa programmeerimise algkursustel. Sealt ilmneb, et kuigi viimasel kahel korral on Pythoni ja Java populaarsus võrdsed, siis Java populaarsus on languses ja Pythoni populaarsus tõusmas.

Tabel 3. Pythoni ja Java populaarsuste muutus Austraalia ja Uus-Meremaa programmeerimise algkursustel.

	2001	2003	2010	2013	2016
Python	0,0%	0,0%	13,6%	27,3%	31,0%
Java	40,4%	40,8%	36,4%	27,3%	31,0%

Sarnaselt eelnevatele näidetele, mindi ka Tartu Ülikooli programmeerimise algkursusel, 2009/2010. õppeaastal Javalt üle Pythonile [15]. Keele vahetuse põhjuseks toodi Java paljusõnalisust, kus lihtsa funktsionaalsuse saavutamiseks tuleb kirja panna palju koodi. Programmeerimise algteadmiseid omandavaid tudengeid võib selline keerukus pigem hirmutada, kui neid programmeerimise juurde tuua. Pythoni poolt toodi argumentideks selle lihtne süntaks ning Pythoni multi-paradigmalisus ehk sellega saab õpetada nii protseduurilist, objektorienteeritud kui ka funktsionaalset lähenemist programmeerimisele.

1.4 Kursuste eesmärgid

Suurbritannias ning Austraalias ja Uus-Meremaal läbiviidud uuringutes küsiti vastanutelt nimetada vabas vormis oma programmeerimise algkursuse kolm peamist eesmärki. Antud vastused kategoriseeriti, et teada saada enim levinud eesmärgid:

1. Õpetada programmeerimise fundamentaalseid konstruktsioone (42%).
2. Arendada õppijates probleemi lahendamise oskust (32,9%).
3. Arendada õppijates algoritmilist mõtlemist (32,6%) .
4. Õpetada programmi loogilist koostamist (25,2).
5. Õpetada konkreetse keele süntaksit (24,4%) .
6. Tekitada õppijates rõõm programmeerimises (22%).

Sulgudes on märgitud kahe uuringu peale kõikide vastanute keskmine osakaal. Antud nimekirjast selgub, et konkreetse keele õpetamine programmeerimise algkursusel on populaarsuselt alles viies eesmärk. Vaadates kõiki eesmärke, mis antud nimekirjas on sellest eespool ilmneb, et programmeerimise algkursuse põhiliseks eesmärgiks on arendada õppijas algoritmilist mõtlemist ning edasi anda keelteüleseid oskuseid, mida hilisemalt saaks kasutada juba keelespetsiifilisemas õppes.

1.5 Programmeerimise algkursused Eesti kõrgkoolides

Antud alapeatükk põhineb Peedoski 2014. aastal kaitstud bakalaureusetööl „Eesti kõrgkoolide programmeerimise algkursused“ [10].

Peedosk toob välja, et otsides programmeerimise algkursuseid Eesti kõrgkoolidest, leiti 16 erinevat kursust kümnest Eesti kõrgkoolist. Lisaks mainib ta, et vaatamata erinevatele otsimismeetoditele, ei pruugi leitu sisaldada kõiki Eesti kõrgkoolide programmeerimise algkursuseid. Vaadates tema poolt nimetatud kõrgkoolides õpetatavaid kursuseid selgub, et 2018. aasta märtsis õpetatakse

endiselt enamikke kursustest. Nendest kursustest, mida õpetatakse, on mõnel muutunud aine maht või kursusel kasutatav programmeerimiskeel.

Oma bakalaureusetöös teeb Peedosk üldiseid kokkuvõtteid Eesti kõrgkoolides õpetatavate programmeerimise algkursuste kohta. Sealt selgub, et IT õppekavadel õpetatavad programmeerimise algkursused viiakse läbi enamasti kas esimesel või teisel semestril ehk õpingute alguses. Kursuste maht jääb vahemikku 2-6 EAP, kuid IT õppekavadele mõeldud kursused on enamasti 4-6 EAP. Domineerivamaks keeleks algkursustel osutus Python, samas paljudes koolides, kus alustati Pythoniga, oli jätkuaineks objektorienteeritud programmeerimine, mida õpetati enamasti keeles Java. Kuna tegemist on programmeerimise algkursustega, siis peale mõne erandi, tudengite eelteadmistega ei arvestata. Peedosk toob välja, et kursustel, kus arvestatakse varasemat kogemust omavate tudengitega, lubatakse tudengitel kursus varem ära lõpetada, neile antakse raskemaid ülesandeid, nad ei pea tundides kohal käima või on neile loodud süvendatud kursus.

2. Eksamid

Käesolevas peatükis antakse ülevaade kujundavast ja kokkuvõtvast hindamisest ning sellest, kuidas eksam täidab kokkuvõtva hindamise rolli. Seejärel analüüsitakse programmeerimise algkursuste eksameid. Selleks kirjeldatakse, kuidas valmivad programmeerimise algkursuste eksamid ning mis teemasid seal käsitletakse.

2.1 Hindamine

Järgnev osa põhineb Biggsi ja Tangi raamatul „Õppimist väärtustav õpetamine ülikoolis: keskmes õppija tegevused“ [16].

Biggs ja Tang toovad välja, et peamised hindamise põhjused on kujundava tagasiside andmine (inglise keeles *formative feedback*) ja kokkuvõtva hinde panemine (inglise keeles *summative grading*). Kuigi enamasti nimetatakse mõlemat hindamiseks (inglise keeles *assessment*), on nende eesmärgid erinevad [16:170].

Kujundava hindamise eesmärgiks on anda õppija omandatud teadmiste ja oskuste kohta tagasisidet õppimise jooksul. Antud tagasisidest saab õppija teada, kui hästi tal läheb, ning milles on puudujääke. Selle tulemusel saab õppija end täiendada, et likvideerida teada saadud puudused [16:170]. Lisaks õppija arendamisele saab kujundav hindamine parandada ka õpetamist. Biggs ja Tang toovad välja, et mida suurem on loengutes saadava tagasiside osakaal, seda tõhusamad on loengud ja seda aktiivsemad on üliõpilased õppima.

Kokkuvõtva hindamise eesmärgiks on fikseerida, kui hästi üliõpilased õppisid seda, mida nad pidid õppima [16:171]. Kokkuvõttev hindamine viiakse läbi kursuse lõpus, kui õppeprotsess on lõppenud. Kokkuvõtva hinnangu tulemuseks on enamasti hinne, pärast mille teada saamist enam õppimist ei toimu. Selleks, et saada parem hinne, üritavad üliõpilased oma vigu varjata. See on kujundava- ja kokkuvõtva hindamise üks suurimaid erinevusi.

Kui kokkuvõtval hindamisel kipuvad õppijad oma vigu varjama, et saada parem hinne, siis kujundaval hindamisel peaksid nad vead välja tooma, sest läbi nende vigade toimub õppimine. Samas kasutatakse kursuse jooksul hinnatavaid vahendeid kokkuvõtva hinde panemiseks. See tekitab vastuolus, kus õppija peab oma vigu välja tooma, et paremini õppida ning samas vigu varjama, et saada parem lõplik hinne.

2.2 Eksam, kui kokkuvõtva hindamise vahend

Eksamiks võib pidada struktureeritud kirjalikku tööd, mis viiakse reeglina läbi kontrollitud tingimustes, piiratud aja jooksul [16, 17]. Selle eesmärgiks, täites kokkuvõtva hindamise rolli, on anda hinnang, kui hästi õppija on omandanud õpiväljundid. Biggs ja Tang [16] paigutavad eksami deklaratiivsete teadmiste (need on teadmised millegi kohta, ehk „millegi teadmine“ [16:73]) kokkuvõtva hindamise vahendiks. Eksami struktureeritus tagab selle, et küsimused on konkreetsemad ja kontrollivad täpselt seda, mida eksamineerija soovib teada saada. Kontrollitud

oludes eksami korraldamine vähendab akadeemilist petturlust, sest eksamineerija jälgimisel on spikerdada ja kellegi teise vastustega vastata keerulisem, kui kontrollimata oludes tööd kirjutades.

Kirjalik eksam võib koosneda mitmetest erinevatest küsimuste liikidest. Pilli toob välja raamatus „Väljundipõhine hindamine kõrgkoolis“ [17] kaheksa erinevat küsimuste liiki, nagu näiteks valikvastustega- ja kirjalikku (lühi)vastust nõudvad küsimused. Lisaks sellele mainitakse ära ka liik „Ülesannete lahendamine“. Programmeerimise algkursuste eksamitel kasutatakse ülesanneteks programmeerimisülesandeid. Täpsemalt liikidest ja nende kasutamisest programmeerimise algkursuste eksamitel antakse ülevaade antud bakalaureusetöö neljandas peatükis.

Eksami, kui piiratud aja jooksul läbiviidava kirjaliku töö ohtudeks tuuakse selle pealiskaudsust. Nii Pilli [17], kui ka Biggs ja Tang [16] toovad välja, et kirjalik eksam suudab hinnata pigem deklaratiivseid teadmiseid. Kuigi eksamineerija võib kasutada avatumaid küsimusi, mis paneb eksaminandid olukorda, kus tuleks argumenteerida kõrgema taseme teadmistega (teadmised, kus kasutatakse deklaratiivseid teadmiseid üldisema probleemi lahendamiseks), siis ajapiirang ei pruugi selleks aega jätta. Antud probleemi aitab lahendada avatud materjalidega eksam, kus eksaminandil lubatakse eksami ajal kasutada teatud abivahendeid. Nendeks võivad olla erinevad raamatud, õpikud, eksaminandi enda konspektid kui ka Internetist abi otsimine. Täpsem ülevaade avatud ja suletud materjalidega eksamitest antakse käesoleva bakalaureusetöö kolmandas peatükis.

Programmeerimise algkursuse eksamil saab eelmises lõigus mainitud probleemi lahendada kasutades programmeerimisülesandeid. Need on ülesanded, kus eksaminandil tuleb vastuseks esitada programm, mis sisaldab ülesandes nõutud funktsionaalsust. Sedasi saab eksaminand kokku koondada oma teadmised programmeerimisest, ning demonstreerida nendele toetudes oma programmeerimisoskust. Täpsemalt antakse ülevaade erinevatest küsimuste tüüpidest programmeerimise algkursuste eksamites bakalaureusetöö neljandas peatükis.

2.3 Programmeerimise algkursuste eksamid

Järgnev osa põhineb Sheard jt 2013. aastal ilmunud artiklil „*Assessment of programming: pedagogical foundations of exams*“ [18]. Seal viidi läbi uuring programmeerimise algkursuste õppejõududega, et selgitada välja, kuidas valmivad programmeerimise algkursuste eksamid.

