TARTU ÜLIKOOL

Loodus- ja täppisteaduste valdkond

Arvutiteaduse instituut

Matemaatika- ja informaatikaõpetaja õppekava

Jelizaveta Reitsakas

Programmeerimise sissejuhatava kursuse õppijate teadmisi ja oskusi mõõtev test

Magistritöö (15 EAP)

Juhendajad: Eno Tõnisson, PhD,

Marina Lepp, PhD

Tartu 2020

Programmeerimise sissejuhatava kursuse õppijate teadmisi ja oskusi mõõtev test

Magistritöö

Jelizaveta Reitsakas

Lühikokkuvõte: Programmeerimise õpetamisel tuleb pöörata tähelepanu õppijate teadmiste ja oskuste taseme hindamisele nii õppe alguses kui ka selle lõppemisel. Seda saab teha testi vormis. Magistritöö eesmärgiks on luua valideeritud test, mille abil saab hinnata õppijate teadmisi ja oskusi programmeerimiskursuse "Tehnoloogia tarbijast loojaks" alguses ja/või lõpus. Töö esimeses peatükis on toodud ülevaade Venemaal ja USAs kasutusel olevatest valideeritud testidest. Töö teises peatükis on kirjeldatud testi loomise protseduuri ning piloteerimise protsessi. Töö tulemusena loodud test piloteeriti kursuse "Tehnoloogia tarbijast loojaks" arvestustööks ettevalmistava testina. Võib öelda, et testi on võimalik kasutada nimetatud kursuse õppijate teadmiste ja oskuste hindamisel.

Võtmesõnad: Programmeerimisülesanded, õppematerjalid, programmeerimise algõpe

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

Test for measuring the knowledge and skills of students in programming introductory course

Master's Thesis

Jelizaveta Reitsakas

Abstract: When teaching programming, attention must be paid to assessing the level of students' knowledge and skills both at the beginning and at the end of the study. This can be done in the form of a test. The aim of the master's thesis is to create a validated test, which can be used to assess students' knowledge and skills at the beginning and/or end of the programming course "Introduction to Programming". The first chapter of the work provides an overview of validated tests used in Russia and the USA. The second chapter describes the test creation procedure and the piloting process. The test created as a result of the work was piloted as a preparatory test for the accounting work of the course "Introduction to Programming". It can be said that the test can be used to assess the knowledge and skills of students in this course.

Keywords: Computer science education, CS1, introductory programming, programming tasks, teaching materials, teaching programming to novices

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

Sisukord

Sissejuhatus

Õppimisel ja õpetamisel on hindamisel suur roll: õppija teadmiste, oskuste mõõtmine on olulised nii õppijale kui ka õpetajale. Programmeerimise aluste õppimise valdkonnas on vähe uuringuid, kuidas ja mida hindama peab, millised on õppijate vigade, oodatust halvema õppimise põhjused, on vähe ka valideeritud hindamismudeleid, meetodeid, vahendeid (Tew, 2010; Parker, Guzdial & Engleman, 2016; Xie, Davidson, Li & Ko, 2019; Kotkas, 2018). Mida ja kuidas hinnata programmeerimise sissejuhatava kursuse puhul? Tavaline ning tõhus variant on lasta lahendada programmeerimisülesandeid. Lisaks sellele on tarvis hinnata ka üldiseid teadmisi ning oskusi. Teadmiste hindamiseks on võetud kasutusele ülesannete ja/või küsimuste kogumid − testid, mille lahendamise hindamine annab võimaluse hinnata õpilase taset (EKI ühendsõnastik, 2020). Edasi kasutan töös sõna "test" just ülesannete/küsimuste kogumi tähenduses.

Üldiselt viiakse teste läbi kolmel eesmärgil: eelhindamine, õppimistulemuste kontroll õppimise käigus ning kursuse lõpus (Tchelyshkova, 2002). Nendest esimest on kõige vähem uuritud ning vähe kasutatud. Samal ajal on see väga tõhus viis õppijate/õppe diferentseerimiseks algava kursuse jaoks. Eelhindamine on üks kujundava hindamise olulistest meetoditest (Karus, 2015). Valideeritud testide kasutamine lubab hinnata õppijate oskusi ning teadmisi objektiivselt, õppija saab tagasisidet kiiresti, korraga saab hinnata väga paljusid inimesi. Hea test sisaldab küsimusi kõikide kursusel õpetavate teemade kohta, samal ajal ei tohi olla dubleerivaid küsimusi. Kontrollitakse esmatähtsaid oskusi ja teadmisi, testis ei või olla küsimusi teisejärguliste teadmiste/oskuste kohta (Tchelyshkova, 2002). Testi valideerimisel saab hinnata selle raskust testitavate jaoks, vaadatakse üle küsimuste üldkarakteristikud (nt sõnastus, küsimuste järjekorranumbrid, juhiste olemasolu jne). Valideerimise tulemusena saab informatsiooni, millised küsimused on efektiivsed õppijate diferentseerimiseks ning saab teada, kui hästi hindab antud test õppija teadmisi/oskusi. Järgmisena muudetakse testi, parandades esile tulnud vigu, kitsaskohti (Xie jt, 2019). Valideerimata testi kasutamisel jääb alati küsimus, kas testi lahendamise tulemused näitavad, kui hästi on õppija teadmised ja oskused omandanud või näitavad seda, et test kontrollib nt väheolulisi asju või testi küsimused on halvasti sõnastatud. Õppijatel õppimise algul olevaid eelteadmisi ja -oskusi (lähtetaset (HAR)) hindav test (edasi "eeltest") on leidnud kasutust erinevates valdkondades, arvutiteaduses vähem. Mõnikord kasutatakse sama testi nii eeltestina kui kursuse lõpus õpitulemuste hindamiseks, õppeprotsessi analüüsimiseks (Tew, 2010). Nii hoitakse kokku vaeva kahe testi loomisel ning tekib võimalus hinnata muutust.

Valideeritud testide loomine on aeganõudev, seega võetakse seda harva ette. Eestis puudub valideeritud test programmeerimisaluste kontrollimiseks, mida saab põhjendada sellega, et

Eestis on koolides seni õpetatud programmeerimist väga erinevalt: sisuliselt on igas koolis, igal õpetajal oma kursus. Puniste (2015) toob näiteks välja, et 16 kursuse käigus õpetatakse 14 erinevat programmeerimiskeelt. Sellise killustatuse puhul ei saa ka testi standardiseerida. Olukorra ühtlustamiseks on Hariduse Infotehnoloogia SA (HITSA) toel valminud gümnaasiumi valikkursus "Programmeerimine". 2022. aastaks õpetab Tartu Ülikool (2019. aastal Majandus- ja kommunikatsiooniministeeriumiga sõlmitud lepingu alusel) vähemalt 500 noorele programmeerimist kursuse "Tehnoloogia tarbijast loojaks" raames (edasi "TTL") (MKM, 2019). TTL on kooskõlas gümnaasiumi valikkursusega "Programmeerimine" ning toimub korduvalt. Kursuse raames tutvustatakse algoritmilist mõtteviisi ja programmeerimist neile, kes pole varem programmeerimisega kokku puutunud või kel on see kogemus olnud väike (TTL, 2020). Osaleda saavad 16−26-aastased noored olemata sellest, kas nad mingis koolis õpivad või mitte.

Ühise kursuse läbiviimisel on suured eelised: ühised materjalid, sarnane õppijate juhendamine, võrreldatavad tulemused. Käesoleva magistritöö põhieesmärk on luua kursuse TTL jaoks õppijate taset mõõtev valideeritud test. Loodav test peab olema kasutatav nii eeltestina kui ka lõpptulemusi mõõtva testina. Eesmärgi saavutamiseks uurin, milliseid teste on üldse olemas ja kas on teste, mida saab (kasvõi osaliselt) kasutusele võtta ning nende testide loomise metoodikaid. Ülevaade olemasolevatest testidest ning nende loomise praktikatest on magistritöö esimeses peatükis. Leitud allikatel põhineb oma testi loomise protseduur, mis on oluline valideeritud testi loomisel. Magistritöö teine peatükk on pühendatud uue testi loomisele. Testi valideerimise üks osa on testi piloteerimine, mille nimel sai läbi viidud intervjuud, mille tulemused on kirjeldatud magistritöö kolmandas peatükis. Samuti on magistritöö kolmandas peatükis kirjeldatud intervjuu põhjal parandatud testi TTL-kursuse osalejatega piloteerimise tulemused.

Magistritööl on 6 lisa:

1. Allison Elliott Tew' loodud testi näidisküsimused.
2. Allison Elliott Tew' pseudokoodi juhendi esimene lehekülg.
3. Tew' testi originaalküsimus ning Parkeri jt poolt kopeeritud küsimus.
4. TTL-kursuse teemade kategoriseerimine.
5. Testi ning arvestustesti tulemuste võrdlemine konstruktide kaupa.
6. Link test magistritöö tulemusena loodud testile. Test on kättesaadav ainult töö retsensendile ning kaitsmiskomisjoni liikmetele testi valiidsuse säilitamise nimel.

Magistritööd kirjutades toetusin mõistete ühtlustamisel "Haridussõnastikule" (HAR). Arvutiteaduse spetsiifiliste mõistete tõlkimisel olid suureks abiks "Standardipõhine tarkvaratehnika sõnastik" (STATS) ning "Andmekaitse ja infoturbe leksikon" (AKIT).

1. Kasutusel olevad eeltestid

Käesolevas peatükis on ülevaade teaduskirjandusest, milles käsitletakse programmeerimise sissejuhatavate kursuste õppijate oskuste/teadmiste hindamist, eesmärgiga leida häid (eel)testide näiteid. Ülevaade põhineb eesti-, inglise- ja venekeelsetel teadusartiklitel ja dokumentidel, mis kajastavad Eesti, USA, Saksamaa ja Venemaa kogemusi.

1.1. Terminoloogiast

Maailmas kasutusel olevate testide otsimisel oli suureks takistuseks ühise terminoloogia puudus: samade sõnade tähendusväljad erinevatel maadel või erinevates kontekstides võivad olla erinevad. Alustades kõige kõrgemalt tasemelt, milleks on distsipliini ja kraadi nimetus, millele osutuvad oma artiklis "What's in a Name?: International Interpretations of Computing Education Terminology" Simon jt (2015). Autorid kirjutavad, et kogu maailmas kasutatakse samu sõnu erinevate õppekavade jaoks, samal ajal sisuliselt samalaadse õppekava jaoks võivad kasutusel olla erinevad nimetused. Näiteks Eestis on kolmes ülikoolis kokku kuus erinevat õppekava: arvutisüsteemid (Tallinna Tehnikaülikool, ingl *computer and systems engineering*), arvutitehnika (Tartu Ülikool, ingl *computer engineering*), informaatika (Tallinna Ülikool ja Tartu Ülikool, ingl *computer science*, Tallinna Tehnikaülikool, ingl *informatics*), äriinfotehnoloogia (Tallinna Tehnikaülikool, ingl *business information technology*). USAs on näiteks järgmised õppekavad: *Computer science*, *Information systems*, *Information technology*, *Computer information systems*. Mõistete tähenduste variatiivsus on oluliselt raskendanud antud magistritöö jaoks materjalide otsimist. Näiteks eeltestide kohta info otsimisel kasutasin järgmisi mõisteid:

- eesti keeles: eeltest, pretest, lähtetest, eelhindamine jt,
- inglise keeles: "*pretest*", "*introducing test*", "*assessment*", "*assessment tool*", "*concept inventories*" jt,
- vene keeles: "*входное тестирование учащихся*", "*пробное (входное) тестирование*", "*тест на полноту/целостность усвоенных знаний*", "*претест*", "*апробируемый тест*" jt.

1.2. Ülevaade testidest

Magistritöö raames on plaanis luua eeltest TTL-kursuse jaoks, mille osaleja on noor vanuses 16–26 ja milles tutvustatakse programmeerimise põhitõdesid programmeerimiskeele Python põhjal. Seega uurides maailmas kasutusel olevaid teste, lootsin leida praktikaid, mille abil

eelhinnatakse õppijate teadmisi ja oskusi programmeerimise aluste kursusel, ennekõike selliseid, kus kursus põhineb programmeerimiskeelel Python.

Oluline küsimus, millele tuleb antud eesmärgi saavutamiseks vastata on, mis on "programmeerimise aluseid tutvustav kursus". Kuidas kogemus on ülekantav Eesti oludele? Kas õpetatakse samu asju?

Ingliskeelses (USA) kirjanduses kasutatakse programmeerimise kursustest kirjutades lühendeid CS1 ja CS2 (*Computer Science 1, Computer Science 2*). Hertz (2010) uuris nende mõistete kasutamist. CS1 ja CS2 on mõisted 1978. aasta ACM'i (Association for Computing Machinery) arvutiteaduse õppekavast, mis koosnes 8 kursusest. Nendest kaks esimest olid arvutiteadusese sissejuhatavad kursused. Hertz näitab oma töös, et aastatega on arusaam "sissejuhatavast kursusest" muutunud ning hetkel puudub üldkokkuleppe, mida nende kursuste raames õpetatakse. Tihti (kuid kaugeltki mitte alati) mõeldakse CS1 all programmeerimise sissejuhatavat kursust ning CS2 all andmestruktuuride kursust (Kotkas, 2018).

Venekeelset kirjandust lugedes ei eristu üldist nimetust, pigem kirjutatakse üldiselt programmeerimise õpetamisest. Samal ajal on Venemaa puhul olemas üldised standardid, mille järgi riigis tuleb ülikoolis õpetada. Föderaalne riiklik kõrghariduse haridusstandard (vn ФГОС ВО[1] − *федеральный государственный образовательный стандарт высшего образования*, edasi riiklik standard**)** sätestab pädevuspõhise õppe**.** Standard peab tagama erinevates ülikoolides saadud hariduse võrreldavuse. Informaatika kohta kehtib standard 02.03.02[2] (vn *Фундаментальная информатика и информационные технологии (уровень бакалавриата*)), täpsema kirjelduse leiab mõnede ülikoolide lehekülgedel, näiteks programmeerimise kohta kehtib pädevus "32 (ОПК -3) –I"[3], mille järgi peab õppija omandama järgmised teadmised ja oskused: tunneb programmeerimiskeelte algoritmiliste konstruktsioonide semantikat ja süntaksit. Seega teoorias peaksid kõik ülikoolid õpetama samu pädevusi, kuid standardeid lähemalt uurides näeb, et need on väga üldised, mis tähendab, et iga ülikooli, kursuse, testi puhul peab eraldi uurima, mida sellega kontrollitakse.

Eestis samuti puudub üldine kokkuleppe, mida tuleb õpetada programmeerimise sissejuhatava kursuse käigus ning mis keel selle jaoks sobib paremini (Puniste, 2015).

[1] "Портал Федеральных Государственных Образовательных Стандартов Высшего Образования." Федеральные Государственные Образовательные Стандарты Высшего Образования, fgosvo.ru/fgosvo/92/91/4. (09.05.2020)

[2] Приказ Минобрнауки России от 12.03.2015 N 224 "Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии (уровень бакалавриата)" http://fgosvo.ru/uploadfiles/fgosvob/020302.pdf (09.05.2020)

[3] http://uup.samgtu.ru/sites/uup.samgtu.ru/files/prilozhenie_2_01.03.02.pdf (03.05.2020)

Käesoleva magistritöö jaoks otsisin ennekõike programmeerimise kursuste õppijate teadmiste/oskuste eelhindamiseks mõeldud teste. Üldised tulemused on tagasihoidlikud. Ingliskeelsetes allikates pigem konstateeritakse, et teste on vähem, kui on vaja. Valideeritud teste on väga vähe (Tew & Guzdial, 2010; Parker jt, 2016; Grover, 2020). Venekeelsete allikate järgi on teste samuti vähe. Mõnes kohas (nagu allpool kirjeldatud Kaasani Tehnoloogiaülikoolis) on mitu aastat järjest süstemaatiliselt arendatud tarkvarainseneride ettevalmistamisel pädevuspõhist õpet, mille parema läbiviimise jaoks viidi sisse pidev teadmiste hindamine, k.a eelhindamine ning loodi vastavad testid. Üldiselt on Venemaal pedagoogilised testid leidnud vähest kasutamist ajalooliste põhjuste tõttu: Nõukogude ajal olid tihtipeale lääne uuringute tulemused poliitilistel põhjustel vastuvõetamatud (Tchelyshkova, 2002). Eestikeelsete allikate järgi ei ole Eestis avaldatud teadusartikleid programmeerimiskursustel kasutatavate eeltestide kohta. Eeltestimisest kirjutab Karus (Karus, 2015) üldiselt kujundava hindamise kontekstis. Programmeerimise kursuste eeltestide kohta töid ei leidu. Internetis leidub infot, et sarnaseid teste on võetud kasutusele, näiteks korraldatakse Tallinna Tehnikaülikooli kursusel "IAS0090 Algoritmid ja andmestruktuurid" eeltest, mille eesmärgiks on selgitada õppijate teadmiste ja oskuste tase spetsiifiliselt programmeerimisel C-keeles. Kursuse veebilehel[4] on kirjas, et "õppejõud vajab neid andmeid selleks, et näha, milliseid lünki tuleb täita." On lootust, et tänu uutele gümnaasiumi programmeerimise kursustele, tekib parem võimalus viia läbi erinevaid uuringuid (ka eeltestide kohta).

Järgnevalt annan lühiülevaate kolme testi kohta (üks Venemaalt ning kaks USA-st). Venemaa näitena toon Kaasani riikliku teadustehnoloogilise ülikooli (vn Казанский национальный исследовательский технологический университет, ingl Kazan National Research Technological University, edasi Kaasani Tehnoloogiaülikool) lahenduse ning Ameerika Ühendriikide näidetena kaks Georgia Tehnoloogiainstituudi (ingl Georgia Institute of Technology) lahendust.

1.3. Pädevuspõhine õpe Kaasani Tehnoloogiaülikooli näitel

Alates 2008. aastast õpetatakse Kaasani Tehnoloogiaülikoolis "IT-süsteemid ja tehnoloogiad" erialal toetudes pädevuste hindamise formaadile (vn *метрический компетентностный формат*), selle ajaga on selle formaadi järgi hariduse saanud rohkem kui 1000 õppijat ning õppijate taset on uuringutega hinnatud paremaks, kui nende oma, kes õppisid "tavaliste" õppemeetoditega (Starygina & Nuriev, 2017). Süsteem on loodud Nurievi ning tema

[4] Kursus: IAS0090 Algoritmid ja andmestruktuurid, Nädal: Nädal 1: eeltest https://moodle-test.taltech.ee/course/view.php?id=22839&section=1 (03.05.2020)

töörühma poolt mitme aasta jooksul. Eesmärgiks on õpetada riikliku standardi järgi tulevasele insenerile kindlaksmääratud pädevused (vn *компетенции*), ehk rakendatakse pädevuspõhist õpet (vn *компетентностный подход*). Seega eriala kursuste planeerimisel sõnastatakse õpipädevused, mille omandamine lubab õppijal jõuda riiklikus standardis nimetatud pädevuseni. Õppekava läbitakse infosüsteemi abil, mis lubab asuda õppima järgmist õpipädevust alles siis, kui sellele loogiliselt (süsteemis kirjeldatud) eelnev pädevus on õpitud piisaval tasemel. Iga õpipädevuse omandamise tase on salvestatud süsteemi eraldiseisvana. Infosüsteem toetab õppijat õpipädevuste õppimisel pakkudes ülesandeid (automaatse kontrolli ning õppejõu määratud taseme abil), mis asuvad õppija lähima arengu tsoonis (Võgotski, 1996). Süsteem on osaliselt integreeritav Moodle'isse või on täismahus kasutatav ülikooli enda loodud õppesüsteemis (MYKNITU https://myknitu.ru/). Üliõpilase taseme määramiseks on võetud kasutusele mitmed elektroonilised testid: eeltestid enne kursuse algust, testid, mida läbitakse kursuse käigus ning kokkuvõttev test. Nende testide abil määratakse, kui hästi on üliõpilane antud õpipädevuse omandanud ning kas ta saab asuda õppima järgmist (Nguen & Ku, 2018; Starygina & Nuriev, 2017).

