TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Triinu Aug

Kontrolltööde koostamise ja hindamise juhendite loomine Tartu Ülikooli kursusele „Programmeerimine“

Bakalaureusetöö (9 EAP)

Juhendaja: Tauno Palts, PhD

Tartu 2022

Kontrolltööde koostamise ja hindamise juhendite loomine Tartu Ülikooli kursusele „Programmeerimine“

Lühikokkuvõte:

Töö eesmärgiks on luua Tartu Ülikooli kursusele „Programmeerimine“ juhendid kontrolltööde koostamiseks ja hindamiseks, et kahe kokkuvõtliku hindamise eesmärki täitva kontrolltöö erinevate variantide koostamine oleks õppejõudude jaoks üheselt arusaadav ja mugav ning oleks tagatud erinevate variantide raskusastmete võrdsus ja lahenduste ühtlane hindamine. Selle jaoks analüüsiti kursuse 2021/2022 sügissemestril toimunud kontrolltööde tulemusi, ülesandeid ja kontrolltöödele antud tagasisidet ning seejärel fikseeriti analüüsi tulemuste põhjal aspektid, millele kontrolltöid koostades peab tähelepanu pöörama. Loodud juhenditele andsid tagasisidet kursuse juhtivad õppejõud ning praktikumijuhendajad. Saadud tagasisidet arvesse võttes valmisid universaalsed ja detailsed juhendid, mis on kursuse töös hästi rakendatavad.

Võtmesõnad: programmeerimise algkursus, kontrolltöö, eksam, kontrolltööde koostamine, hindamisjuhend

CERCS: P175 Informaatika, süsteemiteooria; S270 Pedagoogika ja didaktika

Creating instructions for compiling and grading tests for the course “Computer Programming” at the University of Tartu

Abstract:

This study aims to create instructions for compiling and grading tests for the “Computer Programming” course at the University of Tartu. So the preparation of different variants of the two tests that serve the purpose of the summary assessment is unambiguous and convenient for the lecturers. Also, the equality of severity of the different variants and the uniform evaluation of the solutions is ensured. For this purpose, the results, tests themselves, and feedback on the tests carried out during the autumn semester of course 2021/2022 were analysed. Based on the analysis results, it was fixed, which must be paid attention to when preparing the tests. Leading lecturers and practice supervisors provided feedback on the instructions. Considering the feedback received, universal and detailed instructions were prepared, which can be well-applied in the course's work.

Keywords: introductory programming, CS1, test, examination, test compiling, evaluation guide, grading instructions

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

Sisukord

Sissejuhatus

Programmeerimise õppimise vastu on tänapäeval suur huvi, suuresti tänu sellele, et selles vallas on suur tööjõupuudus ja programmeerimise oskust nõudvate ametite palgad on keskmisest kõrgemal tasemel [1]. Ka Tartu Ülikooli programmeerimise algteadmisi õpetaval kursusel „Programmeerimine“ (LTAT.03.001) osaleb igal sügisel ligikaudu 400 üliõpilast. Sellise üliõpilaste mahu juures on kokkuvõtliku hindamisega seoses kaks põhilist võimalust: kas kõik üliõpilased lahendavad samu ülesandeid samal ajal, et tagada igaühe oskuste kontroll samadel alustel või lahendavad üliõpilased erinevatel aegadel veidi erinevaid ülesandeid. Esimesel juhul ei pruugi olla võimalik kõigil 400 üliõpilasel samal ajal ülesandeid lahendada, sest neil on ka muud kohustused või on keeruline leida kontrolltööde läbiviijaid ja sobivaid ruume. Samuti on nõnda keerulisem kontrollida, kas kõik lahendavad ülesandeid ausalt ja reeglitest kinni pidades. Teisel juhul saavad need väljakutsed küll lahenduse, kuid ülesandeid koostades peab silmas pidama, et erinevate variantide ülesanded oleksid raskusastmelt piisavalt sarnased, et tulemused oleksid võrreldavad, samas piisavalt erinevad, et hilisemat varianti lahendanud üliõpilastel ei oleks eelist seetõttu, et nad juba teavad, mida varasemas variandis küsiti.

Selleks, et Tartu Ülikooli kursusel „Programmeerimine“ oleks kahe kokkuvõtliku hindamise eesmärki täitva kontrolltöö erinevate variantide koostamine õppejõudude jaoks üheselt arusaadav ja mugav ning oleks tagatud erinevate variantide raskusastmete võrdsus, loodi bakalaureusetöö raames juhendid nende kontrolltööde ülesannete koostamiseks. Selle jaoks esmalt analüüsiti kursuse 2021. aasta sügissemestril toimunud kontrolltööde tulemusi, et aru saada, kui palju ühe kontrolltöö variantide lahendamise tulemused erinevad ning seejärel fikseeriti analüüsi tulemuste põhjal, millele kontrolltöid koostades peab tähelepanu pöörama. Arvestades, et kursusel hindavad kontrolltöid erinevad õppejõud, pöörati juhendite koostamisel tähelepanu ka ülesannete eest antavate punktide jaotumise põhimõtetele.

Selle töö esimeses peatükis antakse ülevaade programmeerimise õpetamise olulisusest ning erinevatest teadmiste kontrollimise võimalustest programmeerimise algkursusel. Samuti kirjeldatakse, kuidas toimub programmeerimise õpetamine ja teadmiste kontrollimine töö fookuses oleval kursusel „Programmeerimine“. Teises peatükis antakse ülevaade kontrolltööde tulemuste kogumise ja analüüsimise metoodikast ning kirjeldatakse juhendite loomise protsessi. Kolmas peatükk sisaldab kontrolltööde tulemuste analüüsi ning kirjeldab tulemuste

põhjal valminud juhendeid, mis on olemas ka töö lisades. Neljandas peatükis on töö autori soovitused juhendite rakendamiseks ning ideed võimalikeks edasiarendusteks.

1. Teoreetiline taust

Selles peatükis antakse ülevaade programmeerimise õpetamisest. Esmalt tuuakse välja, miks on oluline programmeerimist õpetada. Seejärel kirjeldatakse, millised võimalused on programmeerimise algkursusel kontrollida, kui hästi on üliõpilased vajalikud teadmised omandanud. Viimaks antakse ülevaade, kuidas toimub programmeerimise õpetamine ja teadmiste kontrollimine Tartu Ülikooli kursusel „Programmeerimine“, mille jaoks kontrolltööde juhendmaterjalid luuakse.