Sheard jt [18] toovad välja, et enamasti kasutatakse programmeerimise algkursuste eksamiteks paberil sooritatavaid eksameid. Seal on nii teoreetilisemaid küsimusi, kui ka paberil koodi kirjutamist. Bennedsen ja Caspersen väidavad, et kuna kursuse jooksul ja ka tegelikus elus programmeeritakse arvutis, siis paberil eksam on kunstlik keskkond ja see pole sobilik hindamaks üliõpilase programmeerimisoskust [19:188].

Oma eesmärgi poolest ei erine programmeerimise algkursuste eksamid, eespool nimetatud eksami peamisest eesmärgist, milleks on hinnata seda, kuidas õppijad omandasid kursuse õpiväljundid. Nagu selgus esimeses peatükis, siis programmeerimise algkursuste peamiseks eesmärgiks on õpetada algoritmilist mõtlemist ning programmeerimise üldisemaid teemasid, samas on olulisel

kohal ka programmeerimise oskus. Haghighi ja Sheard jõudsid järeldusele, et kõige ausamalt annab edasi õppija oskuseid ja olemasolevaid teadmiseid programmeerimises eksam, mis koosneb nii paberil kui ka arvutis vastatavatest küsimustest [20]. Edaspidi nimetatakse käesolevas bakalaureusetöös selliseid eksamiosasid vastavalt eksami paberosaks ja arvutiosaks. Täpsemalt antakse ülevaade eksami paberosas ja arvutiosas kasutatavatest küsimustest bakalaureusetöö neljandas peatükis.

Sheard jt [18] toovad välja, et vähesed programmeerimise algkursuste eksamid luuakse nullist. Enamasti võetakse eeskuju varasemate aastate eksamitest, milles muudetakse küsimusi. Eksamites kasutatavad küsimused on igal eksamil reeglina uued, samas leidub ka eksameid, mis sisaldavad juba varem kasutatud küsimusi. Mõned uuringus osalenud õppejõud mainisid, et nende koostatud eksam koosnes kursuse jooksul olnud kodustest töödest või kasutatud näiteülesannetest, selleks et välja selgitada need, kes kursuse jooksul kaasa ei teinud. Paljud õppejõud toetuvad oma küsimustes mõnele taksonoomiale, näiteks Bloomi taksonoomiale. Taksonoomiale toetumise põhjustena toodi välja, et eksam kontrolliks teadmiseid kõigi vastava taksonoomia astmete kohta. Eksami koostamise protsessist toodi välja, et eksami koostab aine õppejõud reeglina üksinda. Pooled vastanutest väitsid, et eksami koostamine on keeruline tegevus ja ajapuudusel ei saa head eksamit teha.

Eksamitulemusi võib mõjutada see, kuidas õppija selleks valmistub. Sheard jt [18] läbiviidud uuringust selgus, kuidas programmeerimise algkursuste õppejõud oma kursuste osalejaid eksamiks ette valmistasid. Mõned õppejõud viisid läbi kordamisloengu, reeglina kursuse lõpuosas, selleks et kokku võtta kursusel õpitu ning anda ülevaade eksamil küsitavatest teemadest ja eksami üldisest stiilist.

Paljud õppejõud andsid kursusel osalejatele eksamiks õppimiseks kas proovieksami või mõne varasema aasta eksami. Seda tehti selleks, et anda ülevaade tulevase eksami stiilist, ning et oleks näiteküsimusi, mille põhjal korrata. Mõned õppejõud andsid koos eelmiste aastate eksamitega kaasa ka vastused, kuid ülejäänud õppejõud vastuseid ei andnud, tuues põhjuseks, et lihtsalt vastuste pähe õppimisega kasu ei saavutata.

Lisaks varasemate eksamite andmisele, koostasid mõned õppejõud kordamiseks mõeldud veebiteste. Neile vastates said õppijad eksamiks korrata. Selleks, et sellist õppimisviisi hõlbustada, said õppijad testi tehes kohest tagasiside oma vastuste õigsuse kohta ning neil oli võimalus testi korduvalt lahendada.

Programmeerimise algkursustel käsitletavatest programmeerimise teemadest toodi ülevaade antud bakalaureusetöö esimeses peatükis. Eksamites käsitletavad teemad ühtivad kursustel käsitletavate teemadega. Simon jt uurisid programmeerimise algkursuste eksameid Austraalias, Uus-Meremaal kui ka Ameerika Ühendriikides [21]. Nad tõid välja teemad, mida käsitletakse programmeerimise algkursuste eksamites. Järgnevas nimekirjas näitab sulgudes olev number, kui mitu protsenti programmerimise algkursuste eksamitest käsitlesid konkreetset teemat.

Kõige populaarsemateks teemadeks osutusid:

- objektorienteeritud programmeerimise konstruktsioonid (35,8%),
- meetod/funktsioon (34,5%),
- korduslause (32,3%),
- järjend (26,3%),
- programmi disain (16,9%),
- sisend/väljund (12,3%),
- tingimusavaldis (11,3%).

Välja on toodud teemad, mis esinesid rohkem kui kümnel protsendil kõikidest uuritud eksamitest.

Nimetatud teemades ei esine andmetüüpe ja muutujaid – nende populaarsus eksamites oli palju väiksem (4,4%). See võib tuleneda sellest, et ükski ülesanne ei kontrolli otseselt nende teemade valdamist. Kuna tegemist on programmeerimise fundamentaalsete teemadega, siis enamik programmeerimist nõudvatest küsimustest sisaldab nende kasutamist. Samal põhjusel ei ole eraldi välja toodud ka loogikatehteid. Enamik kohti, kus neid kasutatakse, on tingimusavaldised ja eelkontrolliga korduslaused, seega loogikatehete oskamist kontrollitakse läbi nende konstruktsioonide.

Tew ja Guzdial [7] analüüsisid 12 programmeerimise õpikut, selgitamaks välja teemad, mida käsitleda programmeerimise algkursustel. Nimekiri nendest teemadest toodi välja bakalaureusetöö esimeses peatükis. Nad mainisid, et kõiki neid teemasid ei ole võimalik ühte eksamisse paigutada, seega tõid nad välja kümme teemat, mida kasutada programmeerimise algkursuste eksamites [7:100]:

- fundamentaalsed konstruktsioonid (muutujad, väärtustamine jne),
- loogikatehted,
- tingimusavaldis,
- *for*-korduslause,
- *while*-korduslause,
- järjend,
- funktsiooni/meetodi argumendid,
- funktsiooni/meetodi tagastusväärtus,
- rekursioon,
- objektorienteeritud programmeerimise konstruktsioonid.

Eespool olevatest uuringutest välja toodud nimekirjad ühtivad suuresti. Ainsaks erinevuseks on Simoni jt koostatud nimekirjas olevad „programmi disain“ ja „sisend/väljund“ ning Tew’i ja Guzdiali nimekirjas olevad „rekursioon“ ja „loogikatehted“. Kuigi loogikatehted esinesid ka Simoni jt uuritud eksamites, siis seal oli nende otsene kasutatavus väga madal.

3. Eksamite läbiviimise vormid

Käesolevas peatükis antakse ülevaade erinevatest eksami läbiviimise vormidest. Selleks võrreldakse avatud ja suletud materjalidega eksamit ning tuuakse välja mõlema eeliseid ja puuduseid. Lisaks antakse ülevaade õppija enda koostatud abilehe kasutamisest eksamil, tuuakse välja selle eeliseid ja puuduseid ning antakse nõuandeid abilehe koostamiseks.

3.1. Suletud materjalidega eksam

Suletud materjalidega eksam tähendab eksamit tema klassikalises mõistes. Nagu eelmises peatükis välja toodi, on eksam struktureeritud kirjalik töö, mis viiakse reeglina läbi kontrollitud tingimustes, piiratud aja jooksul. Sellisel eksamil on nii positiivseid, kui ka negatiivseid omadusi.

Biggs ja Tang [16] toovad välja, et viies eksamit läbi kontrollitud tingimustes, saab vähendada spikerdamist. Nagu selgub Sheard jt [18] läbiviidud uuringust, siis on ülikoole, kus see on ka ainuke põhjus, miks eksameid sellises formaadis korraldatakse. Eksamil spikerdamist aitab vähendada ka eksamil materjalide kasutada lubamine, millest antakse ülevaade järgnevates alapeatükkides.

Viies eksamit läbi suletud materjalidega, saab eksaminand näidata ainult enda teadmiseid, mille ta on eksamiks omandanud ning see annab ta teadmistest väga hea ülevaate. Biggs ja Tang [16] toovad välja, et kuigi see on hea võimalus hinnata ainult seda, mida eksaminand teab, siis eksami piiratud tingimustes ei pruugi olla aega ega võimalust oma laiapõhiste teadmiste demonstreerimiseks. Kuna eksam on ajaliselt piiratud, siis eksaminand vastab küsimustele kõige olulisemate faktidega, et näidata oma teadmist antud küsimuse juures. See võib jätta vastused pealiskaudseteks.

Pealiskaudsete vastusteni viib juba eksamiks õppimine, kui õppija peab kõige olulisemaks faktide meelde jätmist, mitte õpitavast teemast arusaamist. Selleks, et vähendada faktide pähe õppimist, võib lasta eksami kirjutada sinna kaasa võetud materjalidega. Sedasi saab eksamiks valmistudes süveneda teemast arusaamisele, sest faktiteadmised tulevad eksamile kaasa võetud materjalidest. Järgnevalt antakse ülevaade avatud materjalidega eksamist ning õppija enda koostatud abilehe kasutamisest eksamil.

3.2. Avatud materjalidega eksam

Avatud materjalidega eksami (inglise keeles *open-book examination*) korral on eksaminandil lubatud eksamile kaasa võtta teatud abistavaid materjale. Nendeks võivad olla õpikud, loengukonspektid, õppejõu antud märkmed jms. Eksamil võib olla ka lubatud kasutada Internetti, kui avatud materjali. Eksaminand peaks olema eelnevalt tutvunud eksamile võetud materjalidega, et eksamit kirjutades nendest kiiremini vajalik informatsioon üles leida.

Biggs ja Tang [16] toovad välja, et eksamil, tema klassikalises tähenduses, on mõningaid puudusi. Üheks selliseks on eksami võimekus kontrollida ainult deklaratiivsemaid teadmiseid. Seda süvendab asjaolu, et ajaliselt piiratud eksamil on eksaminandil keeruline vastustega väga süvitsi

minna ja seega võib tulemus jääda pealiskaudseks. Biggs ja Tang väidavad, et avatud materjalidega eksam leevendab pealiskaudsust, sest eksaminandid ei pea peast teadma erinevaid fakte, ning seega on nad võimelisemad vastama kõrgemal tasemel.