Kaasani Tehnoloogiaülikoolis loodud testide loomisel on järgitud allolevat protseduuri (kirjeldus põhineb (Gibadullina, 2017; Starygina & Nuriev 2017; Starygina, Nuriev & Garifjanov, 2018; Nuriev, Starygina & Samerhanov, 2018) artiklitel, kasutusel olevad lühendid POL, CHL, VAL, REL, REP, KSM, U on süsteemi autorite poolt pakutud parameetrite nimetused):

1. On loodud kaks andmebaasi küsimustega. Esimeses on küsimused, mis kontrollivad teadmiste täielikkust (POL, vn *полнота усвоенных знаний*). Teises on küsimused, mis kontrollivad teadmiste terviklikkust (CHL, vn *целостность усвоенных знаний*). Esimesed neist on vormis "ma tean, et ..." − need on küsimused, mis kontrollivad terminite, mõistete, definitsioonide teadmist. Teised on vormis "ma tean, kuidas …" − need on küsimused, mis kontrollivad, kas õppija teab, kuidas ja mida teha, ehk meetodite ja tehnoloogiate tundmist.
2. Eksperdid annavad testi kvaliteedile oma hinnangu U (on lubatud ka ainult ühe eksperdi kaasamine). Testi kvaliteedi hindamisel vaatleb ekspert testi nelja parameetrit, määrates iga parameetri kohta oma hinnangu väärtusega 0 kuni 1 (*VAL, REL, REP, KSM* $\in$ [0,1]. Mõisted ja nende seletused põhinevad süsteemi loojate interpreteerimisel.):
    a. VAL − testi sisu valiidsus. Valiidne on test juhul, kui kõik küsimused korrektsed ja sihipärased.

b. REL − testi relevantsus, kohasus. Kontrollitakse, kas küsimused kuuluvad testitavasse valdkonda.

c. REP − resti representatiivsus, esinduslikkus. Kontrollitakse, kas kõik valdkonda teemad esinevad testis.

d. KSM − küsimuste kallutatuse koefitsient (vn *коэффициент смещения*), mis näitab, kas eelnimetatud teadmiste täielikkust ning terviklikkust kontrollivaid küsimusi on testis võrdselt.

Edasi kontrollitakse ekspertide hinnangute vastavust (ehk kas tulemused on sarnased, nt kui üks ekspert hindab küsimuse 0,2 ning ülejäänud viis 0,8 vääriliseks, siis hindamised ei ole vastavuses). Hinnangute vastavust uuritakse Kendalli kooskõlakordaja (vn *коэффициент конкордации Кендалла-Смита*) arvutamise ja selle statistilise olulisuse uurimise abil (meetod on kirjeldatud põhjalikult (Nuriev jt, 2018) artiklis). Kui vastused ei ole vastavuses, võib sellel olla kaks põhjust: ekspertide tase erineb väga palju või puuduvad spetsialistide hulgas kokkulepped, kuidas antud probleemi hinnata. Esimesel juhul moodustatakse uus ekspertide rühm, teisel juhul konstateeritakse, et "ühiskond pole ekspertiisi jaoks valmis" (Nuriev jt, 2018). Ekspertiisi tulemusi võetakse kasutusele ainult siis, kui see läbis hinnangute vastavuse kontrolli.

Edasi arvutatakse välja eksperthinnangute aritmeetilised keskmised iga parameetri kohta. Kogu testi kvaliteedi hindamiseks arvutatakse leitud keskmiste geomeetriline keskmine:

$$U = \sqrt[4]{VAL_{kesk} \cdot REL_{kesk} \cdot REP_{kesk} \cdot KSM_{kesk}}$$

Eksperdid lepivad kokku, milline U väärtus on halb, hea, väga hea või suurepärane. Enamasti kehtib kokkulepe, et kui $U \geq 0,8$, siis on test suurepärane.

3. Järgmise sammuna hindab ekspert, kui palju tal läheb aega iga küsimuse lahendamise peale (minutites).

4. Testile määratakse selle lahendamiseks kuluv aeg (kolmekordne aeg, mis läks eksperdil küsimustele vastamisele). Seejuures on süsteemi autorid leidnud, et testi lahendamine ei tohi eksperdil võtta aega rohkem kui 20 minutit. Vastasel juhul lisandub õppija väsimus, mis ei luba teadmisi/oskusi korrektselt hinnata.

5. Test viiakse läbi ning hinnatakse õppija tulemusi: kas kompetents on omandatud. Selleks leitakse teadmiste täielikkust POL ning terviklikkust CHL kontrollitavate

küsimuste tulemuste keskmine. Tulemuste keskmised korrutatakse omavahel: $POL_{kesk} \cdot CHL_{kesk}$ ning saadakse testi lõpptulemus.

6. Kursust kokkuvõtva testi puhul arvutatakse lisaks tõenäosus, millega tulemus kehtib (Nuriev, Starygina & Ahmetshin, 2015). Selleks leitakse õppija kõikide kursuse jooksul saadud tulemuste aritmeetiline keskmine. Nt kokkuvõtva testi tulemus on 0,63, lisaks kursusele jooksul on õppija läbinud 3 testi, vastavad tulemused olid 0,87, 0,83, 0,79. Nende aritmeetiline keskmine on 0,83. Seega salvestatakse ning väljastatakse lõpptulemus: õppija omandas kompetentsi tulemusega 0,63 tõenäosusega 0,83.

Kaasani Tehnoloogiaülikoolis kasutusel olevat süsteemi on aastatega kontrollitud ning täiendatud, testide kvaliteedi hindamiseks on võetud kasutusele eksperthinnangud, teoreetilist baasi täiendatakse, kogutakse statistilisi andmeid süsteemi töö hindamiseks (Nuriev jt, 2018). Testide koostamise ning kontrollimise protseduuri on täpselt kirjeldatud ning selle tõhusust on uuritud mitme uuringutega. Kõik uuringud on viidud läbi Nurievi ja/või Starygina juhendamisel, infot välise hindamise kohta ei õnnestunud leida.

1.4. Programmeerimiskeelest sõltumatu programmeerimisoskusi hindav test FCS1 (Georgia Tehnoloogiainstituut)

2010. aastal kaitses Allison Elliott Tew Georgia Tehnoloogiainstituudis (Georgia Institute of Technology) oma doktoritöö, mille tegemise käigus lõi ta testi, mis aitab hinnata õppijate programmeerimise aluste oskusi/teadmisi, kusjuures test ei sõltu programmeerimiskeelest, mida õppijad on õppinud ja/või asuvad õppima. Töö ajendiks oli valideeritud testide vähesus arvutiteaduse valdkonnas. Valideerimata testide puhul ei saa olla kindel, mida tulemused näitavad − võib-olla testi mittetäiuslikkust? Samuti oli Tew' jaoks oluline luua test, mille abil saab võrrelda üldist programmeerimisoskust, mitte spetsiifilise programmeerimiskeele kasutamise oskusi (Tew & Guzdial, 2010).

Oma doktoritöö jaoks (Tew, 2010) kohandas Tew pseudokoodi (ingl *pseudo-code*), mis aitas luua keelest sõltumatu valideeritud testi FCS1 (Fundamental Computer Science 1) programmeerimise sissejuhatavate teadmiste hindamiseks. Eesmärgiks oli luua test, mis oleks kursuste ülene ja seega kasulik uurijatele, sõltumata konkreetsest kursusest.

Testi loomise meetodit kirjeldan Tew' doktoritöö (Tew, 2010) ning Tew' ja Guzdiali artiklite (Tew & Guzdial, 2010; Tew & Guzdial, 2011) põhjal.

Testi loomise sammud:

1. Kontrollitavate konstruktide (ingl *constructs*) kaardistamine, selle põhjal testi sisu määratlemine, testi spetsifikatsiooni loomine (mida uuritakse; küsimuste formaat; kuidas hinnatakse, ehk skaala).

   Testi skoobi defineerimisel lähtus Tew erinevatel kursustel õpetatavatest teemadest. Uurimiseks olid 12 USAs levinud õpikut ning nende sisu analüüsi tulemusena tekkis umbes 400 õpetavast mõistest nimekiri. Skoobi vähendamiseks võttis Tew appi arvutiteaduste sissejuhatavate kursuste soovitusliku raamistiku (Computer Curricula 2001). Peale ülevaatamist jäi 188 mõistet ning ühise kõikehõlmava, kuid mahu poolest mõistliku skoobi otsingud jätkusid. Edasi määras ta põhilised semantilised teemad (muutuja, omistamine, matemaatilised väljendid jne), aluste konstruktid (ingl *construct*) ning otsustas loobuda primitiivsete andmetüüpide (ehk *integer*, *boolean*, *string*), sisendi/väljundi ja nende töötlemise mõistetest (kuna need on enamasti väga keelespetsiifilised). Objekti konstrukti piiras Tew klassi defineerimisega ning meetodi väljakutsega.

   Testi lõplik skoop koosnes järgmistest konstruktidest:

   a. Alused (muutujad, omistamine jne)
   b. Loogika operaatorid
   c. Tingimuslause (*if/else)*
   d. Määratud tsükkel (*for*, edasi for-tsükkel) (ingl *definite*)
   e. Määramata tsükkel (*while*, edasi while-tsükkel) (ingl *indefinite*)
   f. Järjendid
   g. Funktsiooni/meetodi parameetrid
   h. Funktsiooni/meetodi tagastatavad väärtused
   i. Rekursioon
   j. Objektorienteeritud programmeerimise alused (klassi defineerimine, meetodi väljakutse)

   Testis oli iga konstrukti kohta 10% küsimustest. Testi formaadiks oli valitud mitme vastusevariandiga küsimuste formaat. Tulemuste interpreteerimine oli kriteeriumide põhine ja ei sõltunud kaastestitavate tulemustest. Punktide jaotus: 1 punkt korrektse vastuse eest, 0 punkti vale vastuse eest (küsimusi, millele vastust ei antud, ei võetud arvesse).

2. Testi spetsifikatsiooni hindamine ekspertide poolt.

   Testi spetsifikatsiooni hindamise eesmärgiks oli teha kindlaks, et kõik konstruktid esindatud adekvaatselt ja üleliigseid lisatud ei ole.

   Programmeerimisaluste (CS) ekspertide rühm (osalejad jäid doktoritöös mainimata) vaatas üle testi spetsifikatsiooni. Nad andsid hinnangu testitavate konstruktide nimekirjale, küsimuste formaadile ning hindamismeetodile. Testi autor esitas koos testi spetsifikatsiooniga testiküsimuste näidiste esialgse variandi. Tulemuste alusel täiendas Tew konstruktide sisu ning alustas küsimuste välja töötamisega.

3. Küsimuste panga loomine.

   Peale seda, kui testi spetsifikatsioon oli ekspertide poolt kooskõlastatud, lõi Tew küsimuste panga, kus küsimused katsid kõik spetsifikatsioonis olevad mõisted.

   Iga konstruktsiooni kohta olid loodud kolme erinevat tüüpi küsimused (valideerimisprotsessi mitte läbinud küsimuste näide lisas 1):

   a. Mõistete küsimused (ingl *definitional*) − kontrollitakse definitsioonide tundmist, üldist arusaama.
   b. Koodi jälgimise küsimused (ingl *tracing*) − kontrollitakse, kas õppija suudab ennustada koodi käivitamise järgset käitumist (nt muutuja väärtus peale tsükli töö lõppu).
   c. Koodi lõpetamise küsimused (ingl *code completion*) − kontrollitakse õppija oskust koodi kirjutada (“täida tühi koht” tüüpi küsimused).

4. Testi keelest sõltumatuse verifitseerimine ja küsimuste piloteerimine

   Tew pakkus kasutada testi jaoks pseudokoodi. Pseudokoodi adapteeriti alustavate programmeerijate juhenditest (Whitfort (n.d.) ja Shackelford (1997) viidatud Tew, 2010: 41 kaudu). Süntaks oli viidud miinimumi: ei ole semikooloneid, reserveeritud sõnad kirjutatakse suurte tähtedega, programmiplokke lõpetatakse konkreetsete käsklustega (nt END-FOR). Enne testi sooritamist jagati õppijatele pseudokoodi tutvustav juhend (A. Tew' pseudokoodi juhendi esimene lehekülg on toodud lisas 2).

   Keele ja testi valiidsuse tõestamiseks viidi läbi mitme rühmaga eksperiment. Õppija sooritab testi kaks korda (ühe nädala vahega): pseudokoodis ning enda õpitud keeles (Java, Mathlab, Python). Teise testi jaoks kasutati küsimuste pangast samade konstruktide kohta alternatiivseid küsimusi.

   Testi valiidsuse tõestamiseks viis autor läbi valjusti mõtlemise (ingl *think aloud*) intervjuud. Järgmise sammuna tegi Tew analüüsi kirjeldava statistika ning üksikvastuste teooria (ingl IRT − *Item Response Theory*) abil.

5. Reliaabluse uurimine

   Reliaabluse uurimine (kas korduvate testide tulemused on järjepidevad, ehk on usaldusväärsed, reliaablid) jäi nimetatud doktoritöö käigus realiseerimata.

Tew' loodud töö on inspireerinud uurijaid keelest sõltumatu testide uurimiseks, loomiseks, kopeerimiseks. Nt uurisid Lee ja Ko (2015) sarnastel põhimõtetel loodud testide abil programmeerimise veebikursuste efektiivsust ning leidsid, et struktureeritud kursused on efektiivsemad, kui vaba õppekavaga (nt Scratchi sarnases keskkonnas), kus põhirõhk on avastusõppel (Lee & Ko, 2015). Tew' test esitati doktoritöö kaitsmisel ainult paberil ning ainult kaitsmiskomisjoni liikmetele: kaitsmaks seda kopeerimise ja/ning internetis levitamise eest. Test on kaitstud laialdase kasutamise eest, vastasel juhul muutuks test õppijatele kättesaadavaks ning testi ei saaks enam kasutada uurimistööde jaoks, sest siis annaks selline uuring vastuse õppijate mälu, mitte programmeerimisoskuste kohta.

1.5. Programmeerimiskeelest sõltumatu programmeerimisoskusi hindav test SCS1 (Georgia Tehnoloogiainstituut)

Olukorras, kus Tew' test on kaitstud ning selle kasutamise võimalused on piiratud, tõstatasid Parker jt (2016) küsimuse, kuidas saab testi kiiresti kopeerida (ingl *replicate*), et saada valideeritud testi oma uuringu jaoks, vältimaks uue valideeritud testi loomise raskusi. Üks artikli kaasautoritest aitas Tew' testi loomisel ning seega omas seadusliku juurdepääsu testile, kuid mitte selle levitamisele. Parker jt kopeerisid Tew' loodud FCS1 testi, uue testi nimeks sai SCS1 (Second Computer Science 1 Assessment).

Edasi esitan Parkeri jt kirjeldatud testi kopeerimise protseduuri (Parker jt, 2016) artikli põhjal. Uurimisrühma eesmärgiks oli luua valideeritud test, mille abil saaks hinnata õppijate programmeerimise aluste tundmist. Seega jäeti samaks konstruktid (“Alused”, “Loogika operaatorid”, “Tingimuslause”, “Määratud tsükkel”, “Määramata tsükkel”, “Järjendid”, “Funktsiooni/meetodi parameetrid”, “Funktsiooni/meetodi tagastatavad väärtused”, “Rekursioon”), küsimuste tüübid (mõisted; koodi jälgimine; koodi lõpetamine) ning küsimuste arv (27).

Testi kopeerimise protseduur:

- Küsimuste kopeerimine.

  Iga konstruktsiooni kohta võeti iga tüübi küsimus, asendati selles sõnaline probleem, muutujad ning vastused. Küsimuste loomisel ei uuritud otseselt nende raskust. Vanade

küsimuste kopeerimist kasutati uute küsimuste loomise asemel, et esialgse testi valiidsus kanduks üle uuele testile (küsimuse näide lisas 3).

- Intervjuud testitavatega.

  Küsimused, vastused (k.a peibutusvastused) olid muutunud, seetõttu viidi läbi 3 intervjuud testitavate õppijatega. Oli oluline tuvastada, kas uus tekst on õppija jaoks sama tähendusega kui küsimuse koostaja jaoks, kas eksitavad vastused on õppija jaoks sama tähendusega, kui see oli mõeldud FCS1 testi autorite poolt. Intervjuud viis läbi üks autor, kes hiljem arutles nende üle teise autoriga. Intervjuu viidi läbi testi käigus: esimene õppija lahendas esimese poole testist nii, et lahendades ta samal ajal arutles valjusti, kuidas ja mida ta peab antud küsimuse puhul tegema ja miks; teine õppija lahendas samal viisi teise poole testist (ajalise kokkuhoiu mõttes). Kolmanda intervjuu käigus esitas intervjueerija küsimusi nende küsimuste kohta, mis tekitasid kõige rohkem probleeme esimese kahe intervjuu käigus. Iga intervjuu võttis 90 minutit aega. Vead, mis tulid välja olid: kirjavead (probleemi püstitamisel vale sõna kasutamine); probleemid sõnastusega (ähmased omadussõnad, viga alus-öeldis kombinatsioonis); segased küsimused; üks küsimustest oli liiga kerge (analüüsi tulemusena muudeti see raskemaks); mittemõistlikud vastused (õige vastus puudub, mitu õiget vastust, vastus on mõttetu).

- Testi valideerimine.

  Testi valiidsuse tõestamiseks viidi läbi 183 õppijaga kaks testi. Pool rühmast tegi FCS1 ning teine pool SCS1, nädala pärast vastupidi. Testi lahendamiseks oli antud üks tund. Test viidi läbi programmeerimise sissejuhatava kursuse lõpupoole. Õppijad said tasu/tunnustamist uurimuses osalemise eest (tavaliselt eksamihinne lisapunktide näol). Testi lahendamine oli viidud läbi eksamiks ettevalmistava harjutusena. Pearsoni korrelatsiooni abil leiti, et uus test on valiidne.

Parker jt kontrollisid oma SCS1 testi klassikaline testi teooria (ingl CTT − *Classical Test Theory*), abil. Kolm aastat hiljem viisid Xie jt (2019) läbi SCS1 sõltumatu analüüsi, kasutades selleks üksikvastuste teooria (ingl IRT − *Item Response Theory*) analüüsi. Autorid (Xie jt, 2019) tõid artiklis välja, et üksikvastuste teooria abil tehtud analüüs annab parema tulemuse, kui klassikaline testi teooria, kuna aitab leida vastuse küsimusele, miks testi küsimus oli raske (CTT annab vastuse küsimusele, millised testi küsimused on rasked). Testitavate arv oli 507. Küsimuste sõltumatuse osas jõuti järeldusele, et see on kehtiv, kuna test viidi läbi elektrooniliselt, ilma, et oleks võimalus kõiki küsimusi korraga näha ning küsimuste vahel liikumiseks pidi liikuma mööda kõiki vastatud küsimusi, sama konstrukti küsimused ei asetsenud kõrvuti. Analüüsi tulemusena jõuti järeldusele, et kolm küsimust ei ole CS1-ga

seotud. Need on küsimused funktsiooni ning rekursiooni kohta. Autorid tõid välja, et ka teised uuringud jõudsid järeldusele, et need teemad ei sobi programmeerimise aluste üldtesti. Üks põhjustest on see, et erinevates programmeerimiskeeltes on nende teemade roll erinev, seega testi tulemus sõltub programmeerimiskeelest, mida testitav on õppinud. Veel nelja küsimuse osas leiti, et need on liiga rasked. Nii Xie kui ka Parkeri uuringud mõlemad näitavad, et test on liiga raske alustavate õppijate jaoks. Parker jt jõudsid oma testi analüüsimisel samuti järeldusele, et mõned küsimused on liiga rasked, Xie täiendav uuring avastas, et rasked on need samad küsimused, lisaks Xie uuringu tulemusena sai selgeks, et osad küsimused ei sobi, kuna konstruktid, mille kohta need küsimused on, ei ole kursuse skoobis. Paremaid tulemusi näitasid IT-erialade üliõpilased. Xie jt analüüsi tulemuseks eraldati 7 küsimust 27-st ülekontrollimisele, kas nad ikka sobivad antud testi.

Xie täiendav uurimus näitas, et testi kopeerimine on võimalik. Xie jt järeldasid, et SCS1 puhul tuleb teha täiendavaid uuringuid ning nad tõid välja, et mõned küsimused olid testitavatele väga rasked (eriti nende jaoks, kes õppisid programmeerimist lisaainena, mitte oma põhierialana) ning seega võib olla ei ole SCS1 kõige parem variant kasutamiseks eeltestina, kuigi lõpptestina on hea. Xie uuringu fookuses oli SCS1 test. Analüüsi tulemusena leiti, et küsimused rekursiooni kohta ei sobi programmeerimise aluste testi. SCS1 on FCS1 koopia, mis tähendab, et rekursioon esineb konstruktina FCS1 testis. Seega peab järeldus rekursiooni mittesobivuse kohta kehtima ka FCS1 suhtes.