1.1 Programmeerimise õpetamise olulisus

Erinevad programmeeritavad seadmed on tänapäeval iga inimese elu lahutamatu osa. Selleks, et need seadmed oskaksid ja suudaksid vastata kasutaja vajadustele, on vaja inimesi, kes neid mitte ainult kasutada, vaid ka programmeerida oskaksid. Seega on vaja programmeerijaid ja erinevate tarkvaralahenduste arendajaid. Statistikaameti uuringu [1] järgi oli 2021. aasta septembri lõpu seisuga lihttööliste ja meditsiinitöötajate kõrval terav tööjõupuudus ka tippspetsialistidest arendajate seas. Lihttööliste puudus on seletatav madala töötasuga ning meditsiinitöötajate puudus tervisekriisiga. Kuid, nagu selgus statistikaameti uuringust, oli programmeerijate ja arendajate brutotöötasu mediaan teiste nõutud ametigruppide brutotöötasu mediaanidest märgatavalt kõrgem [1]. Järelikult võib programmeerijate puuduse taga olla lihtsalt see, et ei ole piisavalt oskajaid ja seega ongi vaja neid rohkem välja õpetada.

Programmeerimine on selline oskus, mida on võimalik ka iseseisvalt õppida ja alati ei nõuta töötajatelt erialast haridust. Siiski paistab välja, et töötajate puudus ei vähene hoolimata sellest, et aastate jooksul on programmeerimine populaarsemaks muutunud. 1999. aastal juhtis Kanada arvutiteadlane David Parnas ühes intervjuus [2] tähelepanu sellele, et ilmtingimata ei ole tarvis palju programmeerijaid, vaid häid programmeerijaid, sest ühe halva programmeerija tehtu parandamiseks on vaja vähemalt kahte head programmeerijat. Seega on oluline, et programmeerimist õpetataks kontrollitult ja väga heal tasemel.

Samas ei tuleks programmeerimist õpetada ainult informaatikutele. Kolumbia Los Andes Ülikooli teadlaste uuringu [3] järgi on programmeerimise õpetamine väga efektiivne viis algoritmilise mõtlemise arendamiseks. Algoritmiline mõtlemine on oluline oskus nii igapäevaste kui ka spetsiifilisemate probleemide efektiivseks lahendamiseks [4]. Oskus probleeme võimalikult efektiivselt lahendada peaks olema igal inimesel, kes soovib enda

erialal edukas olla. Aastatel 2011–2012 läbi viidud rahvusvahelise täiskasvanute oskuste uuringu PIAAC [5] tulemustest paistab aga, et näiteks tehnoloogiarikkas keskkonnas probleemilahendusoskuse puhul on Eesti täiskasvanud keskmisest kehvemad. Sama uuringu temaatilises aruandes [6] on ka välja toodud, et 2015. aastal valitses "Eestis vajadus ka arvutioskuste kõrgema taseme koolituse järele, sest tööst, ameti- või palgakõrgendusest olid väheste oskuste tõttu ilma jäänud ka heade oskustega inimesed". Seega on oluline lisaks kvalifitseeritud programmeerijate välja koolitamisele õpetada programmeerimist, mis arendab nii arvuti- kui probleemilahendusoskusi, ka neile, kes ei plaani hakata tarkvaraarendajateks.

1.2 Erinevad võimalused programmeerimise teadmiste kontrollimiseks

Biggs ja Tang [7] on kirjutanud raamatus „Õppimist väärtustav õpetamine ülikoolis: keskmes õppija tegevused", et ülikooli kursustel koosneb hindamine tavaliselt kahest peamisest osast: kujundavast hindamisest (ingl *formative feedback*) ja kokkuvõtlikust hindamisest (ingl *summative grading*). Nende sõnul on kujundava hindamise eesmärk kursuse jooksul üliõpilasele ja õppejõududele anda teada, kuidas teadmiste omandamine edeneb, et sellele protsessile kaasa aidata. Kokkuvõtliku hindamise eesmärk on materjali omandamise lõpuks anda üliõpilasele tagasisidet selle kohta, kui hästi on ta need teadmised omandanud [7]. Tartu Ülikooli kursuse „Programmeerimine" kontrolltööd on justkui vaheeksamid, mille eesmärk ongi hinnata, kui hästi on seni õpetatud materjal omandatud, et saaks edasi liikuda, seega töö eesmärgist lähtuvalt uuritakse selles peatükis lähemalt kokkuvõtliku hindamise, teisisõnu kontrolltööde ja eksamite koostamise võimalusi programmeerimise algkursustel.

2013. aastal uurisid Sheard jt millised on programmeerimise algkursuste eksamid [8]. Uuringus selgus, et 20 analüüsitud eksamist (5 erinevast riigist) 18 olid täielikult paberil sooritatavad ning 2 sellised, kus lisaks kirjalikule osale oli 50% ulatuses ka programmeerimise osa arvutis [8]. Bennedsen ja Caspersen [9] väidavad aga, et paberil sooritatav eksam ei ole sobilik viis üliõpilase programmeerimise oskuste kontrollimiseks, sest õppeprotsessis ja ka päriselus programmeeritakse arvutil. Selle asemel soovitavad nemad eksameid, kus kõiki teadmisi kontrollitaksegi ainult praktiliselt läbi programmeerimisülesannete lahendamise, sest need annavad tõesema pildi üliõpilaste programmeerimisoskustest ning võimaldavad hinnata ka programmide kirjutamise protsessi [9]. Haghidi ja Sheardi läbi viidud uuringust [10] tuli välja aga see, et üliõpilaste arvates on ainult arvutis programmeerimist nõudva eksami puhul õhkkond tunduvalt närvilisem ja seega pärsib

keskendumisvõimet, kuid kahe meetodi kombinatsioon oleks tõenäoliselt kõige meeldivam ja õiglasem viis oskuste ja teadmiste kontrollimiseks.