Tussing toob välja 1951. aastal ilmunud artiklis [22], et avatud materjalidega eksamil on teisigi eeliseid. Näiteks asjaolu, et eksamiks ei pea kõiki üksikasju pähe õppima, vähendab eksaminandides ärevust ja hirmu eksami suhtes. Lisaks toob ta välja, et üksikuid fakte pähe õppimata saab eksamiks õppides rohkem rõhku panna teemadest arusaamisele, kui teemade pähe õppimisele. Materjalide eksamile kaasa võtmine vähendab ka eksamil spikerdamist, sest selleks kaob vajadus.

Soloway tõi välja 1986. aastal ilmunud artiklis [23], et ka päris elus ei tea programmeerijad peast kõiki programmeerimiskeele peensusi ja oma töös kasutavad nad erinevaid abimaterjale. Sama kehtib ka tänapäeval, kus programmeerijad otsivad infot erinevate teekide kohta nende dokumentatsioonides või kellegi poolt tehtud koodinäidetes. Seega võiks programmeerimise eksamitel lubada kasutada abistavaid materjale, et ei peaks pähe õppima programmeerimiskeele spetsiifilisi nüansse.

Esimeses peatükis selgus, et programmeerimise algkursuse peamiseks eesmärgiks on arendada õppijas algoritmilist mõtlemist ning edasi anda keelteüleseid oskuseid. Seega selleks, et nimetatud eesmärki täita, ei pea eksaminand peast teadma programmeerimiskeele spetsiifilisi teadmiseid. Piisab sellest, kui ta oskab leida informatsiooni selle kohta, kuidas mingis konkreetses keeles teatud algoritmi või konstruktsiooni kasutada.

Kuigi materjalide kasutamine eksamil võib mõjuda positiivselt, on sel ka negatiivseid mõjusid. Feldhusen [24] väidab, et avatud materjalidega eksamiks valmistutakse vähem, seega eksamile minnes omatakse vähem teadmiseid. Eksaminandid loodavad, et materjale kasutades on lihtsam eksamil parem tulemus saada ning nii palju õppima ei pea. Samas, materjale valdamata on nendest keeruline sobivat kohta üles leida. Selle tõi välja Boniface [25], kes leidis, et üliõpilased, kes kasutasid eksamil rohkem materjale, said keskmiselt madalama hinde. Ta pani tähele, et need on üliõpilased, kellel üldine hinne on madalam, seega nad panustasid pigem materjalidega eksami lahendamisele. Antud asjaolu võib tulla ka sellest, et üliõpilastele olid materjalid võõrad, seega sealt sobiva info otsimisele kulus rohkem aega ning seega jäi vähem aega eksami kirjutamiseks.

Eelnevast selgus, et eksamil materjalide kasutamise suurimaks probleemiks on asjaolu, kus materjalid on võõrad, ehk see, kes neid kasutab, pole neid ise kirjutanud, ning ei pruugi olla enne eksamit ka nendega tutvunud. Võõrast materjalist sobiva info leidmine võtab aega ja seega jääb vähem aega eksami kirjutamiseks. Antud probleemi üritab lahendada õppija enda koostatud abilehe kasutamine eksamil. Järgnevalt antakse sellest ülevaade.

3.3. Õppija enda koostatud abilehe kasutamine eksamil

Õppija enda koostatud abileht (inglise keeles *cheat-sheet*) on abimaterjal, mille ta on endale eksami sooritamiseks koostanud. Sinna võib kirjutatud olla kõik, mida õppija peab vajalikuks eksami kirjutamiseks. De Raadt [26] toob välja, et üldiselt on see kahepoolne A4 paber, mille eksaminand on oma käega kirjutanud.

Abilehe ise ja käsitsi kirjutamine ongi kõige suurem erinevus võrreldes abilehe kasutamist eksamil ja avatud materjalidega eksamit. Kui abileht on eksaminandi enda koostatud, siis selle valmimisel on ta eksamiks vajalikud materjalid läbi töötanud. Käsitsi kirjutamine garanteerib, et materjal ei ole lihtsalt kopeeritud. Seega pole abilehest kasu mitte ainult eksamit kirjutades, vaid abilehe koostamine aitab ka eksamiks valmistuda. Sama märkis ka Erbe [27]. Võttes kasutusele abilehed selgus, et üliõpilastele need meeldisid. Siiski eksamil need eriti kasutust ei leidnud. Erbe tõi välja, et abilehe koostamine õpetas õppijaid piisavalt ning sinna kirjutatud teadmised olid eksamit tehes juba selged.

De Raadt [26] viis läbi uurimuse, selgitamaks välja abilehtede kasutamise kasulikkuse programmeerimise algkursuste eksamitel. Selgus, et abilehe kasutamine mõjub positiivselt eksaminandi eksamihindele. Keskmiselt tõstis abilehe kasutamine eksamihinnet 10% ja abilehe mittekasutamine langetas eksamihinnet 4,5%. Koos eksamiga tuli eksaminandidel esitada ka abileht, mille nad koostanud olid, et neid analüüsida. Sealt selgusid kõige kasulikumad teemad, mida abilehele kirjutada.

Kasutades väga palju infot täis kirjutatud abilehte, oli eksaminandi tulemus eksamil ainult 1% parem, kui seda oli eksami keskmine tulemus. Sellest võib järeldada, et ainult info kvantiteet ei olnud abiks eksamit kirjutades. Tulemust mõjutas ka info organiseeritus. Seda väidet toetab ka asjaolu, kus järjestades abilehel olevad teemad sarnasesse järjekorda nagu neid õpetati kursusel, oli eksami tulemus keskmisest 13% parem.

Abilehele koondatud materjalide sisulisest poolest osutus kasulikumaks kirja panna abstraktsemad programmeerimise konstruktsioonid, kui konkreetsed koodinäited. Eksaminantidel, kes olid abilehele kirjutanud ainult koodinäiteid, oli eksami tulemus keskmisest 7% madalam. Nendel, kes olid abilehele kirjutanud koodinäidete asemel abstraktsemaid konstruktsioone, oli eksami tulemus 21% kõrgem, kui keskmine tulemus. De Raadt järeldas sellest, et abstraktsemaid teadmiseid saab rakendada paremini eksamiülesannetele millega eksaminand ei pruugi olla varem kokku puutunud. Seega konkreetsed koodinäited antud ülesannete juures kasuks ei tule.

Abilehe kasutamine eksamil on hea võimalus, kuidas ära kasutada avatud materjalidega eksami eelised, likvideerides vähemalt osad selle puuduseid. Abilehe kasutamine eksamil vähendab ärevust ja hirmu eksamile minnes. Lisaks vähendab abilehe kasutamine üksikute teadmiste pähe õppimist, lastes eksamil kontrollida ka teadmiseid kõrgemal tasemel.

Eelnevalt sai välja toodud avatud materjalidega eksami puudus, kus materjalides mitte orienteeruv eksaminand ei leia sealt eksamil abi ja seega ta tulemus ajapuudusel halveneb. Kuigi sama võib

juhtuda ka abilehte kasutades, on see oht väiksem. Esiteks, nagu Erbe [27] oma näites tõi välja, siis abilehte koostades õpitakse ja kinnistatakse kursusel õpitud materjale. Eksamile õpikut või loengukonspekte kaasa võttes ei pruugita seda teha ehk avatud materjalidega eksamile minnes ei pruugi olla eksaminand materjale eelnevalt läbi töötanud. Teiseks, abileht on reeglina üks A4 leht, mis on mahult palju väiksem, kui näiteks õpik, seega on sealt info leidmine lihtsam, kui mahukast õpikust. Lisaks, kuna eksaminand on oma abilehe ise koostanud, oskab ta sealt infot paremini leida.

4. Küsimuste liigid programmeerimise algkursuste eksamites

Käesolevas peatükis analüüsitakse teaduslikke artikleid, mis kirjeldavad programmeerimise algkursuste eksamites kasutatavaid küsimuste liike. Selle põhjal tuuakse välja võrdlus erinevate küsimuste liikide vahel. Iga liigi kohta esitatakse ka seda liiki illustreeriv näidisülesanne. Antud peatükis nimetatud küsimuste liigid ei ole kindlasti lõplik nimistu nendest. Tegemist on pigem populaarsemate liikidega, mis esinevad programmeerimise algkursuste eksamites.

4.1 Küsimuste liigid

Eksam, kui struktureeritud kirjalik töö, võib koosneda paljudest erinevat liiki küsimustest. Nagu teises peatükis mainitud, toob Pilli [17:53-58] välja kaheksa erinevat küsimuste liiki, mida saab kasutada kirjalikus eksamis. Nendeks on:

- valikvastustega küsimused,
- alternatiivküsimused, ehk jah/ei küsimused,
- sobitamise küsimused,
- järjestamise või ümberpaigutamise küsimused,
- vabavastuselised küsimused,
- lünkülesanded,
- ülesannete lahendamine,
- esseevastustega küsimused.

Sõltuvalt õppeainest, kus konkreetne eksam läbi viiakse, võib erineda küsimuste liikide valik. Peterseni jt [28] ning Simoni jt [21] läbiviidud uuringutest, kus analüüsiti programmeerimise algkursuste eksameid, tõusid teiste seast esile kolm küsimuste liiki. Kõige populaarsemateks osutusid koodikirjutamise ülesanded, mis eelnevas nimekirjas paigutuvad ülesannete lahendamise alla. Mõlemas uuringus tuli välja, et need moodustavad keskmiselt 54% eksami hindest. Populaarsuselt järgmised olid vabavastuselised küsimused (vastavalt 36% ja 28%) ning valikvastustega küsimused (vastavalt 7% ja 17%).

Mõlemad uuringud tõid lisaks küsimuste liikidele välja ka oskused, mida läheb vaja, et antud küsimustele vastata. Simon jt [21:66] tõid välja, et kui kokku kombineerida kõik programmeerimisega seotud oskused (koodi kirjutamine, -mõistmine, -seletamine, -silumine ja -muutmine), siis need katavad kokku 81% eksamist.

Antud tulemused näitavad, et kuigi koodi kirjutamine moodustab poole programmeerimise algkursuse eksamist, siis konkreetselt koodiga seotud ülesannete osakaal on palju suurem. Lisaks sellele, et eksam sisaldas koodi kirjutamise ülesandeid, sisaldas see ka teisi koodiga seotud ülesandeid, näiteks koodi mõistmist või koodi seletamist.