1.6. Kokkuvõte

Käesolevas peatükis kirjeldasin, kuidas ja milliseid teadmisi ning oskusi hindavaid (eel)teste viiakse läbi programmeerimiskursuste raames. Eesmärk leida teste gümnaasiumis õpetatavate programmeerimiskursuse jaoks jäi saavutamata. Leidub töid, milles kirjutatakse ülikoolides õpetatavate programmeerimise sissejuhatavate kursuste kohta. Nendes töödes on kirjeldatud mitut enda testi loomise protseduuri jaoks olulist momenti. Kaasani Tehnoloogiaülikooli kogemus on oluline, sest selles on rakendatud pädevuspõhist õpet. Lisaks on antud süsteemi kohta mitu tööd, mis aitavad aru saada, kuidas hinnatakse selle süsteemi testi valiidsust ning mida tähendab testi tulemusena saadud hinne õppija jaoks − kuidas sõltub kogu kursuse hinne lõpp- ja vahetestidest. Tew' (FCS1, Georgia Tehnoloogiainstituut) töö on oluline, kuna oma testi loomisel saab toetuda Tew' töös kirjeldatud testi skoobi määramise tulemusele, samuti on Tew' töödes hästi kirjeldatud testi loomise ning kontrollimise metoodikad. Parker jt (SCS1, Georgia Tehnoloogiainstituut) töö on oluline, kuna selles tõestatakse, et valideeritud testi saab kopeerida, säilitades selle valiidsuse. Kokkuvõttes saab kirjeldatud tööde põhjal välja pakkuda oma valiidse testi loomise protseduuri.

2. Testi loomine

Käesolevas peatükis kirjeldan eelmises peatükis uuritud testide loomise kogemuste põhjal protseduuri, mida järgides loon TTL-kursuse jaoks testi. Edasi kirjutangi täpsemalt testi loomisest: testi skoobist ning selle defineerimise protsessist, sõnastan testi spetsifikatsiooni, millega kirjeldan testi küsimuste formaati ja hindamist. Kirjutan testi küsimuste esimese variandi loomisest ning ekspertide arvamusest nendest.

2.1. Testide kopeerimine

Eelmises peatükis uuritud testid on leidnud kasutust vastavalt püstitatud eesmärkidele. Kaasani Tehnoloogiaülikoolis loodud testid on Venemaal riiklikul tasemel kehtivate õppekavade õpipädevuste põhised. Georgia Tehnoloogiainstituudis loodud testid FCS1 ja SCS1 on loodud eesmärgiga uurida üldiseid programmeerimisoskusi sõltumata õpetavast programmeerimiskeelest. Käesoleva magistritöö eesmärgiks on luua test TTL-kursuse jaoks. Oma testi loomisel ei saa võtta teiste loodud teste üks ühele, vaid tuleb üle vaadata, mida uus test kontrollima hakkab, tuleb määrata skoop. Järgmine küsimus on, kuivõrd on võimalik testi kopeerimine teise kultuuri- ning keeleruumi. Eespool kirjeldasin keelest sõltuvaid raskusi testide uurimisel. Oma töö raames uurisid Parker jt (2016) keelespetsiifikast tulenevaid muresid valideeritud testi kopeerimisel. Nad kirjeldavad oma artiklis (Parker jt, 2016) raskusi, mis esinesid nende testide kasutusele võtmisel Saksamaal (SCS1-G). Test tõlgiti inglise keelest saksa keelde ja siis tagasi. Esialgse ja edasi-tagasi tõlgitud versioonide võrdlemisel korrigeeriti saksakeelset testi. Kaks inimest vaatasid testi uuesti üle, ülevaatamisel täheldati keelelisi ning kultuurilisi erinevusi. Uued muudatused viidi sisse. Parkeri jt testis on kõik küsimused mitmikvastusega (viis sisulist vastust). Saksamaa testi autorid võtsid kasutusele kuuenda vastusevariandi: “Pole kindel”, millega nad soovisid vähendada võimalust, et tulevad väärpositiivsed vastused (äraarvamine). Lisaks arvasid Saksamaa autorid, et lisavastus annab paremaid võimalusi uurimiseks, mis põhjusel jäi küsimus vastamata (ei olnud aega, ei tea vms). Parker jt jõudsid järeldusele, et testi valiidsus väheneb selle tõlkimise tagajärjel, testi valiidsust vähendas lisaks Saksamaa autorite lisatud kuues vastusevariant.

Uuritud testidest mitte ükski pole otseselt rakendatav TTL-kursuse testina, piiranguteks on testi skoop ning keel. Lisaks on testid kaitstud kopeerimise eest autoriõigustega ja/või valiidsuse säilitamise eesmärgil. Samas nende testide loomise protseduurid on põhjalikult

kirjeldatud ning põhjendatud. Kirjeldatud protseduure toetab üldine testide loomise teooria (Tchelyshkova, 2002). Tänu sellele on need protseduurid rakendatavad oma testi loomiseks.

Loodav test on õpiväljundite põhine, kursuse oluline osa on programmeerimiskeele Python aluste õppimine, seega test kontrollib ka Pythoni spetsiifilisi teadmisi.

Testi loomisel lähtun ennekõike Tew' testi (Tew, 2010) loomise protseduurist:

1. Kontrollitavate konstruktide (ingl *construct*) kaardistamine.
2. Testi spetsifikatsiooni loomine (mida uuritakse; küsimuste formaat; kuidas hinnatakse).
3. Testi spetsifikatsiooni hindamine ekspertide poolt.
4. Küsimuste panga loomine.
5. Testi keelest sõltumatuse verifitseerimine ja küsimuste piloteerimine.
6. Testi reliaablus.

Oma testi loomise protseduurist jääb välja 5. punkt, kuna loodav test on mõeldud ennekõike kursuse "Tehnoloogia tarbijast loojaks" jaoks ning sellega kontrollitakse Pythoni spetsiifilisi teadmisi. Valiidsuse täielik tõestamine jääb selle töö skoobist välja. Testi reliaablus on oluline, kuid jääb antud töö skoobist välja.

2.2. Testi skoop ning kontrollitavate mõistete kaardistamine

Testi skoobi määramine on kõige raskem ja samal ajal kõige vajalikum etapp testi loomisel, sellest sõltub kogu testi kvaliteet (Tchelyshkova, 2002). Nurievi rühma jaoks oli ennekõike oluline kontrollida õpipädevuste õppimist (pädevuspõhine õpe). Pädevuspõhise õppe organiseerimisest kirjutab oma töös Mulin (2012), kus ta toob välja, et kõige sobivamaks õpetamise organiseerimise mudeliks on integreeritud mudel (distsiplinaarne õppe ja pädevuspõhine õppe lõimimine), mida teostatakse õppe-eesmärkide / õpiväljundite seadmise kaudu. Õpipädevuste sõnastamise kõige üldisem tase on nt riiklikus õppekavas sõnastatud pädevused, mille sõnastus on tavaliselt väga üldine. Nende täpsustamiseks sõnastatakse täpsemad pädevused õppekava, õppemooduli, õppeaine ja ainetunni jaoks. Tew ja Guzdial (Tew, 2010; Tew & Guzdial, 2010; Tew & Guzdial, 2011), Parker jt (2016) ning Higgins jt (Higgins, McAvinia, O'Leary & Ryan, 2019) lähtuvad õppijate teadmiste ja oskuste hindamisel õpetavate mõistete ja konstruktide kaardistamisest. Edasi uurisin, kuidas ja mida on otstarbekas kontrollida testiga TTL-kursuse raames. Alustasin õpiväljunditest ning siis uurisin, mis mõisteid kursuse jooksul õpetatakse.

Esimene samm on uurida ja määrata õpetavaid õpiväljundeid nii kogu kursuse kui ka nädalate kaupa. Üldised õpiväljundid on toodud kursuse leheküljel (TTL, 2020). Mõisteid kaardistan "Programmeerimise" õpiku (Tõnisson, E., Palts, Säde, Tõnisson, K. jt, 2019) abil, toetudes seejuures Tew' (2010), Parkeri jt (2016) ning Higginsi jt (2019) töödes defineeritud mõistete konstruktidele. Uurin mõistete õpetamist terve arvestustöö tulemusena, aga ka nädala teemade kaupa.

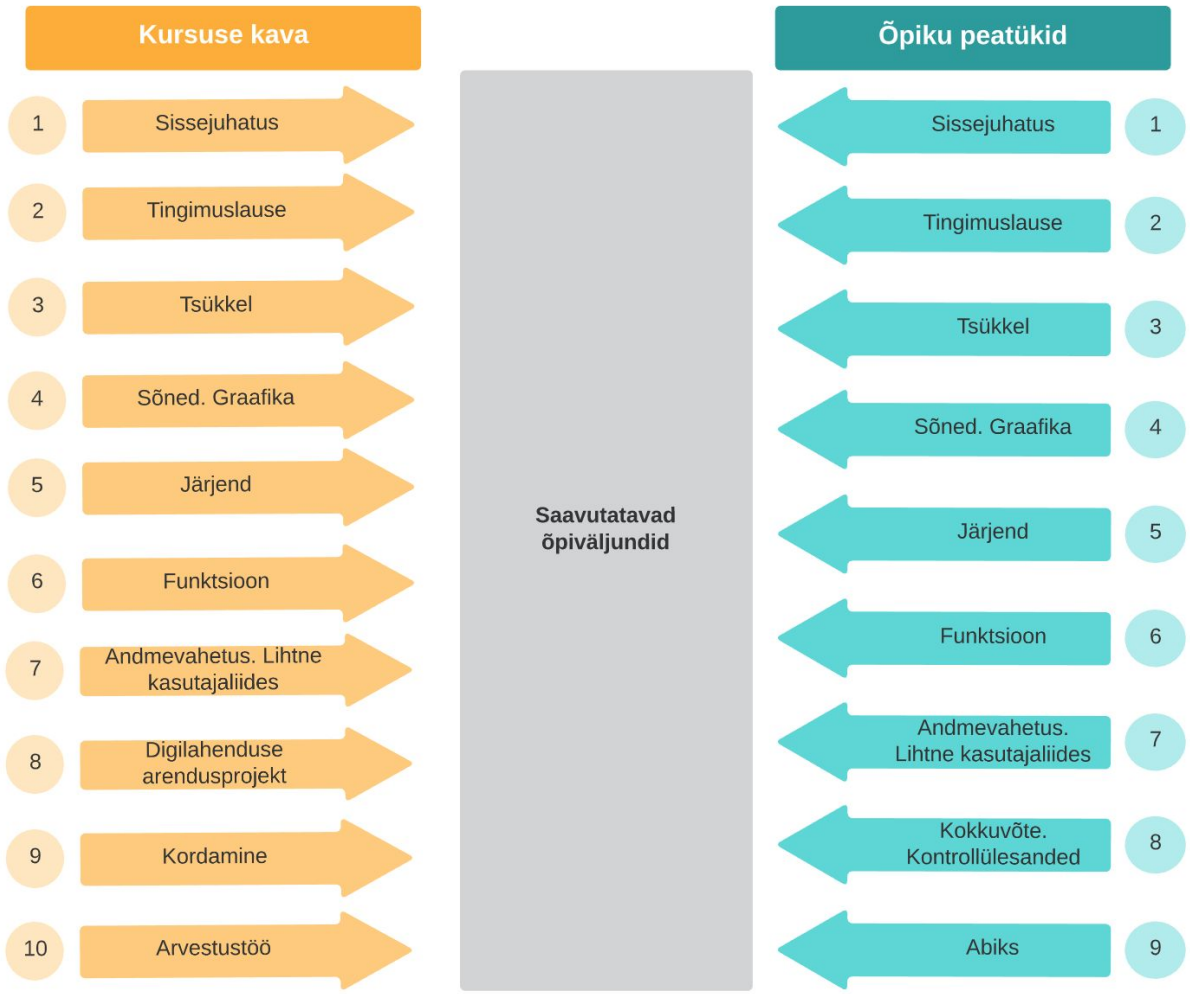
Kursuse õpiväljundid (TTL, 2020):

1) teab algoritmilise mõtteviisi põhitõdesid ning oskab seda mõtteviisi rakendada;
2) oskab kirjeldada ja kasutada programmi elemente (muutuja, tsükkel, funktsioon jne) programmeerimiskeeles Python;
3) oskab luua lihtsamaid igapäevaelu lihtsustavaid rakendusi;
4) oskab analüüsida probleeme, mille lahendamiseks piisab koolitusel käsitletavast materjalist, ning valib probleemi lahendamiseks sobiva algoritmi;
5) oskab testida loodud programmi tööd, leiab ja parandab puudused;
6) oskab analüüsida etteantud programmikoodi ilma seda käivitamata;
7) oskab leida lähtekoodist vigu ning koodi siluda.

Kursuse sõnastatud õpiväljundid on üldised, nimetatud on üldised põhimõisted (muutuja, tsükkel, funktsioon). Õpiväljundite sõnastus on rõhuga praktilisusele: õppija oskab programmeerida lihtsamaid igapäevaelu lihtsustavaid rakendusi. Skemaatiliselt võib nimetatud väljundeid jagada nelja rühma: baasteadmised, analüüsimisoskused, programmeerimise oskused ning oskus luua lihtne rakendus (Joonis 1).

Joonis 1. Kursuse "Tehnoloogia tarbijast loojaks" õpiväljundid rühmade kaupa

Õpiväljundite omandamise toetamiseks on kümmenädalane kursuse kava (TTL, 2020) ning uus programmeerimise õpik (Tõnisson jt, 2019). Oma ülesehituselt on kursusekava ning õpik sarnased, õpik toetab kursuse läbiviimist (Joonis 2). Nii kursuse, kui ka õpiku esimesed teemad on sissejuhatus (1), tingimuslause (2), tsükkel (3), sõned ja graafika (4), järjend (5), funktsioon (6), andmevahetus ja lihtne kasutajaliides (7). Alates kursuse kaheksandast nädalast alustatakse varem õpitu rakendamise ja kontrollimisega: digalahenduse arendusprojekt (8), kordamine (9), arvestustöö (10). Ka õpikus on alates 8. teemast kokkuvõtvad peatükid: kokkuvõtte, kontrollülesanded (8), abiks (9).



Joonis 2. Kursuse kava ning programmeerimisõpik

Kursuse kava ning õpik on tunduvalt konkreetsemad kui taotletavad õpiväljundid. Mõlemas paistavad välja Tew' töös sõnastatud üldised programmeerimisoskuste konstruktid (2 kuni 6: tingimuslause, tsükkel, sõned, järjend, funktsioon). Samuti on näha, et suur osa kursusest on pühendatud õpitud teadmiste rakendamisele loomingulisuses ülesandes − digilahenduse arendusprojektis.

Analüüsides edasi "Tehnoloogia tarbijast loojaks" kursuse ning õpiku teemade sisu, moodustub kontseptuaalne mõistekaart (Joonis 3), mis aitab aru saada, milliseid programmeerimise teadmisi ja oskusi õpilane kursuse läbimisel omandab. Mõistekaardi struktuuri eeskujuks on võetud Higginsi jt (2019) uuritava programmeerimise kursuse teemade loetelu. Higgins jt toovad välja järgmised kursuse teemad (kogu kursuse kestus 24 nädalat):

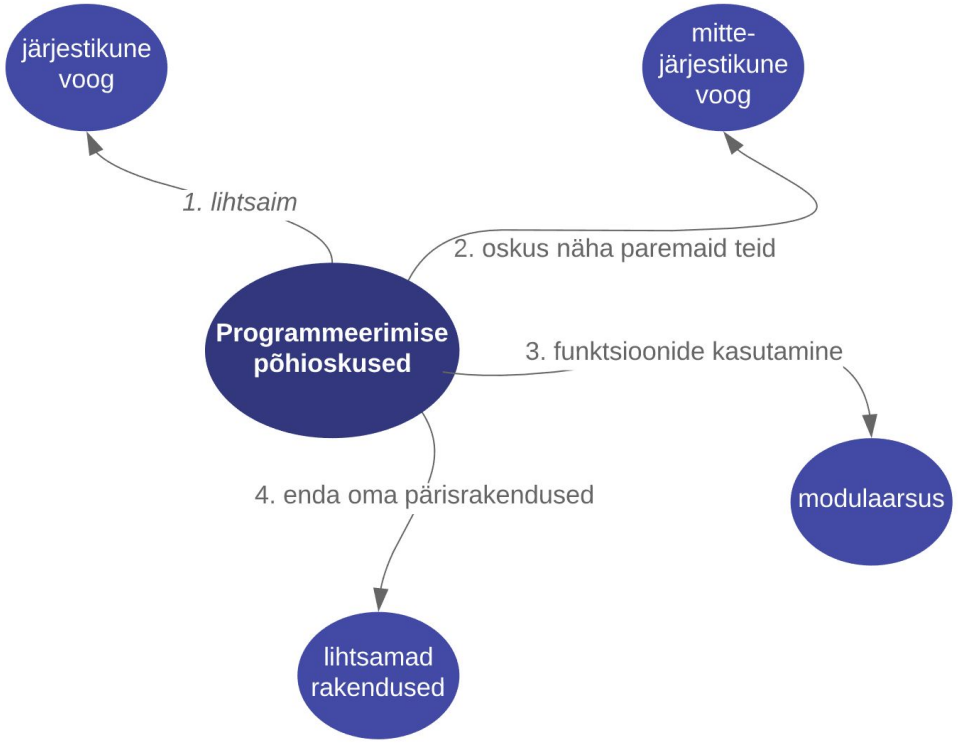
1. Järjestikune voog (ingl *sequential flow*) − nt muutujad, sisend, väljund − 1.–4. nädal.
2. Mittejärjestikune voog (ingl *non-sequential flow*) − nt tingimislause, tsüklid − 5.–9. nädal.

3. Modulaarsus (ingl *modularity*) − nt funktsioonid, parameetrid, muutuja skoop − 10.–15. nädal.
4. Objektorienteeritud interaktsioon/käitumine (ingl *object-oriented interaction/behaviour*) − uute andmetüüpide defineerimine (klass, muutuja, meetod) 16.–24. nädal.

Higginsi jt uuringus (2019) leiti, et nimetatud kursuse struktuur toetas õppijate teadmiste/oskuste õppimise sügavust.

Paigutasin "Tehnoloogia tarbijast loojaks" kursusel õpetatavad teemad (Higginsi jt teemade eeskujul) nelja programmeerimise põhioskuste konstrukti alla (Joonis 3):

1. Järjestikune voog − lihtsaim, mida saab programmeerida.
2. Mittejärjestikune voog − oskus rakendada mittelineaarseid lahendusi.
3. Modulaarsus − funktsioonide kasutamine, koodi taaskasutamise oskus.
4. Lihtsamad rakendused − teemad, mis aitavad luua oma rakendusi. "Käegakatsutav" tulemus õppija jaoks motiveeriv väljund, mille teostamiseks õpitakse lisateemasid.



Joonis 3. "Tehnoloogia tarbijast loojaks" kursusel käsitletavad programmeerimise põhioskused

Iga nimetatud konstrukti all on mitmed programmeerimise üldoskuste või kursuse mõistes olulised alamteemad (Joonis 4).