Kotkase bakalaureusetöös [11] tehtud uurimuse kohaselt on eksami paberosas levinumad viisid teadmiste kontrollimiseks valikvastustega küsimused nii teoreetiliste teadmiste, koodist arusaamise kui ka koodi eesmärgi kohta ning vabavastuselised küsimused, sealhulgas lünkülesanded ja koodijupi kirjutamist nõudvad ülesanded. Valikvastustega küsimuste eelis on nende hindamise lihtsus, kuid Kotkase sõnul võivad need jätta eksami sisulise kvaliteedi madalaks [11]. Samas koodi eesmärgi selgitamise puhul on valikvastustega küsimus eelistatum, sest nii ei ole võimalik üliõpilastel ülesannet valesti tõlgendada [11]. Vabavastuseliste küsimustega on võimalik hinnata üliõpilase laiemaid teadmisi, kuid eksamineerija peab oleme veendunud, et küsimus on üheselt mõistetav [11]. Koodijupi kirjutamist nõudvad ülesanded kontrollivad väga hästi üliõpilase programmeerimisoskust, kuid ei saa olla liiga mahukad ega keerulised, sest keerulise süntaksi kirjutamine paberil on väga ebamugav ja aeganõudev [11]. Kõiki paberosa küsimusi on võimalik esitada paberi asemel ka arvutis vastavate tehnoloogiliste vahendite abil.

Kotkase uurimuse [11] järgi on arvutiosas teadmiste kontrollimiseks kaks peamist meetodit: arvutis lahendatavad programmeerimisülesanded ning Parsoni programmeerimispusle (inglise keeles *Parson's Programming Puzzle*). Kotkase sõnul on arvutis lahendatavate programmeerimisülesannete eelis paberosa ees võimalus hinnata üliõpilase oskust väga paljusid erinevaid programmeerimise konstruktsioone üheks terviklikuks programmiks kokku kirjutada ning seega võimalus lühikese ajaraami sees rohkem teadmisi kontrollida [11]. Parsoni programmeerimispusle näol on tegemist programmeerimisülesande lahendamisega nii, et potentsiaalsed koodiread on ette antud ning lahendaja ülesandeks on nende seast leida vajalikud ning need õigesse järjekorda panna [11]. Parsoni programmeerimispuslest väiksemaid variante kasutatakse sageli eksamitel ja kontrolltöödel nii-öelda paberosas koos teiste valik- ja vabavastustega küsimustega. Järgmistes alapeatükkides luuakse ülevaade, milliseid teadmiste kontrollimise meetodeid rakendatakse kursuse fookuses oleval programmeerimiskursusel.

1.3 Tartu Ülikooli kursus „Programmeerimine“

Selles peatükis tutvustatakse Tartu Ülikooli kursuse „Programmeerimine“ ülesehitust ja sisu 2021/2022 õppeaasta sügissemestril. Kursuse kohta käiv info on võetud õppeaine veebilehelt

Tartu Ülikooli õppeinfosüsteemist, arvutiteaduse instituudi kursuste veebilehelt, Moodle'i keskkonnast ja Tartu Ülikooli arvutiteaduse instituudi programmeerimise algkursuse õpikust [12–15]. Edaspidi kasutatakse Tartu Ülikooli arvutiteaduse instituudi programmeerimise algkursuse õpikule [15] viitamisel lühendatud nime programmeerimise õpik.

Tegu on sissejuhatava programmeerimise kursusega, mis annab üliõpilastele alusteadmised programmeerimise põhikonstruktsioonidest ning esmased oskused algoritmide ja programmide koostamiseks. Õpetamine toimub programmeerimise õpiku alusel, milles kasutatakse programmeerimiskeelt Python. Tartu Ülikooli õppeinfosüsteemis on õpiväljunditeks esitatud järgmised oskused:

- tunneb ja oskab kasutada põhilisi programmeerimiskonstruktsioone: muutuja, avaldis, omistuslause, tingimuslause, tsükkel, alamprogramm, rekursioon, andmevahetus kasutaja ja failidega;
- tunneb põhilisi andmetüüpe ja -struktuure (täis- ja ujukomaarvud, tõeväärtused, sõned, järjendid jne) ning oskab kasutada vastavaid standardoperatsioone;
- oskab analüüsida ja üksikasjalikult selgitada programmi töö käiku ning programmi muuta, täiendada ja edasi arendada;
- oskab luua lihtsamale ülesandele vastava algoritmi, koostada ja korrektselt vormistada lahendusprogrammi ning seda siluda ja testida;
- oskab realiseerida programmeerimisalaseid projekte rühmakaaslastega koostöös.

Õppeaine vorm on päevaõpe ning maht 6 EAP-d ehk 156 akadeemilist tundi, millest 32 tundi on loengud veebi teel, 32 tundi praktikumid auditoorselt ning 92 tundi iseseisev töö.

Kursuse sisu ülevaade on välja toodud lisaks õppeinfosüsteemile ka Tartu Ülikooli arvutiteaduse instituudi kursuste veebilehel. Peamise õpikeskkonnana kasutatakse aga Moodle'i keskkonda ning seal on kõige kohta täpsemad juhised saadaval kõigile kursusele registreerunud üliõpilastele.

Kursuse jooksul toimub 16 loengut ning 16 praktikumi, iga nädal kumbagi üks kord kahe akadeemilise tunni jagu. Loengutes selgitatakse programmeerimise õpikus toodud teemasid teoreetilisest poolest ning praktikumides lahendatakse teemale vastavaid ülesandeid. Praktikumis lahendatavad ülesanded on praktikumijuhendaja valida, kuid peavad vastama teemale. Iga praktikumi alguseks on vaja lahendada kodutöö. Välja arvatud 6., 12. ning 16. nädalal, millal toimuvad praktikumi ajal vastavalt I kontrolltöö, II kontrolltöö ning projektide esitlemine. Kontrolltööd koosnevad arvestuslikust osast ning punktilisest osast. Kontrolltööde

ülesehitusest kirjutatakse täpsemalt järgmises alapeatükis. Projekti all mõeldakse mingit suuremamahulist programmeerimisülesannet, mille loovad ja lahendavad õpilased paaris ning mille teema on nad ise valinud. Lisaks on igal nädalal võimalik vastata elektroonilistele testidele loengumaterjali kohta ning lahendada lisaülesandeid. Kursuse lõpus toimub eksam, mis on ülesehituselt sarnane kontrolltöödele, kuid selles töös sellele tähelepanu ei pöörata, sest eksami vorm on muutumas.