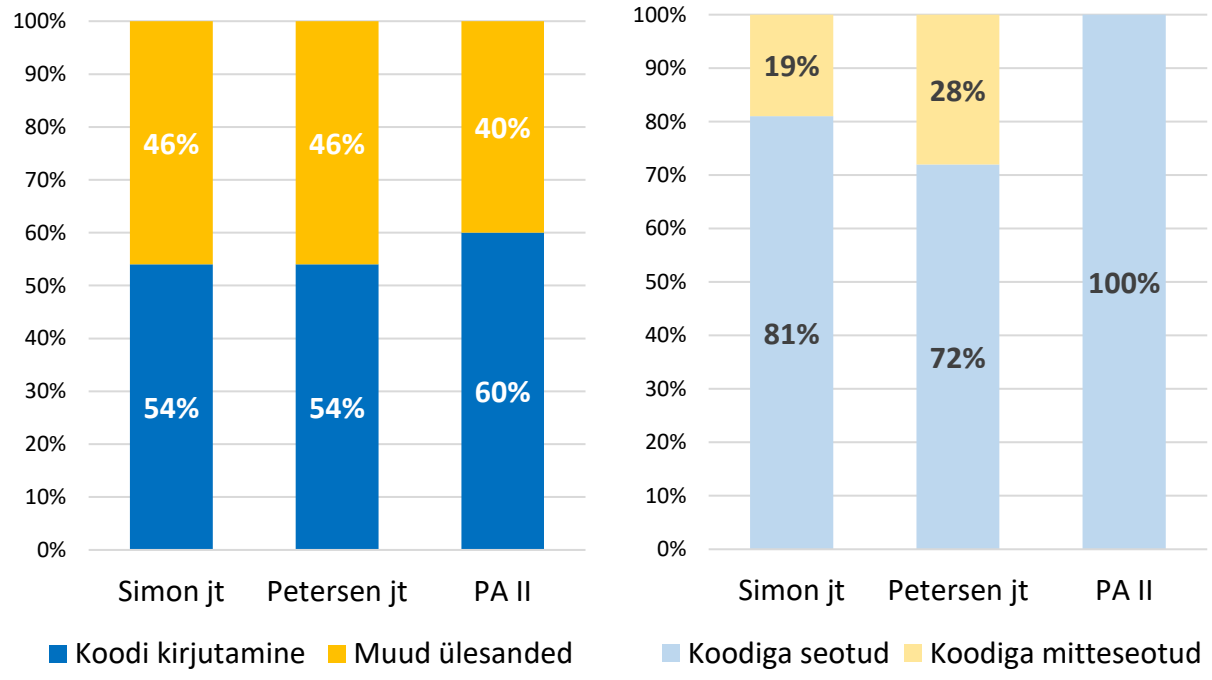
Sama toovad välja ka Petersen jt [28]. Nende uuritud programmeerimise algkursuste eksamites moodustasid küsimused, mis olid koodiga seotud, kuid ei olnud koodi kirjutamine, 19% kogu eksami hindest. Sellest suurem osa, 14%, moodustas vabavastuse vormis küsimused, 3%

valikvastustega küsimused, ning ülejäänud 2% diagrammide joonistamine, koodist arusaamist nõudvates ülesannetes.

Sarnane ülesannete sisuline jaotus tuli välja ka Tartu Ülikooli e-kursuse „Programmeerimise alused II“ eksamist. Seal moodustas 60% eksamihindest koodi kirjutamise ülesanded ja 40% hindest koodi mõistmist nõudvad ülesanded. Antud eksam koosnes kahest osast, paberil lahendatavast paberosast ja arvutis lahendatavast arvutiosast. Paberosa koosnes viiest ülesandest, kus iga ülesanne andis 10% lõpp-punktidest. Ühel ülesandel tuli vastusena kirjutada kood ning ülejäänud neljal ülesandel tuli vastusena kirjutada ette antud programmijupi väljundid. Antud eksami arvutiosa koosnes kolmest programmeerimise ülesandest, kus tuli kirjutada programm, konkreetset probleemi lahendavale ülesandele.

Tartu Ülikooli kursuse „Programmeerimine“ eksam koosneb ainult valikvastusküsimustest. Antud aines nimetatakse seda lõputestiks. Lisaks lõputestile, toimub kursuse jooksul ka kaks vaheeksamit, mis koosnevad ainult arvutis lahendatavatest programmeerimise ülesannetest.

Erinevate eksamite sisulist jaotust näitab Joonis 1. Seal on välja toodud kaks diagrammi, mis näitavad tulemusi erinevate uuringute ja eksamite kohta. Vasakpoolne diagramm näitab, kui suure osa eksami tulemustest moodustavad konkreetselt koodi kirjutamise ülesanded, ja kui suure osa koodi kirjutamist mitte nõudvad ülesanded. Parempoolne diagramm näitab, kui suure osa eksami tulemustest moodustavad kokku kõik koodiga seotud ülesanded ja kui suure osa koodiga mitteseotud ülesanded. Diagrammidel tähendab „PA II“ kursust „Programmeerimise alused II“.



Joonis 1. Eksamiülesannete sisuline jaotus.

Eelnevast selgus, et programmeerimise algkursuste eksamid võivad koosneda kahest osast. Kahe eksamiosa olulisust väitsid ka Haghighi ja Sheard [20], mille täpsem ülevaade oli bakalaureusetöö teises peatükis. Nendeks osadeks on paberil lahendatav paberosa ning arvutis lahendatav arvutiosa. Sellistes eksamites võib paberosa koosneda nii koodi kirjutamise ülesannetest, kui ka ülesannetest, milles ei tule koodi kirjutada. Arvutiosa koosneb enamasti koodi kirjutamise ülesannetest. Erinevus paberil ja arvutis lahendatavatest koodi kirjutamise ülesannetest tuleneb nende mahust.

4.2 Küsimuste liigid paberosas

Antud jaotuses antakse ülevaade programmeerimise algkursuse paberil lahendatavast eksami paberosast. Tuuakse välja erinevaid küsimuseliike, millega kontrollida teadmiseid programmeerimise algkursuste eksamites. Igale liigile tuuakse antud liigi kohta ka näiteküsimus.

4.2.1 Valikvastustega küsimused

Valikvastustega küsimused (inglise keeles *multiple-choice questions*) on küsimused, millele järgnevad vastusevariandid, millest vastaja peab valima ühe või mitu õiget vastusevarianti antud küsimusele. Pilli [17] sõnul peab olema välja toodud, kas õigeid vastuseid on üks või mitu.

Biggs ja Tang [16] toovad välja, et valikvastustega küsimused on väga levinud. Nende populaarsuse tagab nende hindamise lihtsus. Sama toovad välja ka Shuhidan jt [29], kus programmeerimise algkursuste õppejõududega läbiviidud uuringus selgus, et valikvastustega küsimuste hindamise lihtsus on peamine põhjus nende kasutamiseks. Eksami kontrollija peab ainult vaatama, kas vastaja on valinud õige variandi või mitte. Selle põhjal saab väga lihtsalt kokku lugeda õiged vastused ja eksamile tulemuse määrata. Selline hindamise lihtsus teeb valikvastustega küsimused ka väga kergesti hinnatavaks arvutile, tänu millele saab ainult valikvastustest koosnevaid eksameid teostada ka arvutil, võimaldades vastajale kohest tagasiside eksami tulemuste kohta.

Kuigi valikvastustega küsimused teevad eksami läbiviimise lihtsamaks, võib see jätta eksami sisulise kvaliteedi madalaks. Biggs ja Tang [16] väidavad, et valikvastustega küsimused hindavad deklaratiivseid teadmiseid, tavaliselt ainult äratundmise tasemel. Kui üliõpilase jaoks tundub midagi vastusevariantides tuttavat, kipub ta seda eelistama.

Lisaks soodustavad valikvastustega küsimused eksamimängurlust ehk vastuse ära arvamist [16:212]. Biggs ja Tang nimetavad mitmeid strateegiaid nii eksamineerijale, kui ka eksaminandile, kuidas koostada paremaid küsimusi ja kuidas õige vastus ära arvata.

Üks võimalus, et jõuda sobiva vastuseni, on välistamismeetod, ehk vastaja võib proovida välistada vastusevariante, mis tunduvad kindlasti valed. Et sellist välistamist keerulisemaks muuta, tuleks vastusevariante luues sellega arvestada. Pilli [17] toob välja, et valed valikvastused võiksid olla tüüpilised valevastused. Samasuguse strateegia toovad välja ka Shuhidan jt [29], kus lisaks paljule muule, uuriti programmeerimise algkursuste õppejõududelt nende meetodeid valimaks sobivaid vastusevariante valikvastustega küsimustesse. Sealt selgus, et enamasti valivad õppejõud valedeks vastusteks kas tüüpilisemad vead, mida üliõpilased varem on teinud või vastused, mis on

võimalikult sarnased õige vastusega. Nii vähendatakse võimalust, kus ainult valede vastuste välistamisega ja seejärel alles jäänud variantidest õige valimisega saaks õppija anda õige vastuse, vastust tegelikult teadmata.

Pilli [17] lisab, et valikvastustega küsimustele võib juurde lisada täiendavaid küsimusi. Vastajalt võib paluda põhjendada oma valikut või selgitada, miks tema arvates teised vastusevariandid õiged ei ole. Paludes vastajal põhjendada oma valikut, saab ülesande hindamisel arvesse võtta ka põhjuse adekvaatsuse. Selliselt saab vähendada vastajates õige vastuse ära arvamist. Kui ta teab vastust, siis ta oskab ka seletuse tuua, miks see on õige vastus. Vastuse kohta seletuse küsimise juures tuleb tähele panna, et mitte iga küsimuse kohta ei saa seletust küsida. Sellised küsimused on näiteks faktiteadmised. Kui lasta valikvastusküsimuses vastajal välja toodud valemite seast valida Pythagorase teoreemile vastav valem, siis selle kohta seletust anda ei saa – see lihtsalt on selline valem.

Valikvastustega küsimused on populaarsed ka programmeerimise algkursuste eksamitel [21, 28, 29]. Küsimuste sisu poolest saab sellega kontrollida paljusid erinevaid programmeerimisega seotud teadmiseid ja oskuseid. Järgnevalt on nimetatud erinevat tüüpi küsimusi, mida saab esitada programmeerimise algkursuste eksamitel kasutades valikvastustega küsimusi.