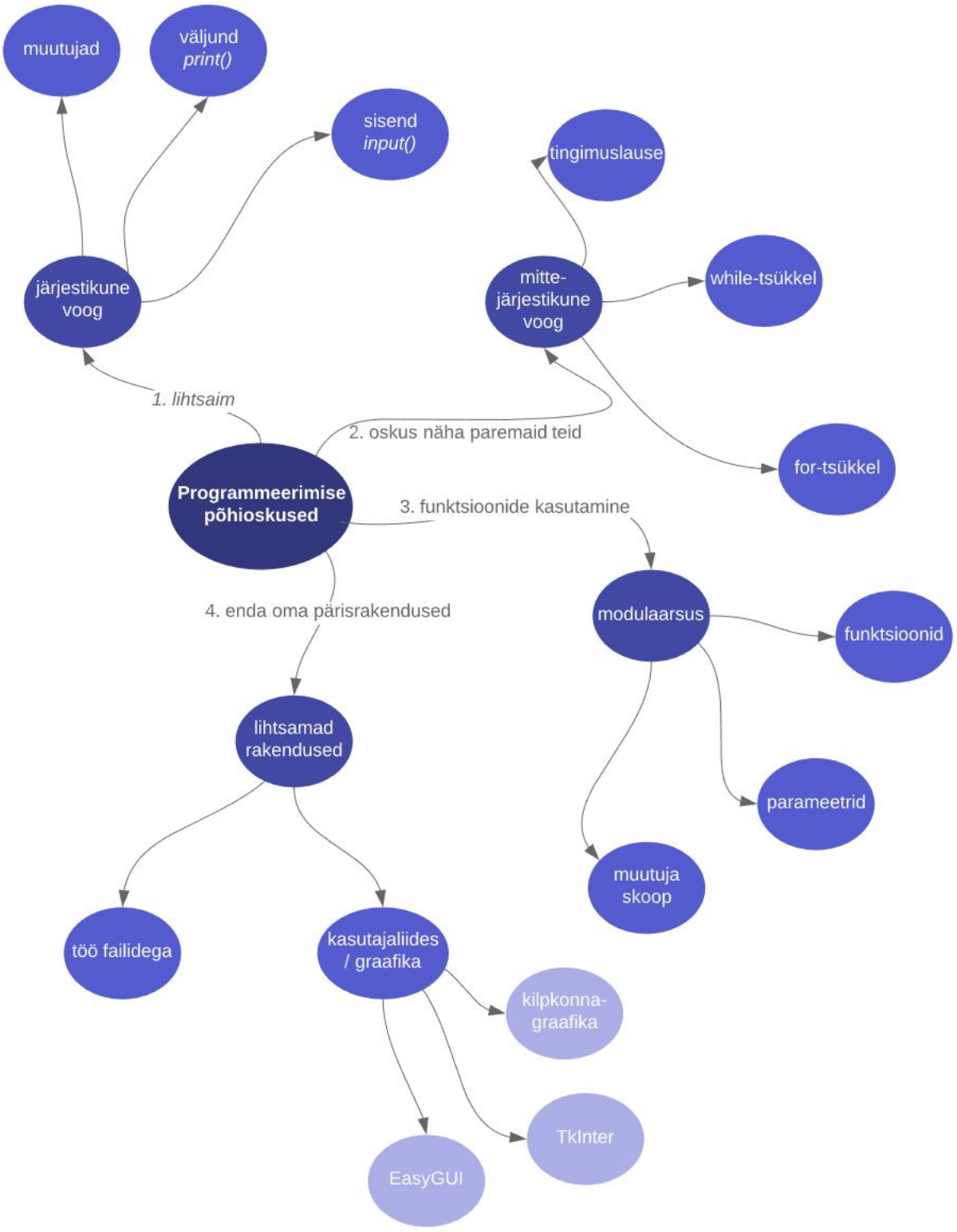
Konstruktide vahel liikumine eeldab eelmisest konstruktist arusaamist. See tähendab, et kui on selge järjestikune voog (muutujad, sisend ja väljund), saab alustada järgmise konstruktiga, ehk mittejärjestikuse vooga (tingimuslause, for-tsükkel, while-tsükkel). Modulaarsuse (funktsioonid, parameetrid, muutuja skoop) õppimiseks peab õppija olema jõudnud järgmisele tasandile, mõtlema abstraktsemalt. Konstrukt "Lihtsamad rakendused" (kasutajaliides/graafika ning töö failidega) sisaldab endas moodulite kasutamist (*turtle*, *tkinter, urllib* jne) ning failidega suhtlemist (alguses failist lugemine, siis uude faili kirjutamine, edasi olemasolevasse faili juurde kirjutamine).

Analüüsimiseks vaatasin läbi kursuse abistava õpiku, kirjutasin välja võtmesõnad, lisades neile õpiku teema ja konstrukti. Et olla kindel, et miski ei jäänud märkamata, lisasin analüüsitava tabelisse (Lisa 4) kõik mõisted õpiku mõistete peatükist. Edasi märkisin iga mõiste kohta, mis peatükis seda õpetatakse (varem esitatud joonisel 2 on näha, et õpiku teemad lähevad kokku nädalas õpitava teemaga). Lõplik nimekiri on pikk, kõike ei saa mõistliku ajaga lahendatava testi abil kontrollida. Võttes arvesse, et töö failidega on väga suur teema ning on kontrollitav projekti loomise raames, ei ole otstarbeks seda testis uurida. Samuti jäävad testi skoobist välja kasutajaliides ning graafika, kuna need on abistavad teemad, mis aitavad õpilastel sisust paremini aru saada ning on väga keelespetsiifilised.

Lõpliku konstruktide nimekirja koostamisel toetusin varem tehtud uuringutele ning nendes väljatoodud teemadele (Higgins jt, 2019; Tew, 2010). Vastendust (ingl *mapping*) analüüsisin tabelis, mille lõpptulemus on toodud lisas 4. TTL on Pythoni keelespetsiifiline, seega osad teemad sõltuvad just keele spetsiifikast: nt muutujate defineerimine ja andmetüübid.

Analüüsi tulemusena moodustasin järgmist nimekirja konstruktidest, mida testis kontrollida:

1	Põhialused	I nädal	Põhialused (muutuja defineerimine, andmetüübid, matemaatilised avaldised). Sisend/väljund
2	Loogilised avaldised	II nädal	Loogilised avaldised
3	Tingimuslause	II nädal	Tingimuslause
4	Eelkontrolliga tsükkel	III nädal	Eelkontrolliga tsükkel (while-tsükkel)
5	Sõne	IV nädal	Sõne
6	For-tsükkel	V nädal	For-tsükkel
7	Järjend	V nädal	Järjend
8	Funktsioon	VI nädal	Funktsioon. Parameetrid
		VI nädal	Funktsioon. Tagastatav väärtus
		VI nädal	Muutuja skoop



Joonis 4. TTL-kursusel omandatavad oskused (alamteemade tasemel)

2.3. Testi spetsifikatsioon ning küsimuste planeerimine

"Testi spetsifikatsioon on instrumendi põhjalik kirjeldamine, milles määratakse, mitu protsendi küsimustest iga konstrukti kohta testis on, küsimuste formaat ja hindamise protseduurid." ("The Standards for Educational and Psychological Testing"[5], 1999 viidatud Tew, 2010: 20 kaudu)

Venemaal on väljatöötatud testide spetsifikatsioon, mis koosneb kaheteistkümnest punktist (Tchelyshkova, 2002), nende hulgas:

1. ülesannete tüüpide arv, ülesannete koguarv;
2. iga ülesande kaal (vn *вес*);
3. testi lahendamise aeg (soovitatav);
4. ülesannete arv teema kohta (teema küsimuste protsent testis);
5. ülesannete järjekord testis.

Toetudes Tew' töös kirjeldatud teooriale ning leides lisatuge ka Venemaa allikatest (Tchelyshkova, 2002), otsustasin luua testi, milles on kaetud kaheksa konstrukti ning iga konstrukti kohta on kolm küsimust. Testi formaat on valikvastustega test. Iga õigesti vastatud küsimus annab maksimaalselt ühe punkti.

Testi formaadiks sai valitud valikvastusega test, kuna sellel formaadil on suured eelised − lihtne testi administreerimine, tulemuste kiire hindamine (Haladyna, 2004; Lukhele jt, 1994, viidatud Tew, 2010 kaudu; Tchelyshkova, 2002). Korrektselt loodud testi abil võib kontrollida kogu kursuse sisu, mis annab parema ülevaade (vajadusel ka hinne) õppijate õpitud oskuste/teadmiste ulatusest. Valikvastustega testide põhiline kriitika seisneb selles, et õppija võib juhuslikult arvata õige vastuse. Selle vastu võitlemiseks kasutatakse erinevaid strateegiad. Nt Tchelyshkova (2002) kirjutab, et sellisteks on:

1. õppijatele antud juhistes soovitatakse pigem jätta ülesanne vahele, kui proovida ära arvata;
2. korrigeeritakse nõrgemate õppijate tulemusi;
3. tulemuste arvutamisel kasutatakse valemit, milles on arvestatud vastuse võimaliku äraarvamisega.

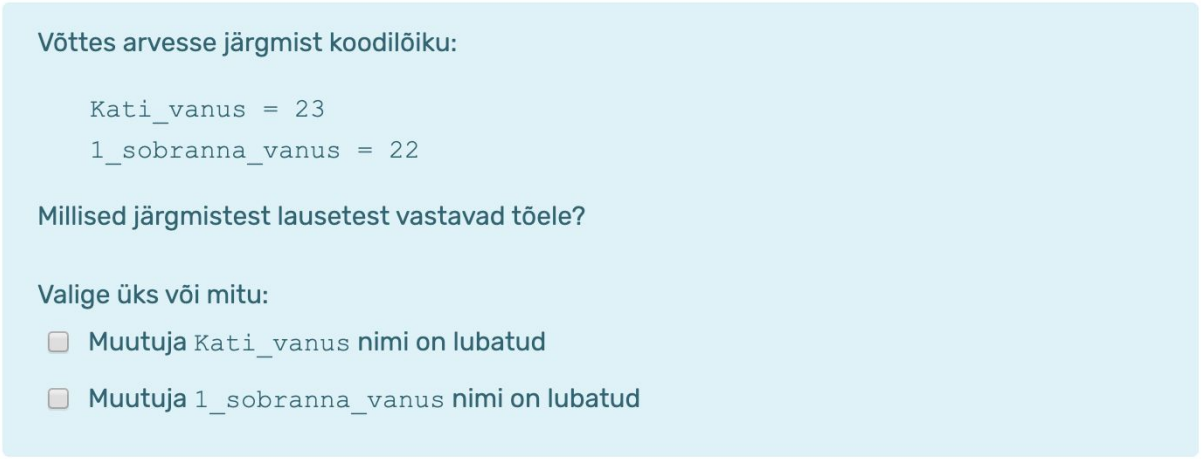
[5] https://www.apa.org/science/programs/testing/standards
American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (1999). Standards for educational and psychological testing. Washington, DC: American Educational Research Association.

Tew toetus oma testi loomisel (Miller, Linn & Gronlund, 2009a, viidatud Tew, 2010 kaudu) järgmistele printsiipidele:

1. igas küsimuses peab olema nii palju infot kui vaja, kuid peab vältima ebaolulist materjali;
2. igal küsimusel on ainult üks õige vastus;
3. kõik peibutusvastused peavad olema usutavad;
4. vastuse pikkus ei tohi vihjata korrektsele vastusele;
5. õige vastus peab ilmuma testi jooksul erinevates kohtades sama arv kordi (juhuslikus järjekorras);
6. mitte kasutada (ilma tungiva vajaduseta) vastuse variantideks "Mitte üks ei sobi" või "Kõik sobivad".

Iga defineeritud teema kohta lõin toetudes ülalnimetatud printsiipidele küsimused, mis aitavad aru saada, mis tasemel teema omandatud on. Selle jaoks kasutasin Tew' (2010) testi kolme tüüpi küsimusi (iga tüüpi kohta on toodud näide magistritöö tulemusena valminud testist):

1. Küsimused mõiste (definitsiooni) tasemel − küsimused mõistete kohta, uurivad õppija üldist arusaamist mõistest/konstruktist. Näide: Joonis 5.
2. Küsimused koodi järgimise tasemel − uurivad õppija oskust ennustada koodi käivitamisel tekkinud tulemust (nt muutuja väärtus tsükli töö lõpetamisel). Näide: Joonis 6.
3. Küsimused koodi lõpetamise tasemel − uurivad, kas õppija on suuteline ise koodi kirjutama. Need on lünkvastustega küsimused. Näide: Joonis 7.



Joonis 5. Küsimus definitsiooni tasemel (näide)

Võttes arvesse järgmist koodilõiku:

```
summa = 0
for i in range(1, 5, 2):
    print(i)
    summa += i
print(summa)
```

Mis arvud väljastab programm ekraanile koodi käivitamise tulemusena?

Valige üks:

- 1, 3 ja 4
- 1, 1, 3 ja 4
- 1, 3, 5 ja 9
- 1, 1, 3, 4, 5 ja 9

Joonis 6. Küsimus koodi järgimise tasemel (näide)

Võttes arvesse järgmist koodilõiku:

```
d = 2

if d < 3:

    print("d ei sobi")

_____

    print("d ei sobi üldse")

else:

    print("võib-olla d sobib")
```

Millised alljärgnevatest koodilõikudest tagavad, et programm väljastaks tulemusena ainult "d ei sobi"?

Valige üks või mitu:

- [x] elif d < 12: ✔
- [ ] if d < 12:

Joonis 7. Küsimus koodi lõpetamise tasemel (näide)

2.4. Testi küsimuste loomise protsess

Testi loomisega sain alustada siis, kui oli selge testi skoop (konstruktide nimekiri valmis, kursusel õpetavate mõisted kategoriseeritud) ning spetsifikatsioon.

Küsimuste näidiseks kasutasin Tew' (Lisa 1) ja Parkeri jt (Lisa 3) küsimusi, lisades on toodud kõik avaldatud küsimused. Tew avaldas üksnes küsimused, mis ei läbinud valiidsuse testi, seega pidin need kasutama ettevaatlikult. Lisaks toetusin küsimustele "Programmeerimise" õpikus (Tõnisson jt, 2019). Minul puudus juurdepääs kursuse käigus lahendatavatele ülesannetele. Kõik loodud küsimused on originaalsed, nende loomist inspireerisid Tew, Parker jt, Tõnisson jt oma töödega.

Planeerides vastuseid küsimustele lähtusin Tew' töös (Miller, Linn & Gronlund, 2009a, viidatud Tew, 2010 kaudu) valitud punktidele:

1. iga küsimus sisaldab ainult ühe õige vastuse;
2. kõik peibutusvastused peavad olema usutavad;
3. vastuse pikkus ei tohi vihjata korrektsele vastusele;
4. õige vastus peab ilmuma testi jooksul erinevates kohtades sama arv kordi (juhuslikus järjekorras);
5. mitte kasutada (ilma tungiva vajaduseta) vastuse variantideks "Mitte üks ei sobi" või "Kõik sobivad".

Lähtudest soovitusest, et õigete vastuste tähised (A, B, C, D) esinevad testis protsentuaalselt võrdselt ning igas küsimuses on ainult üks õige vastus, lõin küsimusi eeldusega, et igal küsimusel on 4 vastust. Kokku on 4 korda 3 (küsimuste tüübid) korda 8 (konstruktide arv) vastuste kohta, ehk jälgisin, et iga õige vastuse tähis esineb testis 24 korda. Valisin testi variandi, kus igal küsimusel on 1 õige vastusevariant ning kolm valet vastusevarianti. Tew' testis on igal küsimusel 4 peibutusvastust, ka Tchelyshkova (2002) kirjutab, et küsimuse kohta peab olema vähemalt 4 peibutusvastust. Uuemad uuringud (Gierl jt, 2017; Kilgour jt, 2015) näitavad, et peibutusvastuste kunstlik lisamine ei ole alati testile kasuks. Vastuseid, mis ei ärata usaldust, ei valita. Sisuliselt jääb ikkagi väiksem vastuste variantide hulk. Lisaks on selline vastuste lisamine halb, kuna testi lahendamise aeg pikeneb ilma kasu toomata. Peibutusvastuste rohkus on takistus hea testi loomisel, kuna selliste vastuste väljamõtlemine võtab aega ning pole alati võimalik. Selles sain veenduda kohe esimese küsimuse loomisel: luua küsimuse definitsiooni tasemel põhialuste konstrukti kohta, millel on 3 peibutusvastust osutus tõeliseks väljakutseks.

Küsimuste planeerimisel lähtusin lisaks lausest “igas küsimuses peab olema nii palju infot kui vaja, kuid peab vältima ebaolulist materjali” (Miller, Linn & Gronlund, 2009a, viidatud Tew, 2010 kaudu). Konstrukti kohta küsimust luues jälgisin, et selles ei kontrollitaks teisi konstrukte: nt “põhialuste” konstrukti küsimuses ei tohi kontrollida “sõne” spetsiifilisi teadmisi. Raske oli nt luua küsimust “eelkontrolliga tsükli (while-tsükkel)” konstrukti kohta “loogiliste avaldiste” spetsiifikat kasutamata.

Testi esialgse versiooni valmimisel esitasin selle ekspertidele (kelleks olid juhendajad) hindamiseks. Esialgse versiooni küsimuste näited on toodud kolmel joonisel (õige vastus on rohelise värviga esile tõstetud):

1. Joonis 8: Küsimus definitsiooni tasemel (konstrukt “Põhialused”)
2. Joonis 9: Küsimus koodi järgimise tasemel (konstrukt “Loogilised avaldised”)
3. Joonis 10: Küsimus koodi lõpetamise tasemel (konstrukt “Põhialused”)

Võttes arvesse järgmist koodilõiku:

```
1. print("Palun sisestage oma nimi: ")
2. nimi = input()
3. pass = input("Palun sisestage passi number: ")
4. print(nimi + ", Teie passi number on " + pass + ".")
```

Millised järgmistest lausetest lausetest vastavad tõele?

I Muutuja `nimi` on korrektselt defineeritud
II Muutuja `pass` on korrektselt defineeritud

Vastused:

A ainult I on tõene
B ainult II on tõene
C I ja II on tõene
D I ja II ei ole tõene

Joonis 8. Küsimus definitsiooni tasemel (konstrukt “Põhialused”)

Võttes arvesse järgmist koodilõiku:

```
1. sporti_teinud = True
2. kuusk_ehitud = False
3. tuba_korras = False
4. kodutööd_tehtud = True
5. raamat_loetud = False
6. hea_laps = (tuba_korras or kuusk_ehitud) and kodutööd_tehtud
7. jõuluvana_toob_kingitusi = hea_laps and (sporti_teinud or
   raamat_loetud)
```

Millised järgmistest lausetest vastavad tõele?

I Laps on olnud hea `hea_laps == True`
II Jõuluvana toob kingitusi `jõuluvana_toob_kingitusi == True`

Vastused:

A ainult I on korrektne
B ainult II on korrektne
C I ja II on korrektsed
D I ja II ei ole korrektsed

Joonis 9. Küsimus koodi järgimise tasemel (konstrukt “Loogilised avaldised”)

Trigonomeetriast teame, et mis tahes nurk on esitatav kujul

$$x = \alpha + n \cdot 360^\circ,\ kus\ 0^\circ \leq \alpha \leq 360^\circ\ ja\ n \in \mathbb{Z}$$

Programm küsib kasutajalt nurga ning esitab seejärel nurga üleval nimetatud kujul.

```
nurk = int(input ("Palun sisestage nurk: "))
print("nurk = ",_____)
```

Milline koodilõik alljärgnevatest valikutest ei sobi antud ülesande lahenduseks:

A `str(nurk % 360), " + ", str (nurk // 360), " * 360"`
B `str(nurk // 360), " + ", str (nurk % 360), " * 360"`
C `str(nurk - nurk // 360 * 360), " + ", str (nurk // 360), " * 360"`
D `str(nurk / 360 * 360 - nurk // 360 * 360), " + ", str (nurk // 360), " * 360"`

Joonis 10. Küsimus koodi lõpetamise tasemel (konstrukt “Põhialused”)

Ekspertiisi tulemusena vajas test suuri muudatusi:

1. Arutelu jooksul sai otsustatud, et konstrukt “Loogilised avaldised” ei sobi antud kursuse puhul eraldiseisvana, pigem peab olema kontrollitud koos konstruktiga “tingimislause”.
2. Küsimused on liiga konteksti spetsiifilised, liiga eluline kontekst võib segada testi lahendamist, kuna õppija tähelepanu võib olla suunatud konkreetsele kontekstile, nt trigonomeetria kursusele (Joonis 10), seega kõik ülesanded tuleb üle vaadata, muutes võimalusel abstraktsemaks.
3. Otsustati loobuda küsimuste formaadist: 1 õige vastus, 3 vale. Uueks formaadiks on valikvastustega test, kus mõne küsimuse puhul õige vastus moodustub mitmest valikvastusest. Osade küsimuste puhul on loomulikum lasta vastajal valikvastuste hulgast õiged valida, kui püüda moodustada keerulisemad vastusevariandid, millest täpselt üks on õige. Samuti, kui vastajal ei ole teada, mitmest valikvastustest õige vastus koosneb, peab ta süvenenumalt mõtlema.
4. Esines ka teisi muresid: keelelised, vormistamine jne.

Tegin testi ümber toetudes ekspertide arvamusele. “Loogiliste avaldiste” konstrukti eemaldamise tagajärjel muutus kohe õigete valikute sagedus: A − 6, B − 5, C − 5, D − 5. Õigete valikute sagedus muutus veelkord, kui muutsin küsimuste tüüpi − mitu võimaliku vastuse küsimuse kohta. Tulemusena esinevad õiged vastused sarnaseid arvu kordi (keskmist ning kogu arvu ei avalikusta testi valiidsuse säilitamise nimel).

Lõplik test koosneb 21 küsimusest: iga konstrukti kohta on kolm küsimust.

2.5. Kokkuvõte

Käesolevas peatükis kirjeldasin Tew’ (2010), Parkeri jt (2016), Tchelyshkova (2002) tööde põhjal TTL-kursuse jaoks loodud testi spetsifikatsiooni. Kirjutasin testi skoobi määramisest, mille viimane muutmine toimus ekspertide arvamuse põhjal, kui esialgne testi versioon oli juba valmis. Kirjutasin küsimuste formaadist, mis samuti ei vastanud täielikult Tew’ töös esitatud formaadile: esialgu tegin muudatused küsimuste arvus, toetusin selles küsimuses uuematele uuringutele (Gierl, Bulut, Guo & Zhang, 2017; Kilgour & Tayyaba, 2015).