1.4 Kontrolltööd kursusel „Programmeerimine“

Selles peatükis kirjeldatakse lähemalt, kuidas toimus kursusel „Programmeerimine“ 2021/2022 õppeaasta sügissemestril teadmiste kontrollimine. Kontrolltööde kohta käiv on pärit Moodle’i keskkonnast [14].

Kursuse kontrolltööd koosnevad kahest osast. Esimesena tuleb üliõpilastel lahendada arvestuslik osa. See on kuuest küsimusest koosnev Moodle keskkonnas läbiviidav test. Küsimused võivad olla kõigi seni läbitud teemade kohta, II kontrolltöö puhul fookusega teemadele, mis on omandatud peale I kontrolltööd. Küsimused on võetud Moodle küsimuste-pangast, kuhu on õppejõud need erinevatel õppeaastatel lisanud ja kategoriseerinud teemade järgi. Küsimuste hulgas on nii vabavastustega, valikvastustega kui ka lünkülesandeid. Enamik küsimusi nõuavad lühikese koodijupi kirjutamist või selgitamist. Küsimustele vastates pole lubatud kasutada kõrvaliste materjalide, vahendite ega inimeste abi. Arvestuslikus osas saab iga küsimuse õigesti vastamise eest 1 punkti, seega kokku 6 punkti. Osaliselt õige vastuse eest on võimalik saada ka 0,5 punkti. Need punktid aga ei lähe lõpptulemuse juures otseselt arvesse, vaid kui arvestuslikus osas saadakse vähemalt 5,5 punkti 6-st, saab üliõpilane kontrolltöö arvestatud. Kontrolltöö arvestatud saamine on tingimus eksamile pääsemiseks ja aine läbimiseks. Arvestusliku osa lahendamiseks on aega kuni 45 minutit. Kui see saab varem lahendatud, saab kohe asuda programmeerimise osa juurde.

Peale arvestuslikku osa tuleb üliõpilastel lahendada programmeerimise osa. See koosneb kahest 10-punktilisest ülesandest. Ülesanded tuleb lahendada programmeerimiskeskkonnas Thonny ning kasutada ka Thonny logimise funktsiooni, mis salvestab kõik üliõpilase tegevused selles keskkonnas. Lahenduste esitamisel tuleb üliõpilasel esitada ka logifail. Nii on õppejõududel võimalik tuvastada, kas üliõpilane lahendas ülesanded ise või kasutas kõrvaliste isikute abi, mis on keelatud. Kusjuures programmeerimise osas on lubatud

kasutada nii kursuse, isiklikke kui internetis leiduvaid materjale. Programmeerimis-ülesannetes on kirjeldatud mingit elulist olukorda või probleemi ning selgitatud kindlate funktsioonide ja põhiprogrammi tööd. Ülesande juures on ka näited sisendi ja väljundiga, et garanteerida, et kõik saavad ülesandest ühtmoodi aru. I kontrolltöö programmeerimise ülesanded katavad kõiki seni omandatud võtmeoskusi ning II kontrolltöö puhul on kindlasti kontrollitud oskusi, mis on omandatud peale I kontrolltööd. Kontrolltöö programmeerimise osa eest on võimalik saada 0–20 punkti, kusjuures eksamile pääsemiseks kontrolltöö punktidele lävendit pole, kuid üliõpilasel tuleb meeles pidada, et aine läbimiseks peab ta kõigist kursuse ülesannetest koguma kokku vähemalt pooled punktid kõigist võimalikest. Programmeerimise osa lahendamiseks on üliõpilasel aega kindlasti 1 tund, kuid kui arvestuslik osa saab lahendatud kiiremini kui 45 minutiga, siis ka sellest üle jäänud aeg.

Kui üliõpilasel arvestuslik osa ebaõnnestub, kuid ta on rahul punktidega, mis sai programmeerimise osast, on tal võimalik järeltööl lahendada ainult arvestuslikku osa. Kui üliõpilasel arvestuslik osa õnnestub, kuid soovib programmeerimise osa tulemust parandada, on tal see võimalus, kusjuures ta ei pea arvestuslikku osa uuesti lahendama, kuid ta peab arvestama, et programmeerimise osas läheb arvesse viimane tulemus, isegi kui tulemus on kehvem kui esialgne.

2021/2022 õppeaastal kontrolliti I kontrolltöös järgmiste teadmiste omandamist:

1. muutujad,
2. põhilised andmetüübid,
3. avaldised,
4. lihtlaused,
5. tingimuslaused,
6. korduslaused,
7. funktsioonid,
8. algoritmid,
9. lihtsam programmi suhtlemine kasutaja ja failidega.

II kontrolltöös kontrolliti põhiliselt nende teadmiste omandamist:

1. järjendid ja operatsioonid nendega,
2. ennikud ja operatsioonid nendega,
3. sõnastikud ja operatsioonid nendega,
4. hulgad ja operatsioonid nendega,

5. mitmemõõtmelised andmestruktuurid
6. mitmekordsed korduslaused,
7. keerulisem kasutaja ja failidega suhtlemine.

Seega katavad I ja II kontrolltöö kõik õpiväljundid peale rekursiooni, mis omandatakse peale II kontrolltööd ning projekti realiseerimise, mis on eraldi hinnatav ülesanne.

2021/2022 õppeaasta sügissemestril koostasid õppejõud kontrolltööde ülesanded eelmise aasta kontrolltööde ja hindamisjuhendite eeskujul ilma konkreetsemate suunisteta[1]. Kursuse alguses jaotati kontrolltööde koostamise ja läbiviimisega seotud ülesanded õppejõudude vahel ära, kusjuures iga õppejõud sai ise valida, milliseid ülesandeid ta täita sooviks. Kummagi kontrolltöö jaoks esitasid 3 praktikumijuhendajat kontrolltöö mustandid 2–3 nädalat enne kontrolltöö toimumist. Seejärel luges ja lahendas iga variandi läbi üks õppejõud ning esitas parandusettepanekud, mille üle koos teiste õppejõududega arutleti ja viidi muudatused sisse. Seejärel läbisid kõik ülesannete tekstid ka keelelise kontrolli. Kummagi kontrolltöö ülesannetes olid 3 lõplikku varianti valmis 3 päeva enne kontrolltöö toimumist.