Küsimused teoreetiliste teadmiste kohta

Teoreetilisi teadmiseid kontrollivate küsimustega võib küsida kõike programmeerimise algkursusel õpetatu kohta. Küsimused võivad olla nii programmeerimise üldistest teemadest, kui ka küsimused konkreetse, kursusel õpetatava keele kohta. Kõik välja toodud ülesanded on programmeerimiskeeles Python. Ülesanded, mis originaalkujul ei olnud keeles Python, on tõlgitud Pythonisse. Kõikide ülesannete originaalid on välja toodud antud bakalaureusetöö lisades. Järgmised küsimused on bakalaureusetöö autori poolt loodud.

Küsimus 1. Näiteküsimus programmeerimiskeele Python süntaksi kohta

Milline avaldis on võrdne järgneva avaldisega?

```
summa = summa + 1
```

a) `summa = 1`
b) `summa = +1`
c) `summa += 1`
d) `summa++`

Küsimus 2. Näiteküsimus loogiliste tehete kohta.

Millistel tingimustel väärtustub järgnev avaldis tõeseks? Vali üks või mitu.
`a > 2 or b < 3`

a) Siis, kui a on 2 ja b on 3.
b) Siis, kui a on väiksem kui 2 ja b on suurem kui 3.
c) Siis, kui a on suurem kui 2 ja b on suurem kui 3.
d) Siis, kui a on 2 ja b on suurem kui 3.
e) Siis, kui a on 2 ja b on väiksem kui 3.

Koodist arusaamine

Koodist arusaamise (inglise keeles *code tracing*) ülesande puhul tuleb vastajal vastusevariantidest välja valida sobiv väljund, mis väljastatakse etteantud programmijupi läbides.

Näites kasutatud koodijupp koodist arusaamise ülesandes, pärineb Simoni jt artiklist „*Benchmarking Introductory Programming Exams: Some Preliminary Results*“ [30:105]. Antud ülesandes on originaaliga võrreldes muudetud järjendi `nums1` elemente.

Küsimus 3. Näiteküsimus koodist arusaamise kohta

Mis on muutuja `result` väärtus pärast koodi käivitamist?

```
nums1 = [1, -5, 1, 0, 4, 2,-3]
nums2 = [1, -5, 2, 4, 4, 2, 7]
result = 0
j = 0
while j < len(nums1):
    if nums1[j] != nums2[j]:
        result = result + 1
    j = j + 1
```

a) 0
b) 3
c) 4
d) 7

Koodi eesmärgi selgitamine

Koodi eesmärgi selgitamise ülesande puhul tuleb vastajal vastusevariantidest välja valida sobiv vastus, mis kirjeldab antud koodijupi üldist eesmärki.

Näide koodi eesmärki küsivast ülesandest. Näide on pärit Simoni ja Snowdoni artiklist „*Multiple-choice vs free-text code-explaining examination questions*“ [31:92].

Küsimus 4. Näiteküsimus koodi eesmärgi selgitamise kohta

Mis on järgneva koodijupi eesmärk?

```
result = 0
for j in range(0, len(number)):
    if number[j] < 0:
        result = result + 1
```

a) Leida järjendi väikseim element.
b) Lugeda kokku järjendis olevad negatiivsed arvud.
c) Leida järjendis olevate negatiivsete arvude summa.
d) Liita igale järjendis olevale negatiivsele arvule 1.
e) Leida esimese järjendis oleva negatiivse arvu indeks.

Antud tüüpi küsimust saab kasutada ka vabavastust nõudva küsimusena. Simon ja Snowdon [31] tõid välja, et jättes vastusevariandid andmata, võib vastajale jääda segaseks, mida tähendab „koodijupi eesmärk“. Kui küsiti sarnast küsimust vabavastust nõudva küsimusena, siis selle eesmärgi asemel vastasid paljud rea haaval seletuse, mida kood teeb. Täpsemast antud uuringus läbiviidud võrdlusest antakse ülevaade järgmises alapeatükis.

4.2.2 Vabavastuselised küsimused

Kirjalikku vabavastust nõudvad küsimused on küsimused, millele vastamiseks tuleb lahendus vastajal endal välja mõelda. Seal ei ole ette antud vastusevariante, nagu valikvastusküsimustes, mille seast õige valida. Ette võivad olla antud pidepunktid, millele vastus täpsemalt toetuma peaks. Selliselt on võimalik eksamineerijal eksaminandi vastust suunata, et vastus sisaldaks täpselt seda, mida eksamineerija teada tahab.

Biggs ja Tang [16] väidavad, et ideaalis saab vabavastust nõudvate küsimustega kontrollida eksaminandi laiemaid teadmiseid. Selline vastamise vorm võimaldab vastajal edasi anda oma enda mõtteid, demonstreerida oma teadmiseid ja teema valdamist laiemalt, kui pelgalt õigete vastuste ära tundmist. Samas toovad Biggs ja Tang [16] välja, et tegelikkuses ei pruugi see nii olla. Esiteks, on kirjalikul eksamil ajapiirang, seega väga sügavaid teadmiseid näitavate vastuste välja mõtlemisel võib piiratud ajast väheks jääda. See võib mõjuda halvasti vastuse sisulisele kvaliteedile. Teiseks, nagu oli välja toodud ka kolmandas peatükis, siis vastuste sisuline kvaliteet

algab juba eksamiks valmistumisel. Kui eksaminand õpib eksamiks ainult fakte, siis hoolimata küsimuste liigist, ei pruugi ta olla võimeline andma sisuküllast vastust.

Alapeatükis 4.1.1 toodi välja kolm küsimuse tüüpi valikvastustega küsimustele: küsimused teoreetiliste teadmiste kohta, koodi jälgimine ja koodi eesmärgi selgitamine. Neid kõiki saab küsida ka kirjalikku vastust nõudva küsimusena. Küsimus erineb selle poolest, et jäetakse ära vastusevariandid.

Vastusevariantide ärajätmine võib samas tekitada ohu, kus eksaminand ei pruugi aru saada, mida küsitakse, ehk mida täpsemalt vastuseks tahetakse. Selle tõid välja ka Simon ja Snowdon [31], kui nad võrdlesid eksaminandide vastuseid valikvastusküsimustes ja vabavastust nõudvates küsimustes. Koodi eesmärki kindlaks tegemist nõudvates ülesannetes tuli suur erinevus õigetes vastustes. Kui keskmiselt vastas valikvastusküsimustele õigesti 56% vastanutest, siis kirjaliku vastuse puhul oli see 17%. Lisaks õigesti vastanutele vastas kirjalikku vastust nõudvas küsimuses veel 17% vastanutest vastusena koodi rea haaval seletuse. Simon ja Snowdon järeldasid sellest, et kuna kuuendik vastas koodi eesmärgi asemel selle seletuse, siis ilma vastusevariantideta võib vastajatel keeruline olla aru saada, mida täpsemalt vastuseks tahetakse.

Selleks et suurendada arusaadavust nõutava vastuse sisu jaoks, võib koos küsimusega anda ka abistavaid vahendeid vastamiseks. Küsimusele võivad järgneda mõned pidepunktid, nagu eespool kirjeldatud. Need on eriti kasulikud sisukamat vastust nõudva küsimuse puhul, kus tuleb vastus anda mitme asja kohta.

Lühemat ja konkreetsemat vastust nõudvale küsimusele võib lisada näidisvastuse. Eespool toodud näiteülesande juurde, kus tuli välja tuua etteantud koodijupi üldine eesmärk, sobiks tuua näidisvastus, et vastus võiks olla kujul: „Antud koodijupi eesmärgiks on summeerida kokku järjendis olevad arvud“. Näidisvastuse andmise eesmärgiks on aidata vastajat vastuse struktuuri, mitte sisu poolest. Seega ei tohiks välja toodud näidisvastuseks olla õige vastus antud küsimusele, vaid pigem sarnane või täiesti vale vastus. Kõige olulisem on, et struktuur oleks õige ja näidisvastus aitaks vastajal jõuda soovitud vastuseni.

Lünkülesanded

Lisaks eelpool nimetatud küsimuste tüüpidele, saab vabavastuselise küsimusena kasutada ka lünkülesandeid, kus ette antud teksti sees olevasse lünka tuleb kirjutada sealt puudu olev sõna.

Programmeerimise algkursuste eksamites kasutatakse lünktekstiks koodijuppi, kuhu on teatud kohtadesse jäetud lüngad, mis tuleb vastajal ära täita.

Näide lünkülesande kohta pärineb Simoni jt artiklist „*How (not) to write an introductory programming exam*“ [32:142].

Küsimus 5. Näiteküsimus lünkülesande kohta

Täita lüngad nii, et muutuja `answer` väärtustuks täisarvude järjendi `num` väikseimaks elemendiks.

```
smallest = num[0]
for i in range (0, len(________)):
    if num[i] < ________:
        smallest = num[i]
answer = ________
```

4.2.3 Koodi kirjutamist nõudvad küsimused

Programmeerimise algkursuse eksam võib koosneda kahest osast: paberosast ja arvutiosast. Arvutiosa keskendub praktilisele programmeerimisele, kus eksaminandil tuleb kirjutada programm(id) antud probleemi lahendamiseks. Lihtsamaid programmeerimise ülesandeid lahendatakse ka kirjalikult paberil.

Näide koodikirjutamisest eksami paberosal pärineb Simoni jt artiklist „*Benchmarking Introductory Programming Exams: Some Preliminary Results*“ [30:106].

Küsimus 6. Näiteküsimus koodi kirjutamise kohta eksami paberosas

Koostada funktsioon, mis saades argumendiks täisarvude järjendi, arvutab ja tagastab (ujukomaarvuna) kõikide järjendis olevate arvude aritmeetilise keskmise.

Sellised ülesanded on lihtsasti lahendatavad paberil, sest need ei nõu keerulist süntaksit ning loodud programmi on kerge mõttes läbi jooksutada, et selle õigsust kontrollida. Mahukamaid programme, mis lahendavad palju keerulisemaid ülesandeid ja katavad palju rohkemaid programmeerimise konstruktsioone, viiakse läbi arvutis. Järgmises alapeatükis antaksegi ülevaade programmeerimise algkursuste eksamite arvutiosas kasutatavatest ülesannetest.

4.3 Küsimuste liigid arvutiosas

Programmeerimise algkursuse eksami arvutiosana mõeldakse eksami osa, mis lahendatakse arvutis. Kuigi ka paberosas käsitletud küsimusi annab teostada arvutis, siis antud bakalaureusetöös ei nimetata seda eksami arvutiosaks. Arvutiosana käsitletakse mahukamaid programmeerimisülesandeid, mida ei saa läbi viia kirjalikult paberil. Lisaks programmeerimisülesannetele antakse järgnevalt ülevaade ka Parsoni programmeerimispuslest [33] (inglise keeles *Parson’s Programming Puzzle*), mida saab kasutada nii kokkuvõtva hindamise vahendina eksamil, kui ka jooksvalt programmeerimise õpetamiseks.