Olulisim küsimus, millele tuli testi loomisel vastata, oli, kuidas subjektiivne arvamus mõjutab lõpptulemust, kas valitud mõiste/meetod sobivad testi, sest mulle tundub, et nii on õige või on sellel ka objektiivne põhjendus. Võttes arvesse, et lõin testi peamiselt õpiku alusel, oli mul tihtipeale raske võtta vastu otsust: kas ühele või teisele konstruktile ja/või teemale pühendati tundide (juhul, kui need olid), koduste ülesannete puhul sama palju aega (ruumi), kui õpikus. Lisaks mängis siin rolli isiklik kogemus: mis enda jaoks tundub olema raske, huvitav, ei pruugi olla nii kursuse loojate ja/või õppijate jaoks.

3. Testi valideerimine

Käesolevas peatükis kirjutan loodud testi valideerimise protsessist. Esialgu annan lühiülevaate testi valideerimise teooriast ning sellest, kuidas on plaanis teooriast lähtuvalt tõestada loodava testi valiidsust. Seejärel kirjutan läbiviidud intervjuudest ning testi piloteerimisest.

3.1. Testi valiidsusest

"Testi valiidsus näitab, kui hästi test mõõdab seda, mille mõõtmiseks ta on ette nähtud" (Mikk, 2002). "Valiidsus väljendab mõõtmise tõelisust, õigsust ja tegelikkusele vastavust" (Schiavetti & Metz, 2002 viidatud (Tulviste, 2013) kaudu). "Valiidsus näitab seda, kas mõõdetakse ning uuritakse seda, mida uurida ja mõõta tahetakse" (Elmes jt, 2008 viidatud (Tulviste, 2013) kaudu).

Mikk toob oma töös välja, et on olemas neli testi valiidsuse liiki: sisuvaliidsus (ingl *content related evidence*, vn *репрезентативность содержательности теста*), ennustav valiidsus (ingl *criterion-related validity*[6]), võrdeline valiidsus (ingl *correlate validity*), faktorvaliidsus. Faktorvaliidsus on (Tulviste (2013) järgi konstruktivaliidsuse üks variantidest; ingl *construct related evidence*).

Mikk (2002) ning Tchelyshkova (2002) kirjutavad, et nendest esimene valiidsuse liik, sisuvaliidsus, on kõige olulisem. Sisuvaliidsuse puhul tuleb teha kindlaks, kas testis kontrollitavad konstruktid kirjeldavad kursusel õpetavat, tuleb teha kindlaks, et testis ei kontrollita seda, mida pole kursusel õpitud. Sisuvaliidsust tõestatakse korrektse sisu analüüsi ja planeerimisega, mida kontrollib ekspertide rühm. Käesolevat testi kontrollis ekspertide rühm, millest on kirjutatud eelmises peatükis.

Ennustav valiidsus näitab, kuidas saab antud testi tulemuste põhjal ennustada mingit tulemust. Näiteks, kui viia läbi eeltestimist, kas eeltesti tulemus näitab, kui hästi oskab õppija ainet selle lõpuks. Sellist eesmärki loodaval testil ei olnud, seega seda ei kontrollitud.

Võrdelise valiidsuse tõestamist viiakse läbi võrdlemise teel. Kui on olemas samu teadmisi/oskusi kontrolliv valideeritud test, siis viiakse läbi uuring, milles samad inimesed lahendavad esialgu uue testi ning hiljem eelnevalt valideeritud testi. Teine võimalus on võrrelda uue testi tulemust õppijate eksami tulemusega (Tchelyshkova, 2002). Sarnast lähenemist kasutas oma töös ka Tew. Kui nende tulemuste vahel leitakse oluline seos, siis

[6] Mart Murdvee "Loeng 1. Psühholoogia − sissejuhatus" https://www.ttu.ee/public/m/mart-murdvee/Psuhholoogia_ja_loogika/Loeng_1._Psuhholoogia_-_sissejuhatus.pdf (12.05.2020)

tunnistatakse uus test valiidseks. Meie puhul puudub programmeerimisaluste teadmisi/oskusi kontrolliv valideeritud (eestikeelne) test. Testi tulemusi võib võrrelda TTL-kursuse hindamise tulemustega.

Faktorvaliidsus on kõige täpsem. "Selle leidmiseks on vajalik mitu sama omadust mõõtvat testi, millega määratakse mõõdetava omaduse tase katseisikutel ja tehakse tulemuste faktoranalüüs." (Mikk, 2002) Faktoranalüüs eeldab, et uuritavate objektide arv on vähemalt 300, kuigi kirjeldava meetodina võib kasutada ka väiksemate valimite korral (Niglas, 2013). Kuna testi piloteerimisel ei saanud piisavalt palju (300 objekti) vastuseid, siis jäi faktoranalüüs (ka kirjeldava meetodina) antud magistritöö raames teostamata.

Testi valiidsuse tõestamiseks viisin läbi testi skoobi põhjaliku analüüsi, ekspertide hinnangu läbisid testi esimene ning teine variant, teises variandis said parandatud ekspertide nimetatud kitsaskohad. Edasi viisin läbi kaks valjusti mõtlemise (ingl *think aloud*) intervjuud. Intervjuude abil avastatud vead said parandatud. Järgmise etapina toimus testi piloteerimine. TTL-kursusel õppijatele oli pakutud võimalus lahendada uut testi. Motiveeriva tegurina pakuti võtta seda kui testi, mis aitab enne arvestustööd korrata olulisi asju. Õppijad ei saanud testi lahendamise eest tasu (k.a ei saanud lisapunkte arvestustöö jaoks). Lõpuks viisin läbi statistilise analüüsi, uurides piloteeritud testi tulemusi ning võrdlesin arvestustöö (test ja arvutis lahendatud programmeerimisülesanne) tulemusi uue testi tulemustega.

3.2. Intervjuud

Testi esialgseks kontrolliks viisin läbi kaks intervjuud, mille käigus palusin lahendada testi valjusti mõeldes. Enne intervjuud tutvustasin intervjuu eesmärki. Tutvustasin andmekaitse nõudeid: nimeliselt intervjueeritavaid töös ega mujal ei nimetata. Sooritamise tulemusi ei avalikustata. Palusin nõusolekut intervjuu salvestamiseks. Intervjuu toimus Viberi ning Facebooki kõnede abil (seoses eriolukorraga riigis). Teise intervjueeritavaga rääkisime enne ja pärast testi lahendamist sellest, mida ootan ning kuidas läks, testi lahendas iseseisvalt, vastates kirjalikele küsimustele suuliselt. Mõlemad intervjueeritavad salvestasid oma mõtteid ise ning siis saatsid mulle faili.

Intervjuu eesmärk:

1. esimese taseme kontrollina leida keelelisi vigu, liiga raskeid, lihtsaid, halvasti sõnastatud küsimusi jne;
2. saada esmane vastus küsimusele, kas selline test aitab aru saada, mida õppija oskab või mitte.

Võttes arvesse varasemaid uuringuid ning töö eesmärki, püstitasin küsitluse läbiviimiseks kaks lahtist küsimust:

1. Kuidas põhjendad oma lahendust? Mõtle valjusti! Kirjelda, mis samme sa pead ette võtma, et leida vastust.
2. Kuivõrd aitavad antud test ning selle tulemused aru saada, mida juba tead, mida mitte?

Lisaks palusin vastata iga küsimuse kohta:

1. Kuivõrd selge on küsimuste sõnastus? (Segane/Typo/Selge)
2. Kuivõrd rasked küsimused on? (Raske/Paras/Lihtne)
3. Hinnang oma vastusele. (Olen vastuses kindel, oskan põhjendada/Tundub olema õige vastus, oskan põhjendada/Tundub olema õige, ei oska põhjendada/Ei oska/Juhuslik vastus)

Esimene intervjueeritav oli 8. klassi õpilane, kes on mitu aastat käinud robootikaringis (õppevahendiks on EV3-robotid, programmeerimine plokk-keeles). On programmeerimise alustega tuttav, kuid pole kunagi programmeerinud tekstil baseeruvas keeles. Õpilane õpib vene õppekeelega koolis. Aitasin teda teksti arusaamisel. Teine intervjueeritav oli üliõpilane, kes osales TTL-kursusel.

Intervjuu pikkus (ehk testi lahendamine valjusti mõtlemise meetodil) võttis kooliõpilase puhul 36 minutit ning üliõpilase puhul 23 minutit. Intervjuu lõpus andis kumbki oma hinnangu: esimene ütles, et test oli raske, sest seda keelt ei oska ning pidi palju ära arvama; teine ütles, et test annab head ülevaadet sellest, mida tuleb enne arvestustööd korrata.

Intervjueeritavate programmeerimisoskused ning teadmised olid väga erineval tasemel, mis oli hästi jälgitav tänu valjusti lahendamisele. Esimene intervjueeritav kasutas palju loogikat, mida võiks üks või teine mõiste tähendada, kuidas kood töötab. Mõisted polnud selged (muutuja, tsükkel), kui tuletada meelde või tuua analoogi, siis sai aru (siinkohal pidi olema ettevaatlik, sest pidin aitama ka tõlkimisega). Oma vastustes vaatas ka eelmisi küsimusi ning paar korda soovis parandada vastuse eelmisele küsimusele (vastavalt 8–7 ning 11–10). Vastates funktsiooni kohta mõtles ta graafikule (just koolis õpitud teema), kuna programmeerimismõistena oli funktsioon talle tundmatu. Teise intervjueeritava puhul olid peaaegu kõik vastused õiged ning valjusti mõtlemise käigus erilisi mõttekäike ei esinenud. Kahes kohas parandas intervjueeritav enne lõpliku otsuse oma vastuse õigeks. Kasutas vastamisel ka välistamistaktikat. Üldiselt oli test esimese intervjueeritava jaoks pigem eeltest ning teise jaoks lõpptest.

Intervjuude käigus leiti erinevaid probleemkohti:

1. Leiti keeleline viga.

Testi kolmes küsimuses esines vastusevariandina lause “Programm väljastab veateade”. Intervjueeritav märkas seda viga ühe küsimuse vastustes, kus sama viga oli kahes vastusevariandis. Küsimuste-vastuste järelanalüüsimisel leidsin ka teisi vigaseid kohti. Uus tekst: “Programm väljastab veateate”. Mõjutatud küsimused: 7 (teema 2, esimene küsimus), 13 (teema 4, esimene küsimus), 19 (teema 7, esimene küsimus).

2. Mitmes kohas oli koodis taane lisamata (küsimused 15, 21).
3. Kahes kohas oli vastustes kasutatud muutuja või tagastatava väärtuse jaoks vale kirjastiili: oodati, et koodi osa on Courier New kirjastiilis.
4. Üheksandas küsimuses väljastas programm lause “vana a:”. Intervjueeritav pakkus, et keeleliselt on korrektne alustada suure tähega: “Vana a:”.

Intervjuu tulemusena saadud kitsaskohtade nimekirja abil parandasin testi veel kord ning andsin testi ekspertidele viimaseks kommentaariks. Eksperdid osutasid tähelepanu, et kahes küsimuses on nende hinnangul üleliigne vastus (küsimused 1, 6). Parandasin ning test oli valmis piloteerimiseks.

3.3. Testi piloteerimine

Testi piloteerimiseks pakuti seda pealkirjaga “Test põhiliste asjade kohta” programmeerimiskursuse TTL õppijatele. Testi lahendamine ei olnud kohustuslik, selle lahendamise eest ei saanud tasu (k.a lisapunkte). Motiveerivaks teguriks oli lisavõimalus ettevalmistuda arvestustööks. Käesolevas alampeatükis kasutan sõna “test” tähenduses “TTL-kursuse jaoks loodud test” ning “arvestustest” tähenduses “TTL-kursuse raames läbiviidud arvestustöö test”. Testi lahendamine viidi läbi keskkonnas Moodle. Testi võis lahendada piiramatu arv kordi. Testi piloteerimise tulemusi võrdlesin seda testi lahendatud õppijate TTL-kursuse raames sooritatud arvestustesti ning programmeerimisülesande lahendamise tulemustega. Osad õppijad sooritasid arvestustesti ühel, teine osa teisel päeval. Erinevatel päevadel lahendatud ülesanded erinesid vaid vähesel määral, mis võimaldas nende uurimist koos läbi viia.

3.3.1. Testi ning TTL-kursuse arvestustöö tulemuste võrdlemine

Kokku alustas testi lahendamist 21 õppijat 52 õppijast, kes sooritasid arvestustöö nimetatud päevadel. Seitse õppijat lahendasid testi mitu korda. Kaks õppijat alustasid testi sooritamist, kuid ei lõpetanud, nendest üks vastas ühele küsimusele, teine ei vastanud mitte ühtegi küsimusele. Need kaks jätsin analüüsist välja. Edaspidi uurin ülejäänud 19 õppija tulemusi.

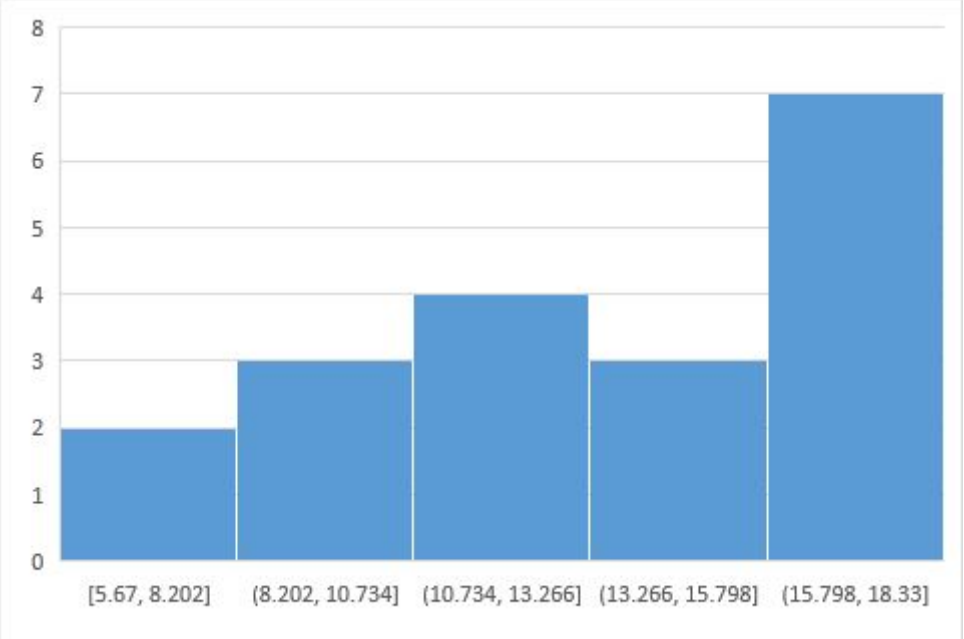
Mitmed õppijad lahendasid testi mitu korda: neli õppijat lahendasid testi 2 korda, kolm õppijat lahendasid testi 3 korda. Analüüsimiseks võtsin eraldi kaks hulka: testi esimene

lahendamise katse ning testi viimane lahendamise katse (samal ajal on need vastavalt testi halvima tulemusega sooritamine ning testi parima tulemusega sooritamine).

Õppijate testi sooritamise esimesed ning viimased tulemused on toodud tabelis 1. Enamik õppijatest saavutasid testi lahendamisel rohkem kui 50% nii esimese, kui ka viimase sooritamise puhul. Kuna kõik ei teinud katset mitu korda, siis võtame edasises vaatluse alla just esimesed tulemused. Vaadates esimese (Joonis 11) lahendamise katsete tulemuste jaotust on näha, et katse tulemused ei moodusta normaaljaotust. Tchelyshkova (2002) toob välja, et normidele suunatud testi puhul peavad tulemused moodustama normaaljaotuse. Sellel võib oma mitu erinevat põhjust: halvasti koostatud test, mitterepresantiivne valim vms.

Tabel 1. Testi sooritamise esimesed ning viimased tulemused

Õppija	o1	o10	o12	o13	o14	o15	o16	o17	o18	o19	o2	o20	o21	o3	o4	o6	o7	o8	o9
Esimene hinne	10.17	12.5	14.5	14.67	16.83	17.67	13.17	18.33	12.83	17.67	5.67	18	15.83	10.33	13.67	11.83	9.67	6.17	15.83
Viimane hinne	10.17	12.5	21	20.17	16.83	20	19.17	18.33	12.83	17.67	5.67	18	15.83	10.33	21	11.83	18.5	6.17	21

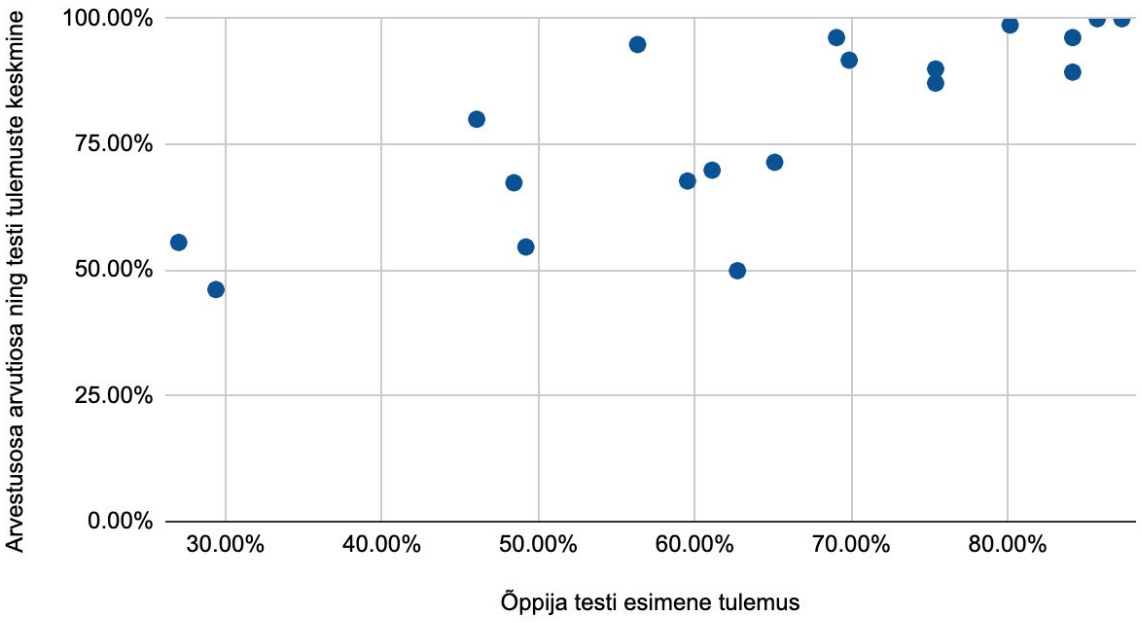


Joonis 11. Õppijate esimesed tulemused histogrammina

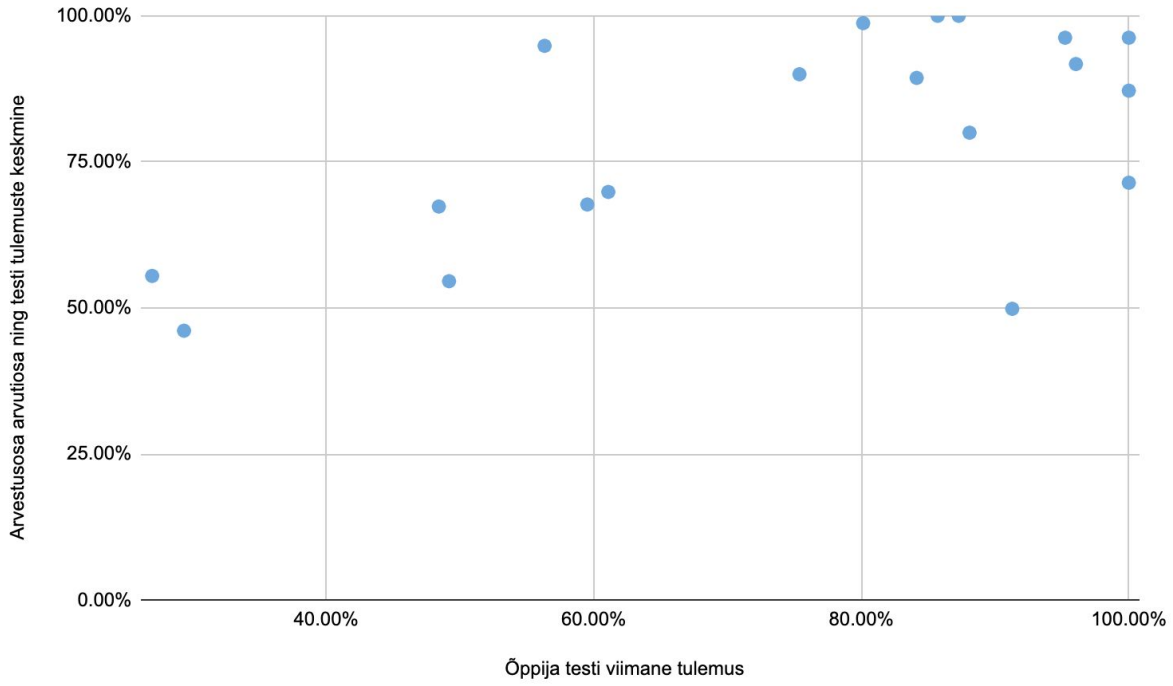
Käesoleva pilooteerimise tulemused annavad võimaluse võrrelda testi ning TTL-kursuse arvestustöö kokkuvõtvate tulemuste (arvestustest ning programmeerimisülesande lahendamine) omavahelisi seoseid. Selleks vaatlesin eraldi iga õppija TTL-kursuse arvestustöö kokkuvõtvaid tulemusi ja testi esmaseid tulemusi (Joonis 12). Huvi pärast vaatlesin analoogiliselt õppija TTL-kursuse arvestustöö kokkuvõtvaid tulemusi ja testi viimaseid tulemusi (Joonis 13).