Järgmises peatükis kirjeldatakse, milline oli protsess kontrolltööde koostamise juhendite loomiseks, et eelnevalt kirjeldatud protsess oleks fikseeritud ning seega ka selgem ja mugavam.

[1] 2021/2022 sügissemestri kontrolltööde koostamise kohta käiv informatsioon pärineb lõputöö juhendajalt ja programmeerimiskursuse vastutavalt õppejõult Tauno Paltsilt.

2. Metoodika

Selles peatükis antakse ülevaade kontrolltöö tulemuste kogumise ja analüüsimise metoodikast ning kirjeldatakse juhendite loomise protsessi. Peatüki esimeses osas on kirjas, milliseid kontrolltööde tulemustega seotud andmeid analüüsiti, milliseid vahendeid ja statistilisi meetodeid selle jaoks kasutati ning mis eesmärke analüüsimisel silmas peeti. Teises osas kirjeldatakse, mille põhjal ja mis põhimõtteid järgides kontrolltööde juhendid koostati ning millised etapid need läbisid enne lõpliku kuju saavutamist.

2.1 Andmete kogumine ja analüüs

Töö raames analüüsiti statistiliselt kursuse 2021/2022 sügissemestri I kontrolltöö, II kontrolltöö ja nende järeltööde arvestusliku ja programmeerimise osa tulemusi.

Arvestusliku osa puhul laaditi 2 nädalat peale kontrolltöö toimumist Moodle'ist alla arvestusliku osa tulemusi koondav tabel, kus oli iga üliõpilase kohta kirjas, mis kell ta lahendamist alustas, mis kell ta vastused esitas ning mitu punkti ta iga ülesande eest sai.

Programmeerimise osa puhul laaditi peale seda, kui õppejõud olid tööd ära hinnanud, Moodle'ist alla kõikide üliõpilaste programmeerimise osa lahendused, õppejõudude tagasiside igale tööle ning tabel, kus oli iga üliõpilase kohta kirjas, mis kell ta enda lahendused esitas ning mitu punkti ta programmeerimise osa eest kokku sai. Programmeerimise osa tulemuste paremaks analüüsimiseks arvutati programmeerimise osa lahendamisaeg lahenduste esitamise aja ja arvestusliku osa esitamise aja (või praktikumi alguse aja) vahena. Selleks, et kummagi ülesande tulemusi täpsemalt analüüsida, lisati andmetabelisse õppejõudude tagasiside põhjal ka kummagi ülesande eest saadud punktide arv. Tabelisse lisati ka, millist kontrolltöö varianti üliõpilane lahendas.

Eraldi koostati tabel, kus oli iga üliõpilase kohta kirjas, mis rühmas üliõpilane õppis ning unikaalne kood, mida kasutati anonümiseerimiseks. Anonümiseerimiseks ühendati tulemuste ja üliõpilase informatsiooni kohta käivad tabelid nime järgi, seejärel eemaldati nimi ning üliõpilase tulemusi jäi identifitseerima ainult unikaalne number.

Seega oli iga üliõpilase kohta kummagi kontrolltöö tulemuste analüüsimisel teada järgnev informatsioon:

1. unikaalne kood,
2. praktikumirühm,
3. arvestusliku osa lahendamise alustamise aeg,
4. arvestusliku osa lahendamise lõpetamise aeg,
5. eelmise kahe punkti põhjal arvutatud arvestusliku osa lahendamise aeg,
6. arvestusliku osa eest saadud punktide arv küsimuste kaupa,
7. programmeerimise osa lahenduste esitamise aeg,
8. punktide 7 ja 4 (või praktikumi alguse aja) põhjal arvutatud programmeerimise osa lahendamise aeg,
9. programmeerimise osa eest saadud punktide arv ülesannete kaupa ja kokku,
10. lahendatud kontrolltöö variandi number (1, 2, 3 või 0, mis tähistas järeltöid),
11. kui üliõpilane lahendas mitut varianti (lisaks põhivariandile ka järeltööd/järeltöid või ainult mitut järeltööd), siis oli tema kohta andmestikus iga variandi kohta eraldi rida.

Kursust alustas 2022/2022 õppeaasta sügissemestril 407 üliõpilast, kellest 6 lahkusid kursuselt juba enne I kontrolltöö toimumist. Kõigist kursusele registreerunud üliõpilastest 67 osalesid kursusel ainult veebi teel, mis tähendab, et nende jaoks ka praktikumid ja seega ka kontrolltööd toimusid veebi teel. Kursusel osalejatest hinnanguliselt 55% olid informaatika erialalt, 13% arvutitehnika erialalt, 12% füüsika, keemia, materjaliteaduse erialalt, 12% matemaatilise statistika erialalt ning 8% muudelt erialadelt.

Kokku analüüsiti 401 üliõpilase kontrolltööde tulemusi. Neist 21 puhul puudus informatsioon selle kohta, millises rühmas nad õppisid, sest nad olid kursuselt lahkunud I kontrolltöö tulemuste ja rühma andmete kogumise vahele jäänud aja jooksul. Analüüsides rühma mõju tulemustele, ei võetud arvesse puuduvate väärtustega üliõpilaste tulemusi, sest neid oli igas rühmas väga vähe ning ei olnud vajadust imputeerimiseks. Kõik 401 üliõpilast ei lahendanud mõlemat kontrolltööd, vaid 362 üliõpilase kohta on teada I kontrolltöö arvestusliku osa tulemus, 358 üliõpilase kohta on teada I kontrolltöö programmeerimise osa tulemus, 320 üliõpilase kohta on teada II kontrolltöö arvestusliku osa tulemus ning 327 üliõpilase kohta on teada II kontrolltöö programmeerimise osa tulemus.

Kontrolltööde tulemuste analüüsi eesmärk oli tuvastada:

1. kas arvestuslikus osas olid kõik küsimused sobiva raskusastmega, arvestades, et see kontrollis minimaalsete teadmiste omandamist,
2. kas programmeerimise osas olid kõik kontrolltöö variandid (sh järeltööd) sama raskusastmega, arvestades, et need kontrollisid samu teadmisi.