4.3.1 Arvutis lahendatavad programmeerimisülesanded

Nagu on välja toodud alapeatükis 4.1, siis programmeerimise algkursuste eksamites moodustab poole eksami hindest praktilised programmeerimisülesanded. Sheardi jt [18] ilmunud artiklis selgus, et kuigi paljud programmeerimise algkursuste õppejõud pooldasid eksamil paberil koodi kirjutamist, nõustusid enamik õppejõududest, et praktiline koodi kirjutamine arvutis on õiglasem hindama üliõpilase programmeerimisoskust.

Arvutis programmeerides saab koostada mahukamaid ja keerulisemaid ülesandeid lahendavaid programme. Selliselt on võimalik saada ülevaade eksaminandi võimekusest ühte programmi kokku kombineerida väga suur osa erinevaid programmeerimise konstruktsioone. Võrreldes paberil koodi kirjutamisega, saab arvutis programmi pidevalt käivitada, et seda testida. Selliselt on võimalik leida vigu, mis võivad paberil kirjutades märkamata jääda. Lisaks pakuvad paljud IDEd (integreeritud programmeerimiskeskkonnad) abi, mis teeb mahukamate programmide koostamise lihtsamaks. IDE soovitab koodi kirjutades juba kasutatud muutujate ja funktsioonide nimesid, et vältida kirjavigadest tulenevaid probleeme ning kiirendab programmeerimist. Selliselt jõuab ajaga piiratud eksamil rohkem kirjutada.

Sisult võivad programmeerimise algkursuste eksamitel kasutatavad programmeerimisülesanded olla väga erinevad. Konkreetse ülesande sisu sõltub sellest, mida on kursuse jooksul õpetatud ja missuguseid oskuseid on soov eksamil kontrollida.

4.3.2 Parsoni programmeerimispusle

Parsoni programmeerimispusleks (inglise keeles *Parson's Programming Puzzle*) nimetatakse Parsonsi ja Hadeni 2006. aastal ilmunud artiklis välja käidud programmeerimise ülesannet [33]. Ülesanne kujutab endast programmeerimispuslet, kus ette antud koodiridadest tuleb kokku panna nõutud probleemi lahendav kood.

Ülesandekirjelduseks on antud probleem, mida nõutud programm peab lahendama. See võib olla nii sõnaline kirjeldus, kui ka vooskeem. Ülesande lahendamiseks on ette antud potentsiaalsed koodiread antud programmi koostamiseks. Vastaja üldiseks eesmärgiks on ette antud koodiread õigesti järjestada, et valmiks sobiv programm. Ette võib olla antud rohkem koodiridu, kui lõplikus programmis vaja läheb, seega vastaja peab lisaks õigele järjestusele välja valima ka sobivad read, millest programm koosneb. Sellisel juhul on ette antud ka arv, mitmest koodireast valmis programm peab koosnema.

Parsoni pusle on mõeldud lahendamiseks arvutis, kus saab ette antud koodiread lohistada õigesse järjekorda. Kui kasutatav programmeerimiskeel on taandetundlik, siis peab koodirea lohistama selliselt, et ka taane oleks õige. Parsoni puslet saab lahendada ka paberil. Selleks saab vastates välja toodud koodiread ära nummerdada ning kui oluline on koodi taane, siis võib valmis programmi eraldi välja kirjutada.

Näide Parsoni puslest programmeerimiskeeles Python. Näiteküsimuse vastuseks on programm, mis on välja toodud näiteküsimuses 4. Lisatud on rida järjendi `number` väärtustamise kohta.

Küsimus 7. Näiteküsimus Parsoni puslest programmeerimiskeeles Python

Koostada programm, mis väljastab ekraanile, mitu negatiivset arvu on järjendis `number`. Pange tähele, et programm peab ka väärtustama järjendi `number`.
Valmis programm koosneb kuuest reast.

```
if number[j] < 0:
print(number)
for j in range (0, number):
print(result)
number = [1, 5, 2, -3, 4, -1, -7, 5, 0]
result = result + 1
if result < 0:
result = 0
result = number[j]
for j in range (0, len(number)):
```

5. Osalejate tagasiside MOOC „Programmeerimise alused II“ eksamile

„Programmeerimise alused II“ on Tartu Ülikoolis õpetatav e-kursus. Tegemist on MOOCiga (inglise keeles *Massive Open Online Course)* ehk vaba juurdepääsuga e-kursusega, seega selle kursuse läbimiseks ei pea olema Tartu Ülikooli tudeng. Suurem osa antud kursuse osalejatest olidki inimesed väljastpoolt ülikooli.

Kursus on mõeldud kõigile, kes soovivad omandada programmeerimise algteadmiseid. Kuna tegemist on jätkukursusega „Programmeerimise alused“ kursusele, siis osalejatelt eeldatakse teadmiseid „Programmeerimise alused“ mahus.

„Programmeerimise alused II“ toimus perioodil 15. jaanuar kuni 11. märts 2018. Eksam toimus 25. märtsil 2018. Kokku oli kursuse maht 3 EAP ehk ca 78 tundi. Sellel osales 1050 osalejat. Kursuse eksam oli vabatahtlik. Kõik, kes olid Tartu Ülikooli tudengid ja sooritasid antud eksami, said kirja ka sellekohase tulemuse. Kursuse hindamine oli eristav ja hinne määrati ainult eksami tulemuste põhjal. Kokku osales eksamil 33 kursuse läbinut. Info kursuse kohta pärineb kursuse koduleheküljelt [34].

Kõigile eksamil osalenutele saadeti personaalselt, koos eksami eest saadud punktidega, tagasisideküsitlus eksami kohta. Sellele vastas 14 eksaminandi. Vastajatelt sooviti saada tagasiside:

- eksami korralduslikule poolele,
- eksami keerukusele,
- eksami meeldivusele,
- eksami sisule,
- eksamiks valmistumisele.

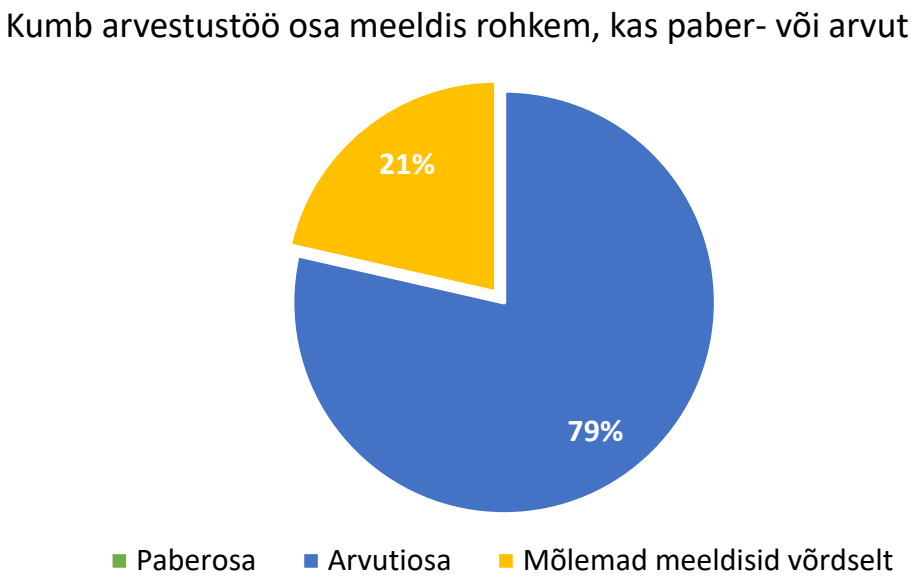
Viimase küsimusena paluti soovi korral kirjutada üldiseid kommentaare eksamist. Tagasisideküsitlus on välja toodud bakalaureusetöö lisades. Tärniga tähistatud küsimused olid vastamisel kohustuslikud.

Eksami korraldusliku poole pealt küsiti eksaminandidelt tagasiside eksami kestvuse ja eksami läbiviimise kohta. Eksam koosnes kahest osast. Eksami paberosa viidi läbi täielikult paberil. Arvutiosas anti ülesanded ette paberil, ning programmeerimine viidi läbi arvutis, kus tuli ka valminud programmid esitada. Eksami mõlemad osad toimusid järjest. Eksam algas paberosaga, ning kui eksaminand oli selle valmis saanud ja ära esitanud, sai ta alustada arvutiosaga. Eksami kestvuse, 180 minuti jooksul, tuli ära lahenda eksami mõlemad osad. Küsitlusest selgus, et 71% vastanutest oli rahul sellise toimumisega, ülejäänud 29% vastanutest oleks eelistanud teha eksamit küll samal päeval, aga vaheajaga kahe eksamiosa vahel.

Arvamused eksami kestvusest jaotusid erinevalt. Valitud oli kõiki vastusevariante. 36% vastanutel jäi aega kas natuke või palju üle. Sama paljudel vastanutel jäi ajast kas natuke või palju puudu. 28% vastanutest oli rahul eksami kestvusega. Selline võrdne jaotus näitab seda, et eksaminandid olid väga erineva tasemega. Oli neid, kellel jäi aega üle, samas oli ka neid, kellel tuli sellest puudu.

Tagasiside küsiti ka eksami keerukuse kohta, seda eraldi nii paberosa, kui ka arvutiosa kohta. Saadud tulemustest ilmneb, et paberosa peeti keerulisemaks, kui arvutiosa. Paberosa keerukuse kohta vastas 57% vastanutest, et see oli pigem keeruline ning 36% vastanutest, et see oli täiesti paras. Arvutiosa keerukuse kohta vastas 64% vastanutest, et see oli täiesti paras, ning 29% vastanutest, et see oli pigem lihtne. Mõlema osa kohta vastati ainult üks kord ehk 7% vastanutest, et konkreetne osa oli liiga keeruline. Individuaalseid vastuseid vaadates selgus, et tegemist ei olnud sama inimesega. Vaadates keerukust uurinud küsimuste vastuseid, siis paberosas domineerisid variandid „pigem keeruline“ ja „täiesti paras“, arvutiosas domineerisid variandid „pigem lihtne“ ja „täiesti paras“. See näitab, et eksami arvutiosa hinnati lihtsamaks. Välja tulnud keerukus kajastub ka eksamiosade meeldivusel.