Korrelatsioonivälja l on näha, et mõlemal juhul on positiivne vastastikune seos olemas. Kusjuures seos on tugevam selle võrdluse puhul, kui õppija kursuse lõpptulemuse võrreldakse testi esmase tulemusega.

Kasutasin korrelatsioonikoefitsiendi väljaarvutamiseks Google Sheet funktsiooni CORREL (lineaarne ehk Pearsoni korrelatsioonikordaja). Korrelatsioonikoefitsient on esimese puhul 0,79 (0,7927632314), ehk tugeva seosega ning viimase puhul on korrelatsioonikoefitsient 0,62 (0,6225637395), ehk keskmise tugevusega (Laanpere, Niglas, Osula & Pata, 2013). Kordaja on positiivne, ehk tegu on positiivse seosega: mida parem on testi tulemus, seda parem on arvestustöö tulemus ning vastupidi, mida tagasihoidlikum on testi tulemus, seda halvem on arvestustöö tulemus.



Joonis 12. Testi esimeste tulemuste ning arvestuse tulemuste korrelatsiooniväli



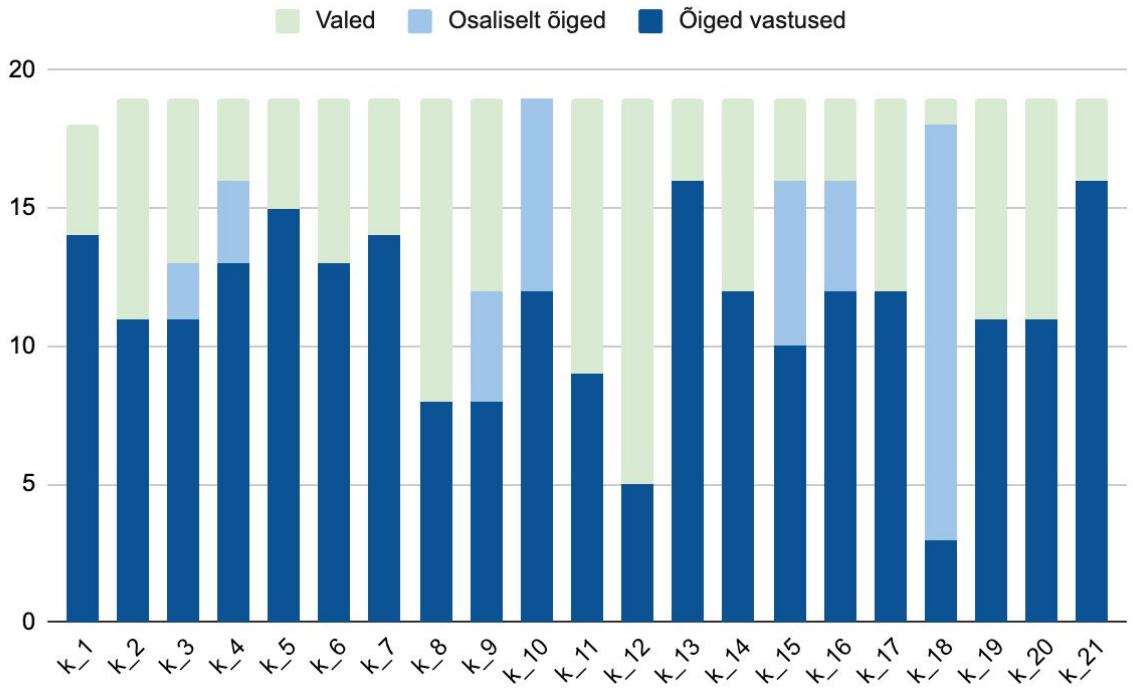
Joonis 13. Testi viimaste tulemuste ning arvestuse tulemuste korrelatsiooniväli

3.3.2. Testi küsimuste analüüs

Tabelis 2 on esitatud testi esimese lahenduse tulemused, eraldi on välja toodud õigete, osaliselt õigete ning valede vastuste koguarvud, graafiliselt on need andmed esitatud joonisel 14. Esimese küsimuse puhul on vastuste koguarv 18, üks õppija on jätnud küsimusele vastamata. Kõige vähem tuli maksimaalseid tulemusi 18. küsimuse puhul (3 korda), kõige rohkem tuli maksimumpunkte 13. ja 21. küsimuste puhul (16 korda). Tabeli teises reas on osaliselt õigete vastuste koguarv (iga küsimuse kohta). Kõikide küsimuste puhul ei saanud olla osaliselt õiget vastust, sest mõned küsimused on ühe õige vastusega. Küsimuste 10 ja 18 puhul tuli kõige vähem täiesti valesid vastuseid (vastavalt 0 ja 1). Küsimuse 18 puhul on väga suur erinevus õigete ja osaliselt õigete vastuste vahel (3 inimest said maksimumpunkte ning 15 said osa punkte). Küsimuste 8 ja 11 puhul tuli intervjuude käigus, et need aitavad parandada eelmise küsimuse vastuse (vastavalt 7 ja 10). Täpsemaid analüüse on vaja, et leida kinnitust või ümberlükkamist küsimusele, kas küsimuste 8 ja 11 abil saab parandada küsimuste 7 ja 10 vastuseid.

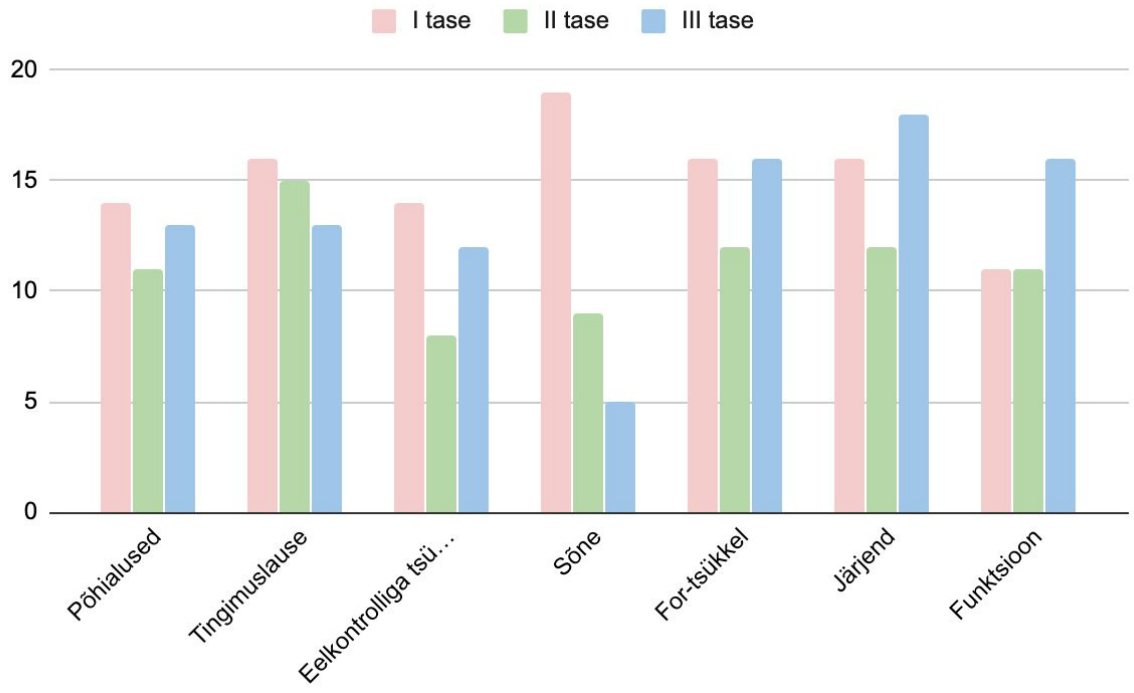
Tabel 2. Testi esialgsed tulemused

	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_10	k_11	k_12	k_13	k_14	k_15	k_16	k_17	k_18	k_19	k_20	k_21
Õiged vastused	14	11	11	13	15	13	14	8	8	12	9	5	16	12	10	12	12	3	11	11	16
Osaliselt õiged	0	0	2	3	0	0	0	0	4	7	0	0	0	0	6	4	0	15	0	0	0
Valed	4	8	6	3	4	6	5	11	7	0	10	14	3	7	3	3	7	1	8	8	3

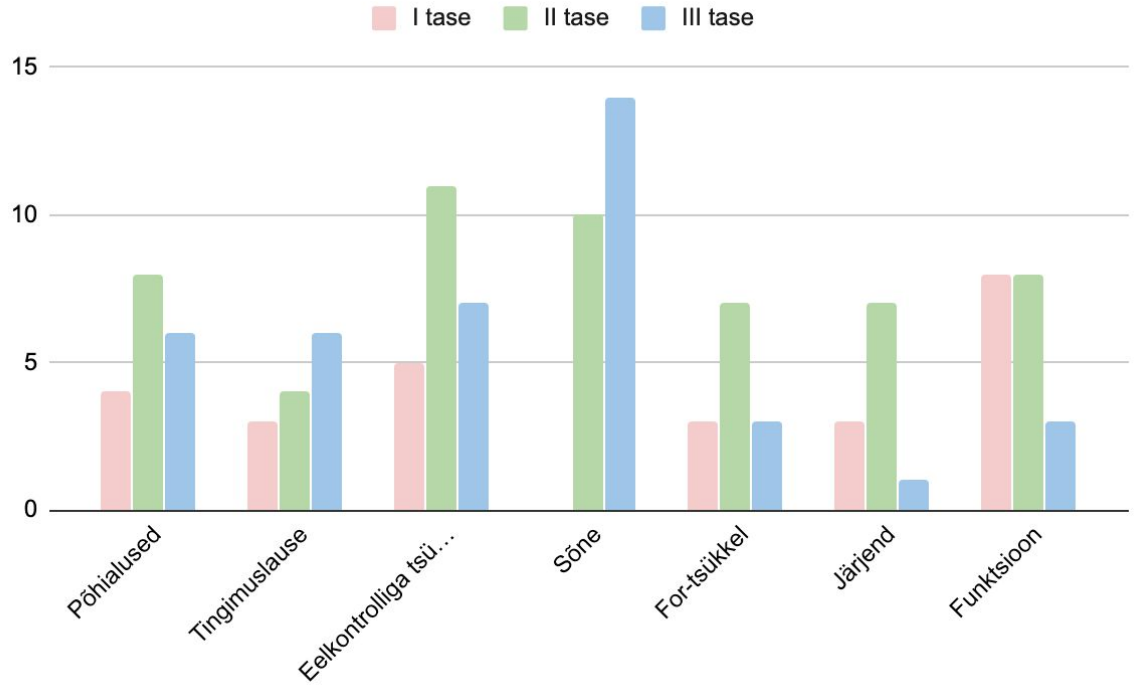


Joonis 14. Õigete/osaliselt õigete/valede vastuste arv − testi esimese lahendamise tulemused

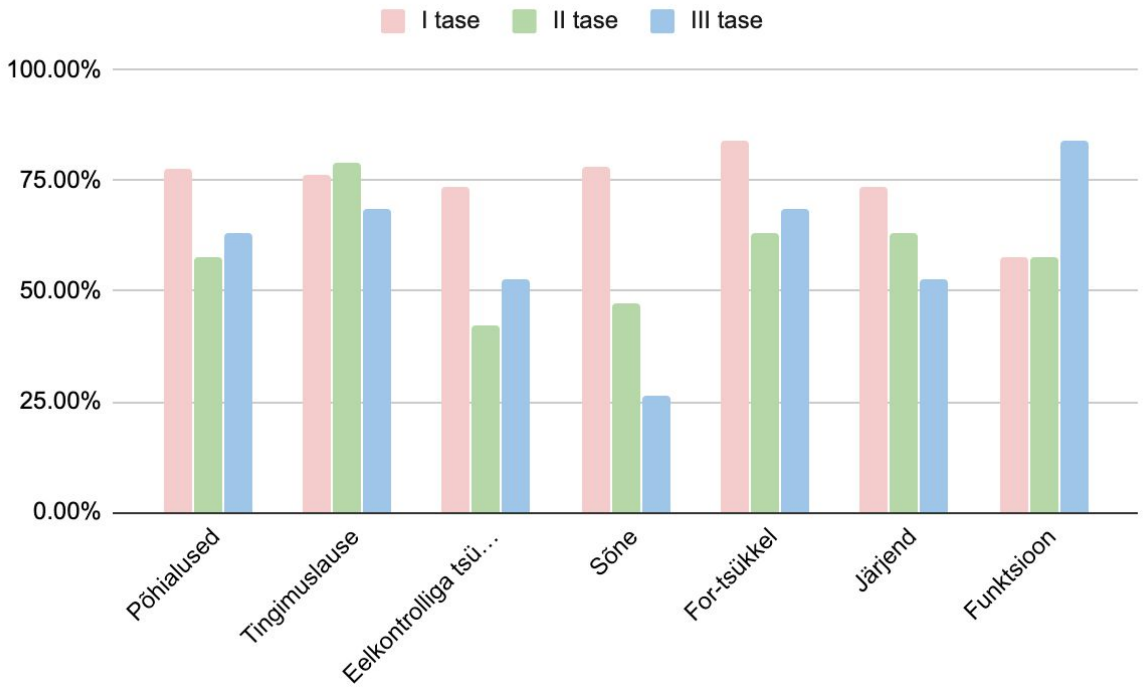
Iga konstrukti kohta on testis küsimusi kolmel tasemel. Võrdlesin testi piloteerimise tulemusi, võrreldes esialgu, kui palju kokku on iga konstrukti iga taseme kohta õigeid või osaliselt õigeid vastusi (Joonis 15) ning siis, kui palju on valesid vastusi (Joonis 16). Võtsin analüüsiks mittevaled vastused, mis tähendab, et need on õiged ja osaliselt õiged vastused kokku. Kahe konstrukti puhul on III taseme kohta valesid vastuseid vähem kui I taseme kohta (“Järjend”, “Funktsioon”). Ainult kahe konstrukti (“Tingimuslause”, “Sõne”) puhul on II taseme küsimused saanud vähem valesid vastuseid, kui III taseme küsimused. Selle põhjus võib olla selles, et III taseme küsimused pole piisavalt rasked, aga ka selles, et küsimused on mitmikvalikuga, mille puhul on tõenäosus vastata midagi õigesti suurem, kui siis, kui ainult üks vastus on õige. Täpsema analüüsi jaoks esitasin diagrammil vastuste keskmise hinde (Joonis 17). Selle järgi on konstrukti “Funktsioon” puhul I taseme tulemus halvem kui III taseme tulemus. II taseme tulemus on parem kui III taseme tulemus kolme konstrukti puhul (“Tingimuslause”, “Sõne”, “Järjend”). Küsimuste definitsiooni järgi on I tase kõige lihtsam ning III tase on kõige raskem. Analüüsist ei tule selline seos välja.



Joonis 15. Õigete ja osaliselt õigete vastuste arv konstrukti taseme kohta



Joonis 16. Valede vastuste arv konstrukti taseme kohta



Joonis 17. Vastuste keskmine tulemus konstrukti taseme kohta

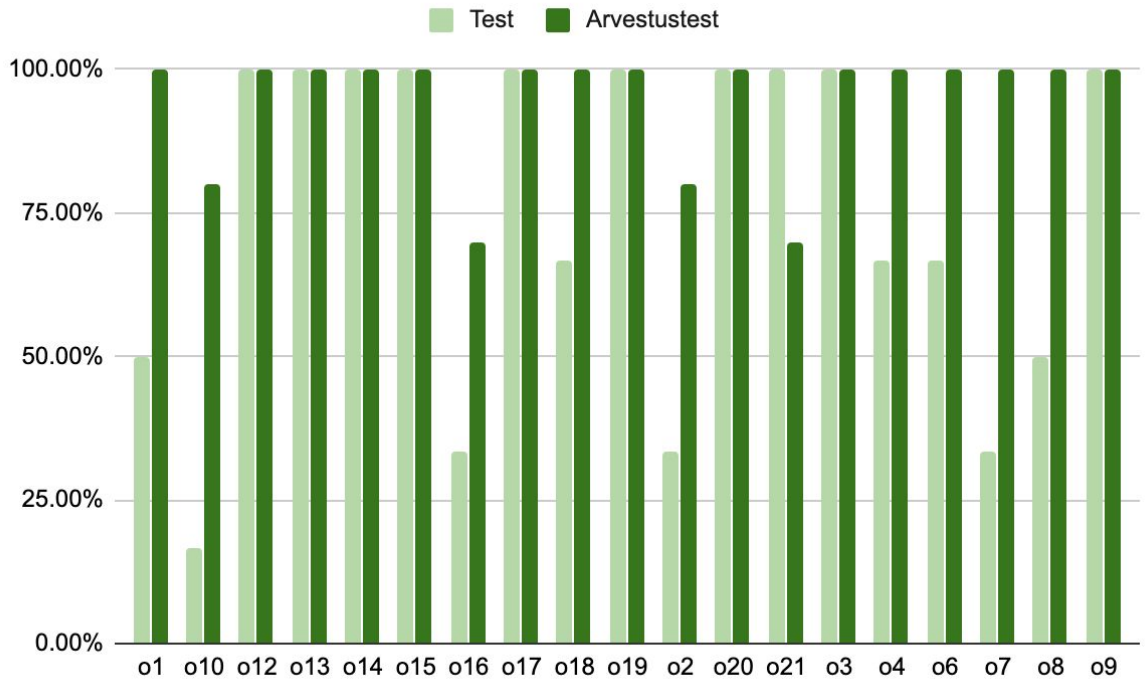
3.3.3. Testi ning arvestustesti tulemuste võrdlemine

Järgmisena uurisin, kuidas on omavahel seotud testi ning arvestustesti küsimused sama konstrukti kohta. Arvestustesti puhul ei olnud eesmärki kontrollida konkreetset konstrukti, ühes küsimuses on küsitud mitme konstrukti kohta.