Tulemuste analüüsimiseks koostati ja korrastati tabelid esmalt Microsoft Exceliga ning seejärel analüüsiti rakendustarkvaraga R [16], kasutades lisaks standardteegile järgmiste pakettide võimalusi: openxlsx [17], tidyverse [18], reshape2 [19], knitr [20], scales [21], ggpubr [22]. Analüüsimiseks valiti rakendustarkvara R, sest tegu on vabavaralise programmiga, kus on kõik antud tööks vajalikud vahendid olemas ning töö autoril on selle programmiga kõige suurem kasutuskogemus võrreldes alternatiiviga (Python). Kuna analüüsimiseks ei võetud andmetest valimeid, vaid oli võimalik kasutada kõiki andmeobjekte, kasutati peamiselt ainult kirjeldava statistika meetodeid. Graafikute ja jaotustabelitega uuriti järgmist:

1. arvestusliku osa eest saadud punktide jaotumine,
2. arvestusliku osa eest saadud punktide jaotumine küsimuste kaupa,
3. arvestusliku osa eest saadud punktide jaotumine praktikumi rühmade kaupa,
4. arvestusliku osa eest saadud punktide ja programmeerimise osa eest saadud punktide vaheline seos,
5. programmeerimise osa eest saadud punktide jaotumine varianditi,
6. programmeerimise osa I ülesande eest saadud punktide jaotumine varianditi,
7. programmeerimise osa I ülesande eest saadud punktide jaotumine nendel töödel, kus II ülesande eest saadi 0 punkti, arvestades ka lahendamisaega, lahendatud varianti ja praktikumirühma,
8. programmeerimise osa II ülesande eest saadud punktide jaotumine varianditi,
9. programmeerimise osa II ülesande eest saadud punktide jaotumine nendel töödel, kus I ülesande eest saadi 0 punkti, arvestades ka lahendamisaega, lahendatud varianti ja praktikumirühma,
10. programmeerimise osa eest saadud punktide jaotumine praktikumi rühmade kaupa, arvestades ka lahendatud varianti,
11. programmeerimise osa eest saadud punktide ja lahendamisaja vaheline seos varianditi,

12. esimesel katsel programmeerimise osa eest saadud punktide võrdlus järeltööl saadud punktidega.

Programmeerimise osa eest saadud punktide jaotumist varianditi testiti ka Kruskal-Wallise testiga, mis on mitteparameetriline test jaotuste erinevuste tuvastamiseks rohkem kui kahes sõltumatus grupis ehk antud juhul punktide jaotumise erinevuse tuvastamiseks ühe kontrolltöö erinevate variantide puhul [23]. Variantide erinevust testiti 5%-sel olulisusnivool ehk variantide eest saadud punktide jaotumine loeti erinevaks, kui tõenäosus juhuslikust sama või veel ekstreemsemat tulemust (suurema erinevusega punktide jaotusi) saada oli 5% või vähem. Kui Kruskal-Wallise test kinnitas, et variantide eest saadud punktide jaotus erineb, võrreldi iga variandi jaotusi omavahel, et tuvastada, millise või milliste variantide jaotused ja kuidas erinevad teistest.

Kui arvestusliku osa tulemuste analüüsil tuvastati küsimusi, mille puhul saadud punktide jaotumine erines märgatavalt teistest, uuriti lähemalt nende sisu ja võeti leitut arvesse kontrolltööde juhendite koostamisel. Sarnaselt ka programmeerimise osa puhul – kui analüüsil leiti, et ühe kontrolltöö erinevate variantide eest saadud punktide jaotus erineb oluliselt, uuriti, millest võib erinevus olla põhjustatud, analüüsides ka taustatunnuseid ning võrreldes variante sisuliselt. Leitut võetigi põhilise sisendina kontrolltööde juhendite koostamisel.

2.2 Kontrolltööde koostamise ja hindamise juhendite loomine

Kontrolltööde koostamise juhendite loomisel lähtuti kontrolltööde tulemuste statistilise analüüsi tulemustest, ülesannete sisulisest analüüsist ning kontrolltöö ja eksami järgsete vabatahtlike tagasisideküsitluste vastustest. Samuti võeti arvesse, et 2022/2023 õppeaasta sügissemestril on plaanis kursus läbi viia uuendatud teemade läbimise järjekorraga.

Arvestusliku osa puhul uuriti peale tulemuste statistilise analüüsi tegemist täpsemalt, kuidas erinesid küsimused, mille õigesti vastamise protsent oli erinev. Seejärel uuriti kontrolltöö ja eksami järel läbiviidud tagasisideküsitluste vastuseid ning tehti neist järeldusi. Uuest teemade järjekorrast lähtuvalt fikseeriti, mis teemasid küsimused kindlasti katma peavad. Viimaks koostati kõiki tulemusi silmas pidades juhendid kummalegi kontrolltööle arvestusliku osa koostamiseks.

Programmeerimise osa puhul analüüsiti läbi kummagi kontrolltöö kolme põhivariandi ülesannete tekstid ja nõudmised ning fikseeriti variantide vahelised erinevused. Seejärel valiti kummastki kontrolltööst statistilise analüüsi tulemuste ja variantide erinevuste analüüsi põhjal variant, mis sobiks kõige paremini konkreetsete teadmiste hindamiseks. Selle variandi ülesannete struktuurist lähtuti suures osas üldisema juhendi koostamisel. Samuti töötati läbi kontrolltöö ja eksami järgsete tagasisideküsitluste vastused ning tehti neist järeldused. Viimaks koostati kõiki tulemusi silmas pidades juhendid kummalegi kontrolltööle programmeerimise osa koostamiseks, kus on kirjas suunised nii koostamise protsessiks kui ka nõudmised ülesannetele ning iga ülesande punktide jaotumise põhimõtted.