Uurides vastanutelt nende meeldimist paber- ja arvutiosa kohta selgus, et 79% vastanutest meeldis arvutiosa rohkem ja 21% vastanutest meeldisid mõlemad osad võrdselt. Tulemused eksamiosade meeldivuse kohta on välja toodud Joonisel 2.



Joonis 2. Vastanute meeldivus eksamiosadele

Eksami meeldimist teada soovivale küsimusele järgnes võimalus oma vastust kommenteerida, antud küsimusele vastas kokku 6 inimest. Kommentaarid erinesid palju. Kuigi pooled nendest tõid välja negatiivset hoiakut eksami paberosa vastu, siis konkreetsed põhjused erinesid. Kaks vastanut tõid välja, et paberosa eeldab erinevate peensuste pähe õppimist, mis reaalses situatsioonis pole eriti mõistlik. Lisaks mainis üks vastanutest, et arvutiosa oli see, mida kursusel rohkem õpetati, seega jäi tal puudu paberosas nõutavatest teadmistest.

Tagasisides küsiti vabavastustena eksamiks valmistumise kohta. Kategoriseerides saadud vastused, selgus kuidas vastanud valmistusid eksamiks ja missuguseid teemasid nad rohkem kordasid. Eksamiks kordamise viisidest osutusid kõige populaarsemaks näidisülesannete lahendamine ja materjalide ülelugemine. Neid tõid välja vastavalt 43% ja 36% kõikidest

vastanutest. Teemad, mida kõige enam eksamiks korrati olid erinevad andmestruktuurid ja rekursioon. Erinevaid andmestruktuure kordasid 43% vastanutest ja rekursiooni kordas 36% vastanutest. Andmestruktuurid, mille kordamist küsitluse vastustes mainiti olid järjendid, hulgad, ennikud ja sõnastikud. Enim toodi välja hulgad ja sõnastikud.

Eksamiks valmistumiseks anti kursuse läbijatele näidisülesanded. 93% vastanutest märkis, et näidisülesanded olid kas pigem kasulikud või väga kasulikud. Ainult 1 vastaja, kes moodustas kõikidest vastanutest 7% vastas, et näidisülesanded ei olnud eriti kasulikud. Vastajatelt uuriti ka kuidas vastas eksam juhendi ja näidisülesannetega kujundatud ootustele. 72% vastanutest tõi välja, et eksam vastas kas natuke, või täielikult ootustele. Ühe vastaja arvates ei vastanud eksam eriti ootustele ja üks vastaja tõi välja, et eksami paberosa tundus raskem, kui antud aine eelkursuse, „MOOCi Programmeerimise algused“ eksam. Kaks vastajat ei osanud antud küsimusele vastata.

Vabas vormis kommentaare eksami ja kursuse kohta jätsid 6 vastanut. Kaks vastajat tõid otseselt välja, et nad ei näe eksami paberosa mõtet. Põhjustena mainiti, et väikeste nüansside ja trikiga küsimuste küsimine pole eluliselt eriti sobiv, sest sellised asjad on kergesti guugeldatavad ja seega nende pähe õppimine kasutu. Üks vastanu küll ei jaganud kommentaare eksami paberosa kohta, kuid märkis ära, nagu mõnigi teine, et praktikat jäi väheks.

Teises peatükis selgus, et Bennedseni ja Casperseni väitel pole programmeerimise eksamit ainult paberil korraldada eriti sobilik – ausama hinnangu eksaminandi oskustest annab eksam, milles on sees ka arvutis programmeerimist [19:188]. Arvutis programmeerimise meeldivus ilmnes ka käesoleva küsimustiku vastanute seast. Suuremale osale vastanutest meeldis arvutiosa rohkem kui paberosa. Vastajad leidsid, et paberosa eeldab programmeerimiskeele erinevate peensuste peast teadmist. Tegelikus elus ei pruugi programmeerija kõiki vajaminevaid üksikasju teada, sest neid saab guugeldada. Seega arvati, et ka eksamil pole mõistlik selliseid teadmiseid kontrollida.

Teises peatükis toodi välja, et paljud programmeerimise algkursuste õppejõud andsid õppijatele eksamiks kordamiseks proovieksami või mõne varasema aasta eksami. Seda tehti eesmärgil, et anda ülevaade tuleva eksami stiilist, ning et oleks näiteküsimusi, mille põhjal korrata. Antud eesmärgi täitmiseks anti „Programmeerimise alused II“ kursuse osalejatele eksamiks kordamiseks näidisküsimusi. Küsitlusest selgus, et need täitsid nimetatud eesmärki, andes sobiliku ettekujutuse nii eksami sisust kui ka vormist. Suurem osa vastanutest märkis, et näidisülesannetest oli eksamiks õppides kasu ning nende arvates vastas eksam tekkinud ootustele.

Kokkuvõte

Antud bakalaureusetöö peamiseks eesmärgiks oli anda ülevaade erinevate programmeerimise algkursuste eksamitest. Selle eesmärgi täitmiseks vastati uurimisküsimustele „Millised on programmeerimise algkursused?“ ja „Millised on programmeerimise algkursuste eksamid?“. Nendele küsimustele vastamiseks analüüsiti teaduskirjandust ning selle põhjal toodi välja teemad ja programmeerimiskeeled, mida käsitletakse programmeerimise algkursustel. Teaduskirjandusest selgus, et programmeerimise algkursustel domineerivad programmeerimiskeeled Java ja Python ning Pythoni populaarsus võrreldes Javaga on kasvamas. Lisaks selgus, et programmeerimise algkursuste peamiseks eesmärgiks on õpetada algoritmilist mõtlemist ning programmeerimise üldisemaid konstruktsioone, samas on olulisel kohal ka programmeerimise oskus.

Programmeerimiskursuste eksamite kohta toodi välja asjaolud, kuidas need valmivad ja kuidas õppejõud aitavad õppijatel eksamiks valmistuda. Lisaks anti ülevaade erinevatest teemadest ja eksamiosadest programmeerimise algkursuste eksamites. Teaduskirjandusest selgus, et programmeerimise algkursuse eksam oleks mõistlik korralda kahes osas. Paberil läbiviidavas paberosas ja arvutis läbiviidavas arvutiosas. Paberosas kontrollitakse eksaminandi teadmiseid nii programmeerimise kohta, kui ka oskust kirjutada lihtsamaid koodijuppe ning oskust mõista etteantud koodi. Arvutiosas kontrollitakse eksaminandi programmeerimisega seotud oskuseid läbi mahukamate programmide koostamise.

Bakalaureusetöö teiseks eesmärgiks seati teaduskirjandusest koondada erinevad küsimuste liigid ja eksami läbiviimise vormid. Nimetatud eesmärgi täitmiseks vastati uurimisküsimustele „Millised on erinevate küsimuste liikide eelised ja puudused?“ ning „Kuidas toetavad erinevad eksami läbiviimise vormid eksami sooritamist?“. Antud küsimustele vastamiseks toodi teaduskirjandusest välja erinevaid küsimuste liike, mida kasutatakse programmeerimise algkursustel. Neid liike võrreldi ja iga liigi kohta esitati ka seda liiki illustreeriv näiteküsimus. Lisaks anti kirjanduse põhjal ülevaade avatud ja suletud materjalidega eksamist ning abilehe kasutamisest eksamil. Neid võrreldi ning toodi välja kõikide eeliseid ja puuduseid.

Bakalaureusetöö viimaseks eesmärgiks oli teada saada Tartu Ülikooli e-kursuse „Programmeerimise alused II“ osalejate arvamus antud aines läbiviidava eksami kohta. Antud eesmärgi täitmiseks vastati uurimisküsimusele „Mida arvavad osalejad „Programmeerimise alused II“ kursusel toimunud eksamist?“. Sellele küsimusele vastamiseks koostati antud eksami sooritanutele küsitlus, mille põhjal selgitati välja eksaminandide arvamus antud eksamist. Sealt selgus, et eksaminandid hindasid lihtsamaks arvutis programmeerimist nõudvaid küsimusi, kui paberil vastatavaid küsimusi. Eksamiks kordamise vormidest osutusid kõige populaarsemateks näidisülesannete lahendamine ja materjalide ülelugemine. Peaaegu kõik vastajad leidsid, et kordamiseks antud näidisülesanded olid kasulikud ning üldiselt vastas eksam ootustele.

Bakalaureusetöö autor loodab, et antud tööst on kasu edaspidiste eksamite loomisel ja olemasolevate eksamite analüüsimisel ning võimalusel muutmisel. Lisaks võib olla kasu töös viidatud kirjandusest, et edaspidiste analüüside koostamisel oleks lihtsam leida juba avaldatud

artikleid antud teemast. Tulevikus saaks bakalaureusetöö käigus valminud küsitlust kasutada uuesti kursuse „Programmeerimise alused II“ eksami uurimiseks. Esiteks, oleks huvitav jälgida eksami muutudes eksaminandide tagasisidet ning võrrelda seda varasemate eksamitega. Teiseks, et saada paremad ja kindlamad tulemused antud tagasisidest, võiks tulevikus saada vastuseid rohkematelt eksaminandidelt.

Viidatud kirjandus

[1] Daniswara JA. Up & To The Right—The Growth of Computer Science at Yale. *Medium*, 2018. https://medium.com/@johnamadeo/up-to-the-right-the-rise-of-computer-science-at-yale-fd9983b0596c (16.02.2018)

[2] Bernhard MP. CS50 Logs Record-Breaking Enrollment Numbers. *The Harvard Crimson*, 2014. http://www.thecrimson.com/article/2014/9/11/cs50-breaks-enrollment-records/ (16.02.2018)

[3] Tartu Ülikooli statistika. https://www.ut.ee/et/sisseastumine/statistika-0 (16.02.2018)

[4] Esimese astme vastuvõtt 2005. aastal. https://www.ut.ee/sites/default/files/ut_files/stat_bak%202005.htm (16.02.2018)

[5] Tartu Ülikooli vastuvõtustatistika 2017. https://www.ut.ee/sites/default/files/www_ut/sisseastumine/tu_vastuvotustatistika_ba_2017.xlsx (16.02.2018)

[6] Sarap I. Vaba juurdepääsuga e-kursused kõrgkoolis. Kursuse "Programmeerimise alused" näide. Tartu Ülikooli arvutiteaduse instituut, Bakalaureusetöö. 2017. https://comserv.cs.ut.ee/home/files/sarap_informaatika_2017.pdf?study=ATILoputoo&reference=4AFA7537D03967D1AD3507E78E4890F501B9207F.

[7] Tew AE, Guzdial M. Developing a validated assessment of fundamental CS1 concepts. *SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, p. 97-101.