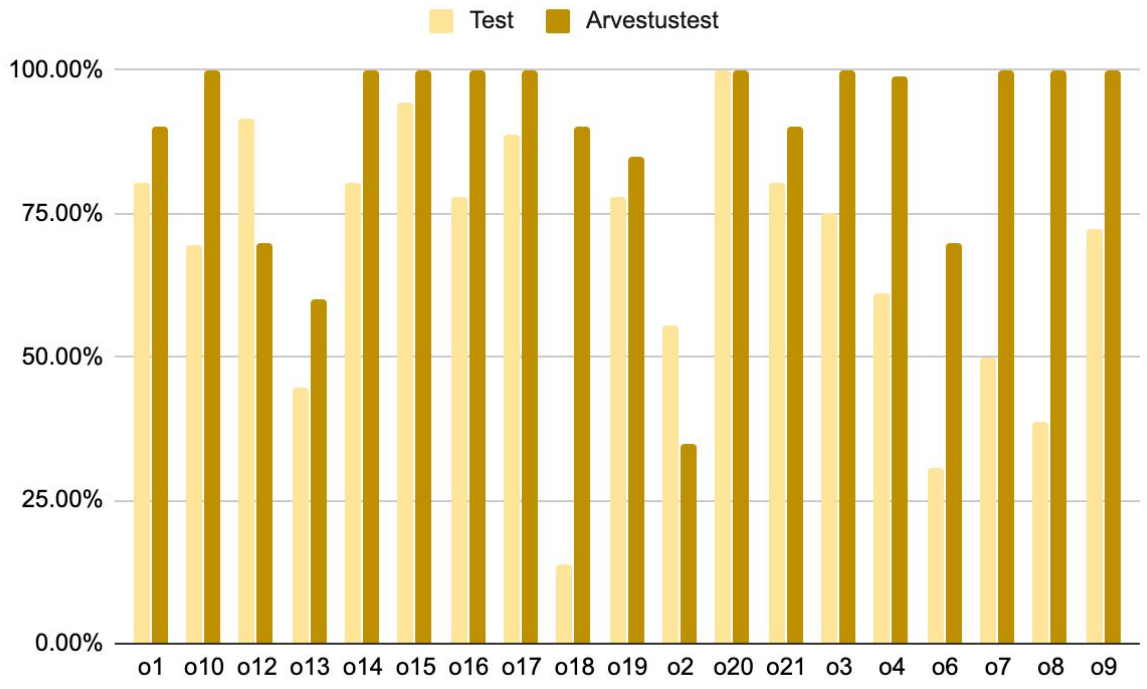
Arvestustesti küsimused kontrollivad järgmisi konstrukte: küsimused 1 kuni 5 kontrollivad konstrukti "Tingimuslause", 6. küsimus kontrollib konstrukte "For-tsükkel" ning "For-tsükkel. Järjend" (edasi "Järjend"), 7. küsimus kontrollib konstrukti "Eelkontrolliga tsükkel" ning "Funktsioon", 8. küsimuses kontrollitakse oskust lugeda/kasutada taanet tingimuslauses ning for-tsüklis. 8. küsimuses on koos kolm konstrukti ning ma jätan selle keerukuse tõttu analüüsist välja. Edasi võrdlen arvestustesti ning testi küsimuste tulemusi vastavalt sellele, mis konstruktide kohta need küsimused on (nt arvestustesti 1 kuni 5 küsimuste tulemused võrdlen testi 4 kuni 6 küsimuste tulemustega, kuna nad kontrollivad teadmisi konstrukti "Tingimuslause" kohta).

Võrdluseks koondasin õppijate tulemused tabelisse (Lisa 5). Tabelis on esitatud testi küsimuste 4–6 vastuste keskmine ning arvestustesti küsimuste 1–5 vastuste keskmine (tingimuslause); testi 13–15 (for-tsükkel), 16–18 (järjend), 13–18 (for ja järjend) küsimuste vastuste keskmine ning arvestustesti küsimuse 6 tulemus; testi 7–9 (while-tsükkel), 19–21 (funktsioon) vastuste keskmine ning arvestustesti küsimuse 7 tulemus. Tabeli põhjal loodud diagrammidel on toodud õppijate testi ning arvestustesti tulemus konstruktide kaupa. Joonisel

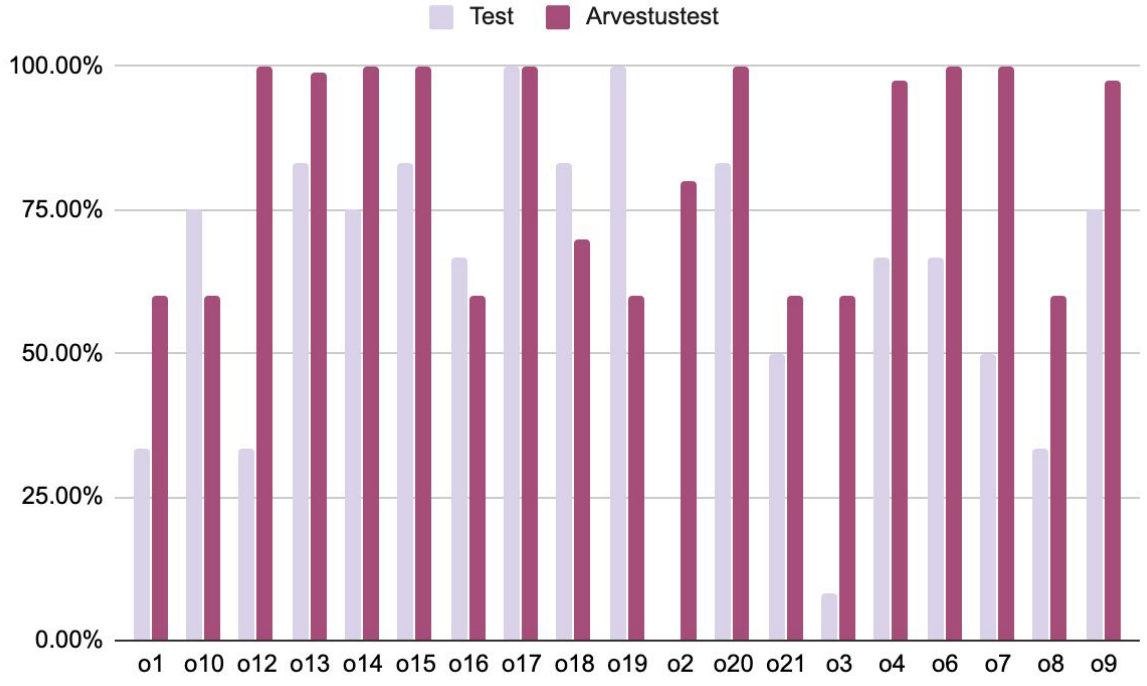
18 on näha õppijate testi ning arvestustesti tulemusi konstrukti “Tingimuslause” kohta. Ühel õppijal (o21) on arvestustesti tulemus halvem kui testi puhul, ülejäänutel on tulemus sama või parem. Joonisel 19 on näha õppijate testi ning arvestustesti tulemusi konstruktide “For-tsükkel” ning “Järjend” kohta. Kahel õppijal (o12, o2) on arvestustesti tulemus halvem kui testi puhul, ülejäänutel on tulemus sama või parem. Joonisel 20 on näha õppijate testi ning arvestustesti tulemusi konstruktide “While-tsükkel” ning “Funktsioon” kohta. Neljal õppijal (o10, o16, o18, o19) on arvestustesti tulemus halvem kui testi puhul, ülejäänutel on tulemus sama või parem. Täpsemaks analüüsiks võrdlesin nii konstruktipõhist kui ka kahe konstrukti aritmeetilist keskmist arvestustesti küsimuse tulemusega. Joonisel 21 on näha õppijate arvestustesti 6. küsimuse tulemus, sellega võrdlen kolm väärtust: 1) konstrukti “For-tsükkel” tulemuste keskmine, 2) konstrukti “Järjend” tulemuste keskmine, 3) kahe konstrukti tulemuste keskmine. Joonisel on näha, et “Järjendi” puhul on tihti tulemus halvem, kui konstrukti “For-tsükkel” puhul. Joonisel 22 on näha õppijate arvestustesti 7. küsimuse tulemuse, sellega võrdlen kolm väärtust: 1) konstrukti “Eelkontrolliga tsükkel” tulemuste keskmine, 2) konstrukti “Funktsioon” tulemuste keskmine, 3) kahe konstrukti tulemuste keskmine. Joonisel on näha, et “Eelkontrolli tsükli” ning “Funktsiooni” puhul ei eristu, milles rohkem eksitakse. Mõlemad küsimused (6 ja 7) on vaba vastusega küsimused, kus tuleb kirjutada, mida küsimuses pakutud kood väljastab ekraanile. Tulemuste täpsemaks analüüsiks on vaja teada, mida õppija vastas.



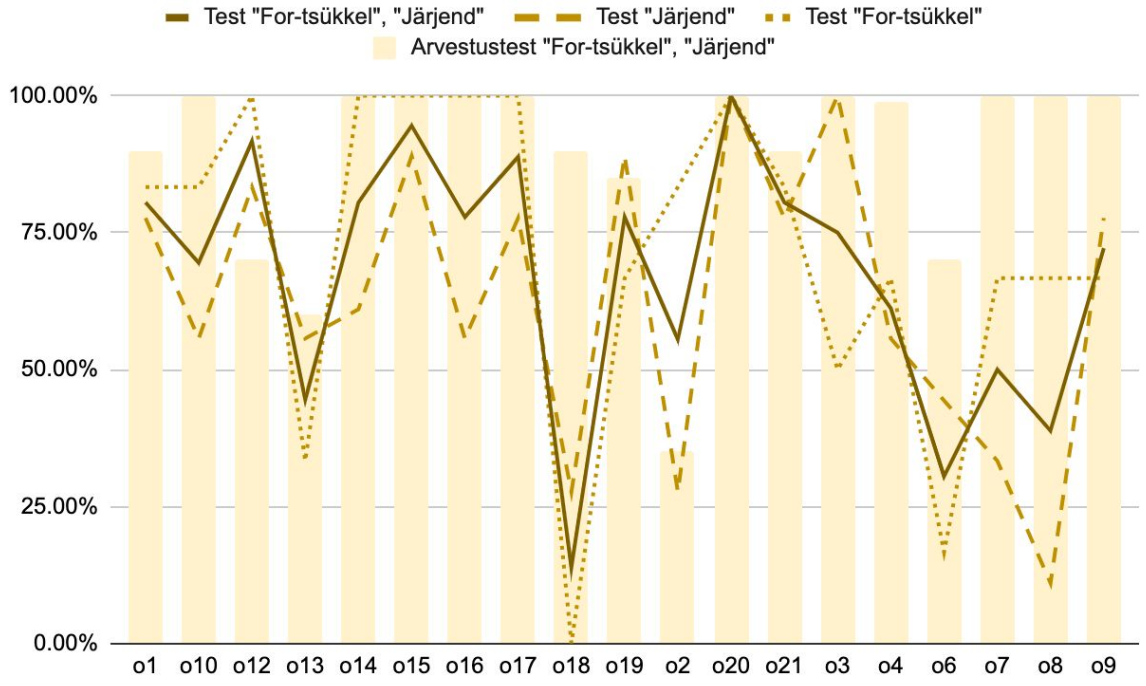
Joonis 18. Testi ning arvestustesti tulemused konstrukti “Tingimuslause” kohta



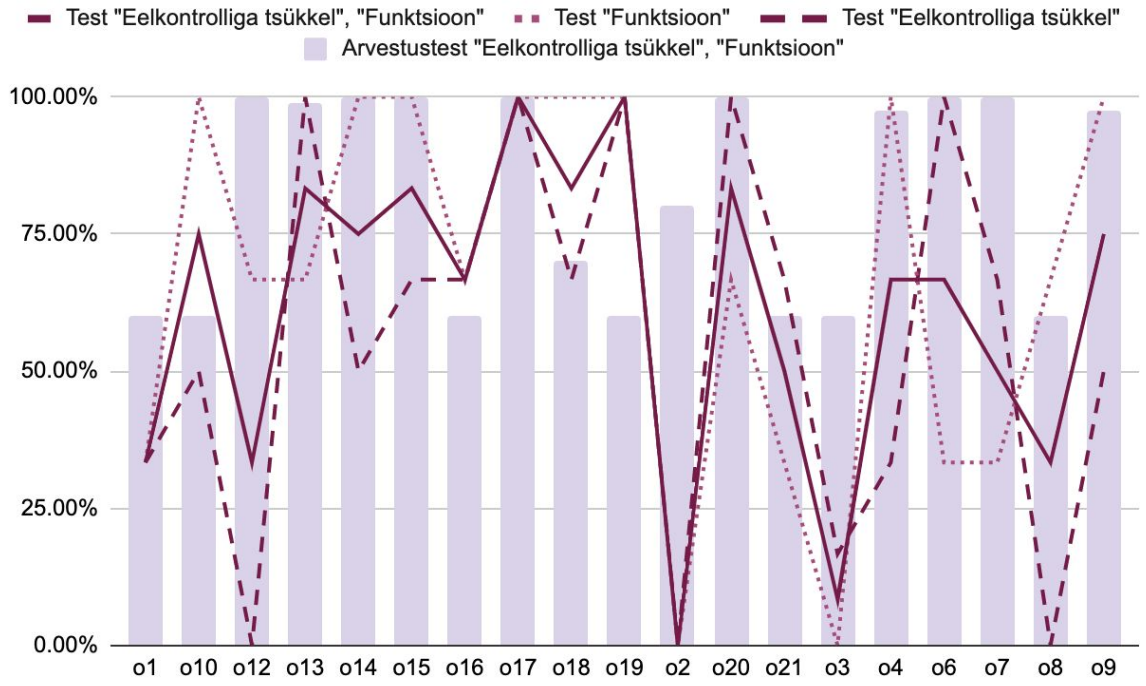
Joonis 19. Testi ning arvestustesti tulemused konstruktide “For-tsükkel” ning “Järjend” kohta



Joonis 20. Testi ning arvestustesti tulemused konstruktide “Eelkontrolliga tsükkel” ning “Funktsioon” kohta



Joonis 21. Testi ning arvestustesti tulemused konstruktide “For-tsükkel”, “Järjend” kohta (koos ja eraldi)



Joonis 22. Testi ning arvestustesti tulemused “While-tsükkel”, “Funktsioon” (koos ja eraldi)

3.4. Kokkuvõte

Käesolevas peatükis kirjutasin loodud testi valideerimise põhimõtetest ning protsessist. Sisu valideerimine algas testi skoobi täpse määramisega, edasi hindasid eksperdid testi skoopi, ülesannete arvu, tüüpe ning sõnastust. Ekspertide poolt aktsepteeritud testi lahendasid valjusti mõtlemise intervjuu käigus kaks intervjueeritavat. Intervjuu viisin läbi poolstruktureeritud kujul: testi lahendamine käis kindlas järjekorras, esitasin suunavaid küsimusi vastavalt vajadusele. Leitud kitsaskohad said parandatud ning toimus testi piloteerimine, mille käigus said testi lahendada TTL-kursuse õppijad. Uurisin piloteerimise tulemusi, selleks vaatasin testi lahendusi eraldiseisvalt ning koos õppijate arvestustesti ning programmeerimisülesande lahendamise tulemustega. Testi tulemuste analüüsimisel selgus, et küsimused 10 ja 18 eristuvad teiste küsimuste taustal, nende puhul täiesti valesid vastuseid (peaaegu) ei olnud. Mõlema küsimuse puhul on õigeks vastuseks, kui valida 3 valikvastust neljast. Nende küsimuste puhul tasub mõelda, kas vähendada õigete vastuste arvu või muuta antud küsimuse puhul hindamise kriteeriumi. Analüüsi tulemusena on näha, et II ja III taseme küsimused ei eristu nii palju, nagu võiks oodata ning III taseme küsimused ei ole võrreldes I tasemega piisavalt rasked. Selle põhjus võib olla nii küsimuste kvaliteedis, aga ka mujal. Testi ning arvestustesti konstruktide kaupa võrdlemisel on näha, et enamasti said õppijad arvestustestis parema tulemuse kui testis. Võib oletada, et test aitas õppimisel või et arvestustest oli liiga lihtne vms. Testi esmase lahendamise tulemuse ning TTL-kursuse arvestustöö tulemuse vahel on leitud tugev seos. Üldiselt sai teha analüüsi 19 tulemuse põhjal, täpsema tulemuse saavutamiseks on vaja viia läbi lisauuringuid.

Kokkuvõte

Programmeerimise aluste õppimise valdkonnas on vähe valiidseid teste, mis lubavad hinnata õppijate teadmisi ja oskusi. Selle põhjuseks on valiidse testi loomise keeruline ja aeganõudev protsess. Käesoleva magistritöö eesmärgiks oli luua valideeritud test programmeerimiskursuse "Tehnoloogia tarbijast loojaks" jaoks. Eesmärgi täitmiseks uurisin esmalt, milliseid teste kasutatakse programmeerimisoskuste hindamiseks maailmas. Leitud tulemused on toodud töö esimeses peatükis, kus kirjutasin, milliseid teste ja kuidas on loodud Venemaal (Nuriev jt töörühm), Ameerika Ühendriikides (Tew jt ning Parker jt töörühmad). Paraku on akadeemilisi uuringuid antud teemal vähe. Seejärel kirjeldasin magistritöö teises peatükis enda testi loomise protseduuri. Testi skoobi määramiseks uurisin TTL-kursusel taotlevaid õpiväljundeid, õpikus toodud mõisteid ja teemasid, kursuse õppekava nädalate kaupa. Analüüsi põhjal ning toetudes Tew', Parkeri jt kogemustele määrasin testi skoobi ja protseduuri. Iteratiivse protsessi lõpus (testi ülevaatamine ekspertide poolt, testi parandamine jne) valminud testi lahendasid valjusti mõtlemise intervjuu käigus kaks intervjueeritavat. Intervjuudest saadud tagasiside põhjal sai test uuesti parandatud. Intervjuude käigus sai tuvastatud, et testis on küsimusi, mis aitavad ära arvata vastuseid eelmistele küsimustele. Luua täiesti iseseisvaid küsimusi on väga raske. Tavaline soovitus sellises olukorras on mitte lubada õppijale minna eelmise küsimuse juurde. Testi edaspidise kasutamise juures tasub sellele aspektile mõelda. Parandatud test jõudis sihtrühmani seeläbi, et seda said piloteerida TTL-kursuse õppijad. Lõplikus testis on kokku 21 küsimust, 3 eri tasemel küsimust iga konstrukti kohta ("Põhialused", "Tingimuslause", "Eelkontrolliga tsükkel", "Sõne", "For-tsükkel", "Järjend", "Funktsioon"). Küsimused on kolmel tasemel: definitsioon, koodi järgimine, koodi lõpetamine. Kõik küsimused on valikvastustega küsimused.

Testi valideerimise meetoditeks oli sisuvalideerimine, mis toimus läbi detailse testi skoobi määramise ning testi ekspertide poolt ülevaatamise. Edasi uurisin võrdelist valiidsust, mis näitas, et testi tulemuste ning TTL-kursuse arvestustöö tulemuse vahel on tugev korrelatsioon. Lisaks uurisin testi kirjeldava statistika abil. Analüüsiks oli kasutusel 19 õppija esmase lahendamise tulemused. Test sai suuresti loodud Tew' testi eeskujul. Tew' järgi küsimuste kolm taset on keerukuse kasvavas järjekorras. Analüüsi tulemusena ei ole kasvavat keerukust näha: ainult ühe konstrukti puhul on I taseme küsimuse tulemus nõrgem kui III taseme küsimuse tulemus. Testi tulemuste analüüsimisel tuli välja, et kahe küsimuse puhul pole keegi (või ainult 1) vastanud täiesti valesti. Selle põhjus võib olla selles, et nende küsimuste puhul on 3 vastust neljast õiged. Nende küsimuste puhul tasub mõelda, kas muuta peibutusvastuste sisu ja/või muuta hindamist. Testi ning TTL-kursuse arvestustesti tulemuste

analüüsimisel võrdlesin, kuidas on omavahel seotud sama konstruktiga seotud küsimuste tulemused. Arvestustest on loodud teistel põhimõtetel ning selle küsimused ei ole puhtalt ühe konstrukti põhised. Võrdlemiseks kasutasin tulemuste aritmeetilist keskmist. Üldiselt on nende küsimuste puhul näha, et arvestustesti tulemus on parem kui testi oma. Miks, sellele küsimuse olemasolevate andmete põhjal vastata ei saa. Testi piloteerimisel osalesid vähem kui pooled arvestustööd teinud kursuse õppijad. Seega analüüsi jaoks ei moodustunud kõikset valimit. Õppijatel oli lubatud lahendada testi mitu korda. Testi viimase lahenduse ning kursuse arvestustöö tulemuste vahel on nõrgem korrelatsioon, kui testi esmase tulemuse ning kursuse arvestustöö tulemuste vahel. Täiendavaid analüüse on vaja, et otsustada, kas selline mitmekordne lahendamine aitab õppijaid. Testi analüüs annab lisamõtteid, kuidas saab muuta testi paremaks, samas annab see kindlust, et isegi praegu oma mittetäielikus vormis on see hea instrument õppijate oskuste/teadmiste mõõtmiseks. Testi rakendamisel kursuse alguses saab teada, mis on õppijate tase alguses ning sellest lähtuvalt kursuse õpetamist organiseerida. Testi piloteerimine viidi läbi TTL-kursuse lõpus ning selle lahendamise tulemused näitavad hästi, millised on õppijate väljavaated kursuse lõpetamisel.

Käesolevas töös loodud test on kättesaadav töö retsenseerijale, kaitsmiskomisjoni liikmetele ning TTL-kursuse õppejõududele (Lisa 6). Testi ei ole avalikult tööle lisatud, et säilitada selle õppijate seas laia levitamise eest.