Kõigile juhenditele küsiti tagasisidet kursuse juhtivatelt õppejõududelt. Tagasisidestamiseks said nad läbi lugeda kummagi kontrolltöö arvestusliku ja programmeerimise osa koostamise juhendite mustandid (mis olid sama struktuuriga nagu lisas toodud juhendite puhtandid) ning iga osa kohta vastata võimalikult põhjalikult, mida ja kuidas nad antud juhendi juures muudaksid, mis on selle juhendi juures hästi ning lisada ka muid kommentaare või ettepanekuid seoses vastava juhendiga. Saadud nõuandeid rakendati, kui need olid asjakohased ja aitasid juhendeid paremaks teha.

Selleks, et hinnata, kas hindamisjuhend on piisavalt üheselt mõistetav ning kas punktide jaotumise põhimõtted on mõistlikud, koostati eeskujuks olnud kontrolltöödele hindamis-juhendid lähtuvalt uuest ülesannete koostamise juhendist ning paluti kõigil kursuse praktikumijuhendajatel uue hindamisjuhendi abil hinnata, kas I kontrolltöö I ülesande, I kontrolltöö II ülesande, II kontrolltöö I ülesande või II kontrolltöö II ülesande ühte 2021/2022 sügissemestril esitatud lahendust, mis ei olnud ideaalselt lahendatud. Selleks saadeti kõigile eraldi vastav ülesanne, lahendus ja hindamisskeem. Saadud hindeid võrreldi ülesannete kaupa ja tulemuste põhjal hinnati juhendi üheselt mõistetavust ning punktide jaotumise põhimõtete mõistlikkust ning viidi vajalikud parandused programmeerimise osa koostamise juhendisse sisse.

3. Tulemused

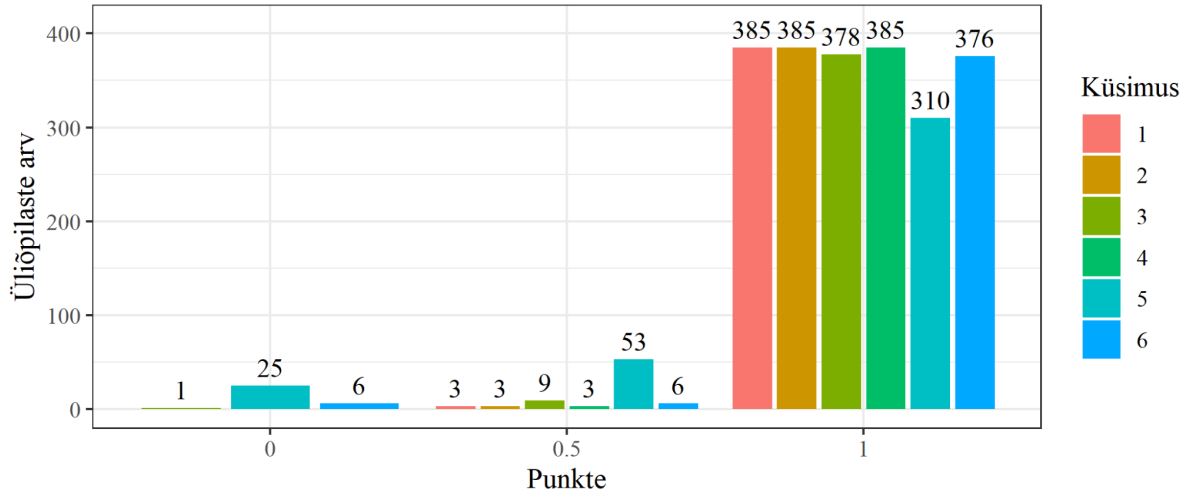
Selle peatüki esimeses pooles kirjutatakse, mis ilmnes kontrolltööde tulemuste ja ülesannete analüüsist, mille metoodikat on põhjalikult kirjeldatud eelmises peatükis, ning üliõpilaste kontrolltöödele antud tagasisidest. Teises pooles kirjeldatakse lähtuvalt esimese poole tulemustest ja ka kursuse õppejõududelt saadud tagasisidest, millised said kummagi kontrolltöö koostamise juhendid.

3.1 Kontrolltööde tulemuste, ülesannete ja tagasiside analüüs

3.1.1 I kontrolltöö analüüs

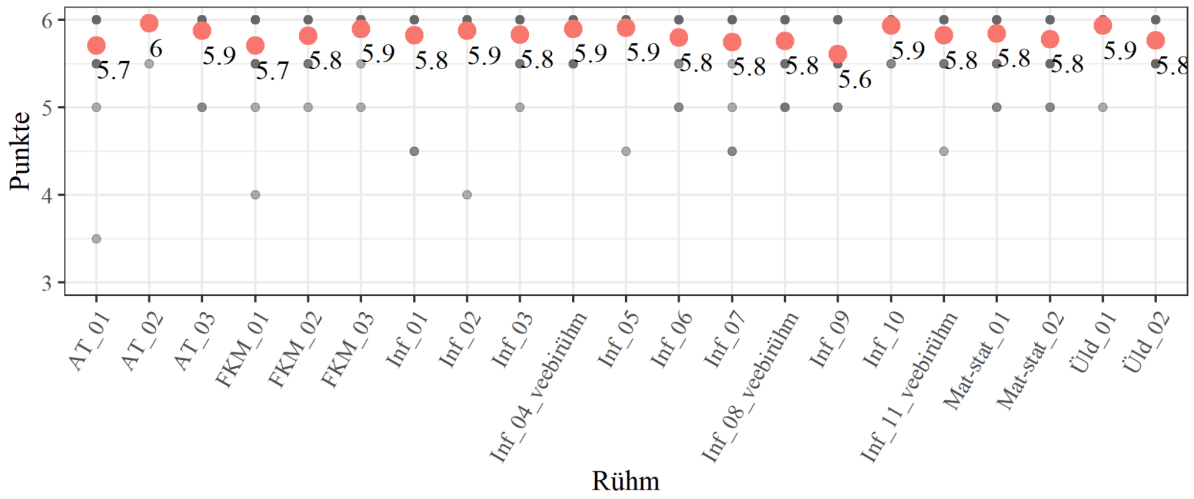
I kontrolltöö arvestuslikku osa lahendas kokku 362 üliõpilast. Kaks korda lahendas arvestuslikku osa 24 üliõpilast ning 3 korda 1 üliõpilane. Esimesel katsel said positiivse tulemuse ehk 5,5 või 6 punkti 331 üliõpilast ehk 91% kõigist lahendanutest. Lõpuks jäi 7 üliõpilase tulemuseks mittearvestatud.