[8] Austing RH, Barnes BH, Bonnette DT, Engel GL, Stokes G. Curriculum '78: recommendations for the undergraduate program in computer science— a report of the ACM curriculum committee on computer science. *Communications of the ACM*, 1979, Vol 22, Ed. 3, p. 147-166.

[9] Hertz M. What Do "CS1" and "CS2" Mean? Investigating Differences In the Early Courses. *SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, p. 199-203.

[10] Peedosk K. Eesti kõrgkoolide programmeerimise algkursused. Tartu Ülikooli arvutiteaduse instituut, Bakalaureusetöö. 2014. https://sisu.ut.ee/sites/default/files/ikt/files/bakalaureusetoo_karl_peedosk.pdf.

[11] Murphy E, Crick T, Davenport JH. An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming*, 2017, Vol 1, Ed. 2, p. 1-23.

[12] Mason R, Simon. Introductory Programming Courses in Australasia in 2016. *ACE '17 Proceedings of the Nineteenth Australasian Computing Education Conference*, 2017, p. 81-89.

[13] Guo P. Python Is Now the Most Popular Introductory Teaching Language at Top U.s. Universities. *Communications of the ACM,* 2014. https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext (20.03.2018)

[14] Mason R, Cooper G. Introductory programming courses in Australia and New Zealand in 2013 - trends and reasons. *ACE '14 Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148,* 2014, p. 139-147.

[15] Leping V, Lepp M, Niitsoo M, Tõnisson E, Vene V, Villems A. Python prevails. *CompSysTech '09 Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing,* 2009.

[16] Biggs J, Tang C. Õppimist väärtustav õpetamine ülikoolis: keskmes õppija tegevused. Tartu: Tartu Ülikooli kirjastus. 2008.

[17] Pilli E. Väljundipõhine hindamine kõrgkoolis. 2009.

[18] Sheard J, Simon, Carbone A, D'Souza D, Hamilton M. Assessment of programming: pedagogical foundations of exams. *ITiCSE '13 Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013, p. 141-146.

[19] Bennedsen J, Caspersen ME. Assessing Process and Product: A Practical Lab Exam for an Introductory Programming Course. *Innovation in Teaching and Learning in Information and Computer Sciences*, 2007, Vol 6, Ed. 4, p. 182-202.

[20] Haghighi PD, Sheard J. Summative Computer Programming Assessment Using Both Paper and Computer. *Proceedings of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation*, 2005, p. 67-75.

[21] Simon, Sheard J, Carbone A, Chinn D, Laakso MJ, Clear T, et al. Introductory programming: examining the exams. *Proceedings of the Fourteenth Australasian Computing Education Conference (ACE2012),* 2012, Vol 123, p. 61-70.

[22] Tussing L. A Consideration of the Open Book Examination. *Educational and Psychological Measurement*, 1951, Vol 11, Ed. 4-1, p. 597-602.

[23] Soloway EM. Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM,* 1986, Vol 29, Ed. 9, p. 850-858.

[24] Feldhusen JF. An Evaluation of College Students' Reactions to Open Book Examinations. *Educational and Psychological Measurement,* 1961, Vol 21, Ed. 3, p. 637-646.

[25] Boniface D. Candidates' use of notes and textbooks during an open-book examination. *Educational Research*, 1985, Vol 27, Ed. 3, p. 201-209.

[26] Raadt Md. Student created cheat-sheets in examinations: impact on student outcomes. *ACE '12 Proceedings of the Fourteenth Australasian Computing Education Conference*, 2012, Vol 123, p. 71-76.

[27] Erbe B. Reducing Test Anxiety While Increasing Learning: The Cheat Sheet. *College Teaching*, 2007, Vol 55, Ed. 3, p. 96-98.

[28] Petersen A, Craig M, Zingaro D. Reviewing CS1 exam question content. *SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011, p. 631-636.

[29] Shuhidan S, Hamilton M, D'Souza D. Instructor perspectives of multiple-choice questions in summative assessment for novice programmers. *Computer Science Education*, 2010, Vol 20, Ed. 2, p. 229-259.

[30] Simon, Sheard J, D'Souza D, Klemperer P, Porter L, Sorva J, et al. Benchmarking Introductory Programming Exams: Some Preliminary Results. *ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research*, 2016, p. 103-111.

[31] Simon, Snowdon S. Multiple-choice vs free-text code-explaining examination questions. *Koli Calling '14 Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, 2014, p. 91-97.

[32] Simon, Sheard J, D'Souza D, Lopez M, Luxton-Reilly A, Putro IH, et al. How (not) to write an introductory programming exam. *Proceedings of the 17th Australasian Computing Education Conference (ACE 2015)*, 2015, Vol 160, p. 137-146.

[33] Parsons D, Haden P. Parson's programming puzzles: a fun and effective learning tool for first programming courses. *ACE '06 Proceedings of the 8th Australasian Conference on Computing Education – Volume 52*, 2006, p. 157-163.

[34] Programmeerimise alused II. https://courses.cs.ut.ee/2018/eprogalused2/spring/Main/HomePage (2.05.2018)

[35] Tew AE. Assessing fundamental introductory computing concept knowledge in a language independent manner. Georgia Institute of Technology, Doctoral Dissertation. 2010.

[36] Sheard J, Simon, Carbone A, Chinn D, Laakso MJ, Clear T, et al. Exploring programming assessment instruments: a classification scheme for examination questions. *ICER '11 Proceedings of the seventh international workshop on Computing education research*, 2011, p. 33-38.

[37] Higgins CA, Gray G, Symeonidis P, Tsintsifas A. Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing (JERIC)*, 2005, Vol 5, Ed. 3, p. 1-21.

Lisad

I. "Programmeerimise alused II" eksami tagasiside küsitlus

1. Kuidas sobis arvestustöö kestus (180 minutit). Kas oleks võinud aega olla rohkem või jäi seda üle? *
    a. Aega tuli palju puudu
    b. Aega tuli natuke puudu
    c. Aega oli parajalt
    d. Aega jäi natuke üle
    e. Aega jäi palju üle
    f. Muu/oma vastus
2. Arvestustöö paberosa ja arvutiosa pidi lahendama vahetult üksteise järel. Millist varianti te eelistaksite? *
    a. Nagu pragu oli
    b. Samal päeval, vaheajaga kahe osa vahel
    c. Erinevatel päevadel
    d. Muu/oma vastus
3. Kogu kursuse hinne kujuneb ainult arvestustöö põhjal. Kas see on sobiv? *
    a. On täiesti sobiv
    b. On pigem sobiv
    c. Ei oska vastata
    d. Pigem ei sobi
    e. Täiesti ebasobiv
    f. Muu/oma vastus
4. Kui keeruline oli Teie jaoks arvestustöö paberosa? *
    a. Liiga lihtne
    b. Pigem lihtne
    c. Täiesti paras
    d. Pigem keeruline
    e. Liiga keeruline
5. Kui keeruline oli Teie jaoks arvestustöö arvutiosa? *
    a. Liiga lihtne
    b. Pigem lihtne
    c. Täiesti paras
    d. Pigem keeruline
    e. Liiga keeruline

6. Kumb arvestustöö osa meeldis rohkem, kas paber- või arvutiosa? *
    a. Paberosa
    b. Arvutiosa
    c. Mõlemad meeldisid võrdselt

    Soovi korral, palun kommenteerige, miks nii vastasite.

7. Arvestustöös oli ülesandeid (nt paberosas sõnastike ülesanne võistkondadega ja arvutiosas kilokaloritega) elulise sisuga. Kuidas suhtute ülesannete tekstide elulisusse? *
    a. Eluline sisu segab väga
    b. Eluline sisu pigem segab
    c. Eluline sisu ei sega ega toeta
    d. Eluline sisu pigem toetab
    e. Eluline sisu toetab väga
8. Kas mingi teema oleks võinud arvestustööl veel sees olla? Kui jah, siis palun nimetage neid teemasid. *
9. Mismoodi valmistusite arvestustööks? *
10. Mis teemasid õppisite ja kordasite rohkem? *
11. Kuivõrd kasulikud olid näidisülesanded arvestustööks õppides? *
    a. Väga kasulikud
    b. Pigem kasulikud
    c. Ei olnud eriti kasulikud
    d. EI olnud üldse kasulikud
12. Kuivõrd vastas arvestustöö, juhendi ja näidisülesannetega kujundatud ootustele? *
    a. Vastas täielikult ootustele
    b. Vastas natuke ootustele
    c. Ei oska vastata
    d. Ei vastanud eriti ootustele
    e. Ei vastanud üldse ootustele
    f. Muu/oma vastus
13. Siia võite kirjutada kommentaare arvestustöö kohta.

II. Koodinäidete originaalülesanded

1. „Küsimus 3. Näiteküsimus koodist arusaamise kohta“. Lehekülg 24.

What will be the value of result after the following code statements are executed?

```
int[] nums1 = {1, -5, 2, 0, 4, 2, -3};
int[] nums2 = {1, -5, 2, 4, 4, 2, 7};
int result = 0;
int j = 0;
while (j < nums1.length)
{
    if (nums1[j] != nums2[j])
    {
        result = result + 1;
    }
    j = j + 1;
}
```

2. „Küsimus 4. Näiteküsimus koodi eesmärgi selgitamise kohta“. Lehekülg 25.

What is the purpose or outcome of the following piece of code?

```
result = 0
for j in range(0, len(number)):
     if number[j] < 0:
         result = result + 1
```

a) to find the smallest number in the array
b) to count the negative numbers in the array
c) to sum the negative numbers in the array
d) to add 1 to each of the negative numbers in the array
e) to find the index of the first negative number in the array

3. „Küsimus 5. Näiteküsimus lünkülesande kohta“. Lehekülg 27.

The following piece of code sets *answer* to the smallest element of the integer array *num*.

```
int smallest = num[0];
for (int i=1; i < num.Length; i++)
{
    if (num[i] < smallest)
    {
        smallest = num[i];
    }
}
answer = smallest;
```

Complete the code in the boxes below so that it also sets *answer* to the smallest element of *num*. Note that the sixth line is different in the two listings.

```
int where = _________;
for (int i=0; i < num.Length; i++)
{
    if num[i] < _________)
    {
        where = i; // Note difference
    }
}
answer = __________;
```

4. „Küsimus 6. Näiteküsimus koodi kirjutamise kohta eksami paberosas“. Lehekülg 27.

Write a method that will be given an array of integers and will calculate and return (as a double) the mean (average) of all the integers in the array.

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kristen Kotkas**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
Programmeerimise algkursuste eksamid,
(*lõputöö pealkiri*)

mille juhendajad on Marina Lepp ja Eno Tõnisson,
(*juhendajate nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14. mai. 2018**