Viidatud kirjandus

AKIT / Andmekaitse ja infoturbe leksikon. http://akit.cyber.ee/

EKI / EKI ühendsõnastik 2020. http://sonaveeb.ee/

Gibadullina / Гибадуллина Э. А. (2017). Алгоритм создания электронного теста на основе показателя глубины усвоенных знаний для оценки качества владения компетенцией // *Материалы IX Международной студенческой научной конференции «Студенческий научный форум»*

Vaadatud 17.05.2020 https://scienceforum.ru/2017/article/2017035373

Gierl, M. J., Bulut, O., Guo, Q. & Zhang, X. (2017). Developing, Analyzing, and Using Distractors for Multiple-Choice Tests in Education: A Comprehensive Review. *Review of Educational Research, 87*(6), 1082-1116. doi:10.3102/0034654317726529

Grover, S. (2020). Designing an Assessment for Introductory Programming Concepts in Middle School Computer Science. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 678-684). doi:10.1145/3328778.3366896

HAR / Hariduse ja kasvatuse sõnaraamat. http://www.eki.ee/dict/haridus/

Hertz, M. (2010). What do" CS1" and" CS2" mean? Investigating differences in the early courses. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 199-203). doi:10.1145/1734263.1734335

Higgins, C., McAvinia, C., O'Leary, C. & Ryan, B. J. (2019). A Study of First Year Undergraduate Computing Students' Experience of Learning Software Development in the Absence of a Software Development Process. *11th International Conference on Computer Supported Education* CSEDU 19, Heraklion, Crete 2nd – 4th May 2019. Vaadatud 17.05.2020 https://arrow.tudublin.ie/schfsehcon/27/

Karus, T. (2015). *Üldhariduskooli ja kutsehariduskooli õpetajate suhtumist väljendavad hinnangud kujundava hindamisega seotud tegevustesse.* Bakalaureusetöö, Tartu Ülikool.Vaadatud 17.05.2020 http://hdl.handle.net/10062/48059

Kilgour, J. M. & Tayyaba, S. (2015). An investigation into the optimal number of distractors in single-best answer exams. *Advances in Health Sciences Education, 21*(3), 571-585. doi:10.1007/s10459-015-9652-7

Kotkas, K. (2018). *Programmeerimise algkursuste eksamid.* Bakalaureusetöö, Tartu Ülikool.

Vaadatud 17.05.2020 http://hdl.handle.net/10062/65985

Laanpere, M., Niglas, K., Osula, K. & Pata, K. (2013) *Arvuti kasutamine uurimistöös. Informaatika valikaine e-õpik gümnaasiumile.* Tallinna Ülikool. Vaadatud 17.05.2020 http://aku.opetaja.ee/wp-content/uploads/2013/05/AKU_opikv10.pdf

Lee, M. J. & Ko, A. J. (2015). Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research,* 237-246. doi:10.1145/2787622.2787709

Mikk, J. (2002). *Ainetestid: loengukonspekt TÜ üliõpilastele.* Tartu Ülikool.

Vaadatud 17.05.2020 http://kodu.ut.ee/~jaanm/ainetestid.pdf

MKM / MKM hakkab koostöös Tartu Ülikooliga noortele programmeerimist õpetama. Tartu Ülikool, 12 Dec. 2019 Vaadatud 17.05.2020 http://www.ut.ee/et/uudised/mkm-hakkab-koostoos-tartu-ulikooliga-noortele-programmeerimist-opetama

Mulin, A. (2012) *Ülekantavate oskuste arendamine infotehnoloogia eriala õppes programmeerimise kursuse näitel.* Magistritöö, Tartu Ülikool. Vaadatud 17.05.2020 http://hdl.handle.net/10062/28532

Niglas, K. (2013). Faktoranalüüs. Vaadatud 17.05.2020 http://www.cs.tlu.ee/~katrin/wp/wp-content/uploads/2013/11/faktor.pdf

Nguen & Ku / Нгуен, Н. К. & Ку, Д. Т. (2018) Проектирование образовательных систем нового поколения с цифровыми технологиями подготовки. *Научно-технический прогресс: актуальные и перспективные направления будущего сборник материалов IX Международной научно-практической конференции*. С. 331-337. Vaadatud 17.05.2020 https://elibrary.ru/item.asp?id=37050490

Nuriev, Starygina & Ahmetshin / Нуриев, Н. К., Старыгина, С. Д. & Ахметшин, Д. А. (2015). Дидактическая инженерия: логистика профессионального развития на основе обучения. *Образовательные технологии и общество, 18(2).* Vaadatud 17.05.2020 https://cyberleninka.ru/article/n/didakticheskaya-inzheneriya-logistika-professionalnogo-razvitiya-na-osnove-obucheniya

Nuriev, Starygina & Samerhanov / Нуриев, Н. К., Старыгина, С. Д. & Самерханов, И. З. (2018). Надежность результата теста для оценки качества владения компетенцией. *Современные проблемы безопасности жизнедеятельности: интеллектуальные транспортные системы и ситуационные центры* (pp.

261-271). Vaadatud 17.05.2020 https://elibrary.ru/item.asp?id=38227619

Parker, M. C., Guzdial, M. & Engleman, S. (2016). Replication, validation, and use of a language independent CS1 knowledge assessment. In *Proceedings of the 2016 ACM conference on international computing education research* (pp. 93-101). doi:10.1145/2960310.2960316

Puniste, S. (2015). *Eesti gümnaasiumides õpetatavad programmeerimiskursused.* Bakalaureusetöö, Tartu Ülikool. Vaadatud 17.05.2020 http://hdl.handle.net/10062/56092

Puusepp, S. (2019). *Programmeerimisülesannete kogu gümnaasiumi valikkursusele „Tarkvaraarendus“.* Bakalaureusetöö, Tartu Ülikool. Vaadatud 17.05.2020 http://hdl.handle.net/10062/66246

Simon, Clear, A., Carter, J., Cross, G., Radenski, A., Tudor, L. & Tõnisson, E. (2015). What's in a name? International interpretations of computing education terminology. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 173-186). doi:10.1145/2858796.2858803

Simon & Snowdon, S. (2011, August). Explaining program code: giving students the answer helps-but only just. In *Proceedings of the seventh international workshop on Computing education research* (pp. 93-100). doi:10.1145/2016911.2016931

Starygina & Nuriev / Старыгина, С. Д. & Нуриев, Н. К. (2017). Дидактическая инженерия как новый тренд в обучении и диагностике. *Образовательные технологии и общество*, *20*(4). Vaadatud 17.05.2020 https://cyberleninka.ru/article/n/17323280

Starygina, Nuriev & Garifjanov / Старыгина, С. Д., Нуриев, Н. К. & Гарифьянов, Н. Ф. (2018). Оценка качества объекта (учебного курса): построение модели, автоматизация расчетов, примеры реализации. *Образовательные технологии и общество*, *21*(2). Vaadatud 17.05.2020 https://cyberleninka.ru/article/n/otsenka-kachestva-obekta-uchebnogo-kursa-postroenie-modeli-avtomatizatsiya-raschetov-primery-realizatsii

STATS / Standardipõhine Tarkvaratehnika Sõnastik. http://stats.cyber.ee/

Tchelyshkova / Челышкова, М. Б. (2002). *Теория и практика конструирования педагогических тестов* (p. 432). М.: ЛОГОС.

Tew, A. E. (2010). *Assessing fundamental introductory computing concept knowledge in a language independent manner.* Doctoral dissertation, Georgia Institute of Technology.

Vaadatud 17.05.2020 http://hdl.handle.net/1853/37090

Tew, A. E. & Guzdial, M. (2010). Developing a validated assessment of fundamental CS1 concepts. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 97-101). doi:10.1145/1734263.1734297

Tew, A. E. & Guzdial, M. (2011). The FCS1: a language independent assessment of CS1 knowledge. *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 111-116). doi:10.1145/1953163.1953200

TTL / Programmeerimiskursus "Tehnoloogia tarbijast loojaks". (i.a.). Vaadatud 17.05.2020 http://didaktika.cs.ut.ee/progttl/

Tulviste, E. (2013). *5-6-aastaste laste kõnetesti diagnostiline valiidsus.* Bakalaureusetöö, Tartu Ülikool. Vaadatud 17.05.2020 https://web-proxy.io/proxy/dspace.ut.ee/bitstream/handle/10062/31994/tulviste_epp.pdf?sequence=1&isAllowed=y

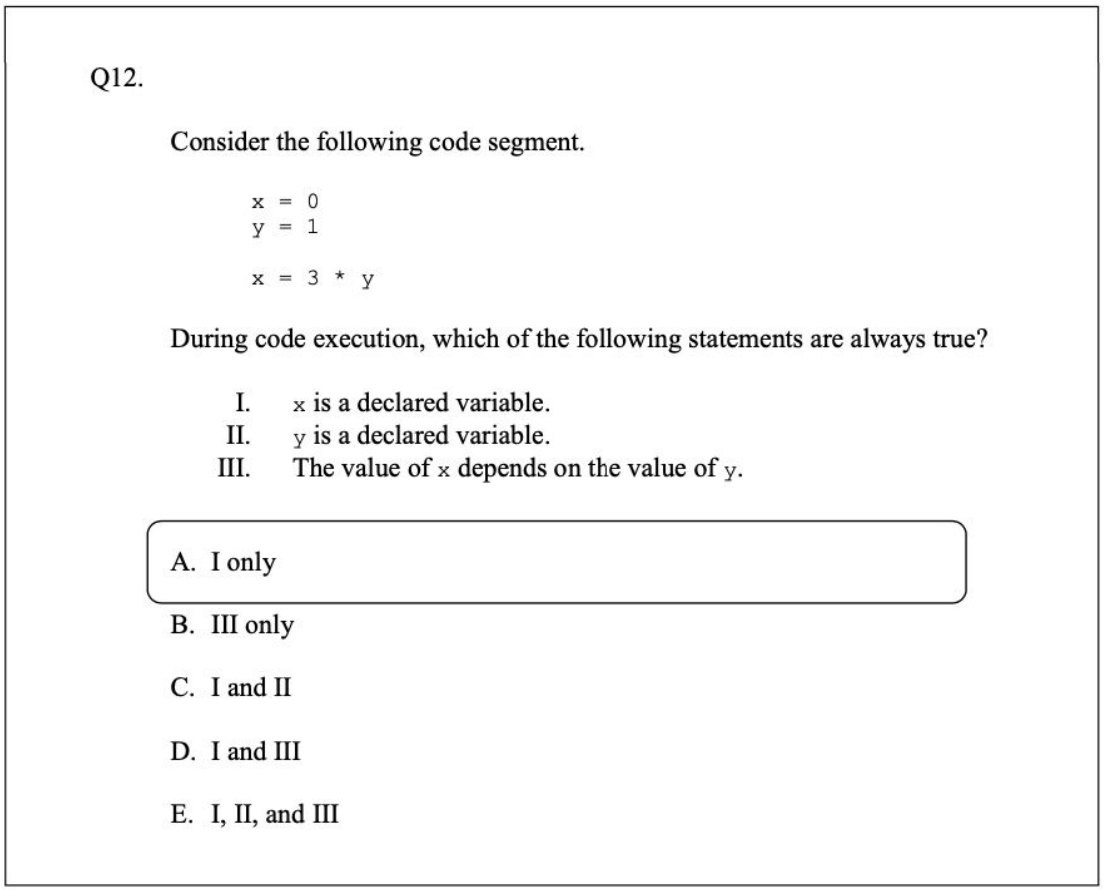
Tõnisson, E., Palts, T., Säde, M., Tõnisson, K. et al. (i.a.). *Programmeerimine*. Tartu Ülikool. Vaadatud 17.05.2020 http://web.htk.tlu.ee/digitaru/programmeerimine/

Võgotski / Выготский, Л. С. (1996). Проблема обучения и умственного развития в школьном возрасте. *Психологическая наука и образование*, *1*(4), 5-18. Vaadatud 17.05.2020 https://psyjournals.ru/files/2162/psyedu_1996_n4_Vygotsky.pdf

Xie, B., Davidson, M. J., Li, M. & Ko, A. J. (2019). An item response theory evaluation of a language-independent CS1 knowledge assessment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 699-705). doi:10.1145/3287324.3287370

Lisad

Lisa 1. A.Tew' (2010) testi küsimuste näited



Q12.

Consider the following code segment.

```
x = 0
y = 1

x = 3 * y
```

During code execution, which of the following statements are always true?

I. `x` is a declared variable.
II. `y` is a declared variable.
III. The value of `x` depends on the value of `y`.

A. I only

B. III only

C. I and II

D. I and III

E. I, II, and III

Küsimus definitsiooni tasemel (Tew, 2010)

Consider the following code segment.

```
flag1 =  False AND (True OR False)
flag2 = (False AND True) OR (False AND False)
flag3 = (False OR True) OR False
flag1 = (flag1 OR (NOT flag2)) AND flag3
```

After the above code is executed, which of the following statements are true?

```
I.    flag1 == True
II.   flag2 == True
III.  flag3 == True
```

A. I only

B. II only

C. III only

D. I and III

E. I, II, and III

Küsimus koodi järgimise tasemel (Tew, 2010)

Given the equation for computing the surface area of a cylinder:

$$\textbf{surface\_area} = 2\pi r^2 + 2\pi rh$$

Which of the following code segments can be used to complete the equation for computing the surface area of a cylinder?

```
surface_area = _________?__________
```

A. `2πrr + 2πrh`

B. `(2 * 3.14 * (r * r)) + (2 * 3.14 * r * h)`

C. `(2 * π * (r * r)) + (2 * π * r * h)`

D. `2(3.14 * (r * r)) + 2(3.14 * r * h)`

E. `(2 * (3.14 * r) * (3.14 * r)) + (2 * 3.14 * r * h)`

Küsimus koodi lõpetamise tasemel (Tew, 2010)

Lisa 2. A.Tew' (2010) pseudokoodi juhendi esimene lehekülg

Pseudo-code Guide

Syntax	Examples
Statements *variable* = *expression* Math Operators: +, -, *, /, % Relational Operators: ==, <, <=, >, >=, != Conditional Operators: AND, OR, NOT Booleans: True/False	total = 0 x = 4 * 3 name = "John Smith" flag = True answer = 1 AND (1 OR 0)
Print # print without new line PRINT *value, value* # print value with new line PRINTLN value	**Output** PRINT "Hello" ➡ Hello World PRINTLN "World" PRINTLN "Hello","World" ➡ Hello World
If Statement (Conditional) IF *condition* THEN *statement(s)* …… ELSE IF *condition* THEN *statement(s)* ELSE *statement(s)* ENDIF	IF testScore >= 90 THEN grade = 'A' ELSE IF testScore >= 80 THEN grade = 'B' ELSE IF testScore >= 70 THEN grade = 'C' ELSE grade = 'F' ENDIF
For Loop (Definite) # counter from *start-value* up to but not # including *end-value* FOR *counter = start-value* TO *end-value* BY # DO *statement(s)* …… ENDFOR	FOR x = 1 to 5 BY 1 DO xSquared = x * x PRINTLN x, xSquared ENDFOR ➡ **Output** 1 1 2 4 3 9 4 16
While Loop (Indefinite) WHILE *condition* DO *statement(s)* …… ENDWHILE	count = 5 WHILE count < 9 DO PRINTLN count count = count + 1 ENDWHILE ➡ **Output** 5 6 7 8
Functions DEFINE *function-name*(*parameter, parameter, …*) *statement(s)* …… RETURN *value* ENDDEF	DEFINE findMax(num1, num2) IF num1 >= num2 THEN maxNum = num1 ELSE maxNum = num2 ENDIF RETURN maxNum ENDDEF

Lisa 3. Parkeri jt testi küsimuse kopeerimise näide

Parkeri jt (2016) kopeerimise teel loodud küsimuse näide: üleval küsimus originaalsest Tew' testist; all uus kopeerimise teel loodud küsimus.

Given the following code segment.

```
array = [5, 2, 1, 3, 4, 6, 0, 8, 9]
i = 0
even = 0
WHILE (i < length(array)) AND (array[i] != 0)
DO
        IF (array[i] % 2) == 0 THEN
                even = even + 1
        ENDIF
                i = i + 1
        ENDWHILE
```

What are the values of the variables `i` and `even` after the while loop completes its execution?

A. `i = 1; even = 0`
B. `i = 5; even = 3`
C. `i = 5; even = 4`
D. `i = 6; even = 3`
E. `i = 6; even = 4`

Given the following code segment.

```
array = [3, 6, 8, 1, 2, 0, 7, 2, 9]
i = 0
odd = 0
WHILE (i < length(array)) AND (array[i] != 0)
DO
        IF (array[i] % 2) == 1 THEN
                odd = odd + 1
        ENDIF
                i = i + 1
        ENDWHILE
```

What are the values of the variables `i` and `odd` after the while loop completes its execution?

A. `i = 1; odd = 0`
B. `i = 5; odd = 2`
C. `i = 5; odd = 3`
D. `i = 8; odd = 4`
E. `i = 8; odd = 5`

Lisa 4. TTL-kursuse teemade kategoriseerimine

Tabel asub aadressil:

https://docs.google.com/spreadsheets/d/1buvQSIgQ_U7avBcWBrAwXf9yqSSbJq_yqAiASfmFEQw/edit?usp=sharing

Lisa 5. Testi ning arvestustesti tulemuste võrdlemine konstruktide kaupa

	Tingimuslause		For-tsükkel ja järjend				Eelkontrolliga tsükkel (while) ja funktsioon			
	Test	Arvestus	Test			Arvestus	Test			Arvestus
Õppija	4, 5, 6 kesk	1, 2, 3, 4, 5 kesk	13,14,15 kesk	16, 17, 18 kesk	15-18 kesk	6	7, 8, 9 kesk	19, 20, 21 kesk	7-9, 19-21 kesk	7
o1	50.00%	100.00%	83.33%	77.67%	80.50%	90.00%	33.33%	33.33%	33.33%	60.00%
o10	16.67%	80.00%	83.33%	55.67%	69.50%	100.00%	50.00%	100.00%	75.00%	60.00%
o12	100.00%	100.00%	100.00%	83.33%	91.67%	70.00%	0.00%	66.67%	33.33%	100.00%
o13	100.00%	100.00%	33.33%	55.67%	44.50%	60.00%	100.00%	66.67%	83.33%	99.00%
o14	100.00%	100.00%	100.00%	61.00%	80.50%	100.00%	50.00%	100.00%	75.00%	100.00%
o15	100.00%	100.00%	100.00%	89.00%	94.50%	100.00%	66.67%	100.00%	83.33%	100.00%
o16	33.33%	70.00%	100.00%	55.67%	77.83%	100.00%	66.67%	66.67%	66.67%	60.00%
o17	100.00%	100.00%	100.00%	77.67%	88.83%	100.00%	100.00%	100.00%	100.00%	100.00%
o18	66.67%	100.00%	0.00%	27.67%	13.83%	90.00%	66.67%	100.00%	83.33%	70.00%
o19	100.00%	100.00%	66.67%	89.00%	77.83%	85.00%	100.00%	100.00%	100.00%	60.00%
o2	33.33%	80.00%	83.33%	27.67%	55.50%	35.00%	0.00%	0.00%	0.00%	80.00%
o20	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	66.67%	83.33%	100.00%
o21	100.00%	70.00%	83.33%	77.67%	80.50%	90.00%	66.67%	33.33%	50.00%	60.00%
o3	100.00%	100.00%	50.00%	100.00%	75.00%	100.00%	16.67%	0.00%	8.33%	60.00%
o4	66.67%	100.00%	66.67%	55.67%	61.17%	99.00%	33.33%	100.00%	66.67%	97.50%
o6	66.67%	100.00%	16.67%	44.33%	30.50%	70.00%	100.00%	33.33%	66.67%	100.00%
o7	33.33%	100.00%	66.67%	33.33%	50.00%	100.00%	66.67%	33.33%	50.00%	100.00%
o8	50.00%	100.00%	66.67%	11.00%	38.83%	100.00%	0.00%	66.67%	33.33%	60.00%
o9	100.00%	100.00%	66.67%	77.67%	72.17%	100.00%	50.00%	100.00%	75.00%	97.50%

Lisa 6. TTL-kursuse test

Käesolevas magistritöös loodud test on kättesaadav retsenseerijale, kaitsmiskomisjoni liikmetele ning TTL-kursuse õpejõududele. Test ei ole avalikult kättesaadav, et säilitada selle õppijate seas laia levitamise eest.

Test on kättesaadav lingil:
https://drive.google.com/open?id=1sQlpa-LGTWP3eE5LOLu6C6-17zTtXa3z

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Jelizaveta Reitsakas (24.09.1979),

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Programmeerimise sissejuhatava kursuse õppijate teadmisi ja oskusi mõõtev test", mille juhendajateks on Eno Tõnisson ja Marina Lepp,

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.

Tallinnas, 25.05.2020