Arvestuslikus osas eksiti enim 5. küsimusega (vt. joonis 1). Tegu oli küsimusega, kus paluti üliõpilasel kirjutada, mis järjekorras tsüklit sisaldava programmi ridasid täidetakse ning levinuim viga oli see, et vastuseks kirjutati ainult tsükli esimesel läbimisel täidetavate ridade numbrid, kuid selle eest võeti maha ainult 0,5 punkti, sest küsimuse püstitus võis olla mitmeti mõistetav.



Joonis 1. I kontrolltöö arvestusliku osa eest saadud punktide jaotumine küsimuste kaupa.

Praktikumirühmade lõikes jaotusid arvestusliku osa eest esimesel katsel saadud punktid nagu näha joonisel 2. Keskmiselt kõige kehvemini läks Inf_09 rühmal, kuid ükski selle rühma üliõpilane ei eksinud rohkem kui 1 punkti jagu. Väga hästi läks rühmadel AT_02, Inf_04_veebirühm, Inf_10 ja Üld_02, kus said kõik üliõpilased I katsel arvestusliku osa arvestatud. Kuid üldiselt on jaotused ja keskmised siiski väga sarnased ja sellest iseenesest järeldusi teha ei saa.



Joonis 2. I kontrolltöö arvestusliku osa eest saadud punktide jaotumine praktikumirühmade kaupa.

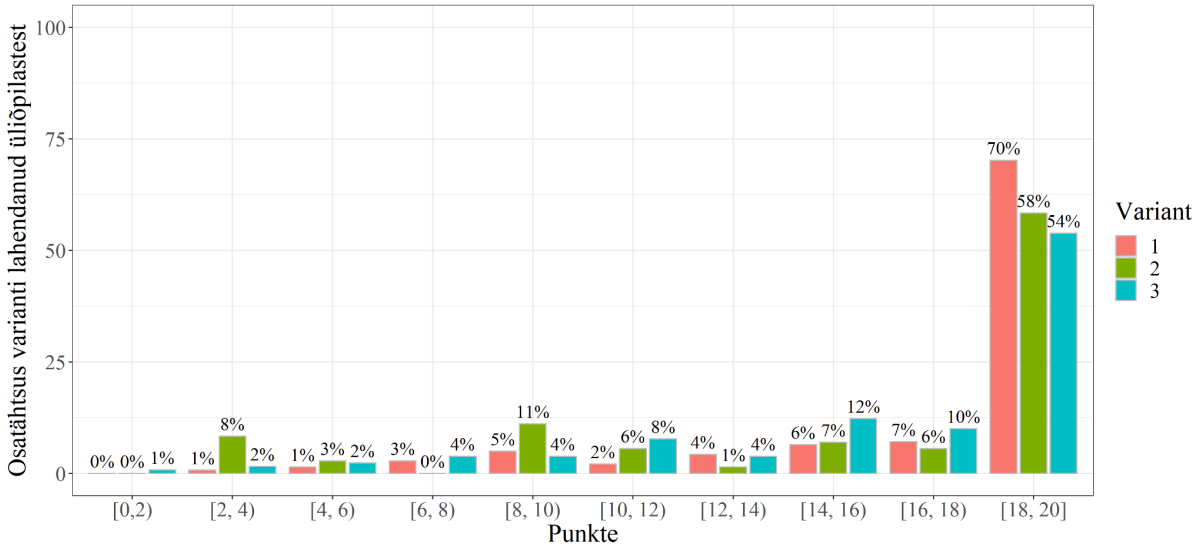
Kontrolltöö järel tagasisideküsitlusele vastanud 130 üliõpilastest 85,4% hindasid arvestusliku osa ülesandeid pigem lihtsaks, lihtsaks või väga lihtsaks. Üksikud üliõpilased viitasid ainult küsimuste sõnastuse mitmetimõistetavusele. Samuti toodi välja, et üliõpilastel ei olnud selge, kas arvestuslikus osas peab kõik küsimused täiesti õigesti vastama või on veidi eksimisruumi, mis võis tekitada neis lisapinget küsimustele vastates.

Nendel üliõpilastel, kes arvestuslikus osas ebaõnnestusid, olid ka programmeerimise osa eest saadud punktid pigem madalamad, kuid oli ka 3 üliõpilast, kes said programmeerimise osa eest vähemalt 90% punktidest. Nendel üliõpilastel, kes läbisid arvestusliku osa edukalt, ei olnud ka programmeerimise osa tulemused väga kehvad. Kuid 23 üliõpilast ehk 7% neist said programmeerimise osa eest vähem kui pooled punktid. Need tulemused on oodatavad, arvestades, et arvestusliku osa eesmärk on kontrollida õpiväljundite minimaalset omandamist ning programmeerimise osa eest saadud punktidel ei ole lävendit, mis tuleb ületada.

Neid, kes said I kontrolltöö arvestusliku osa tehtud, kuid ebaõnnestusid II kontrolltöö arvestuslikus osas, oli 19. Ning lisaks oli veel 19 üliõpilast, kes said I kontrolltöö arvestusliku

osa tehtud, kui kukkusid kursuse läbi, kellest ainult viiel jäi puudu punktidest, et eksamile pääseda, 9 ei ilmunud eksamile ning 5 said eksamil liiga vähe punkte aine läbimiseks. Seega võib arvata, et nii I kui ka II kontrolltöö arvestusliku osa ülesanded olid eesmärgile vastava tasemega.

Programmeerimise osa puhul paistab jooniselt 3, et kontrolltöö I varianti lahendanud üliõpilastel läks kõige paremini ning kontrolltöö II varianti lahendanud said enim alla 10 punkti. Kusjuures 12 üliõpilast II varianti lahendanutest, kes said alla 10 punkti, said II ülesande eest 0 punkti. I variandi puhul oli neid 1 ning III variandi puhul 5. Punktide jaotumise erinevust varianditi kinnitab ka Kruskal-Wallise test. Täpsemalt erineb oluliselt I variandi eest saadud punktide jaotumine teiste variantide eest saadud punktide jaotumistest.

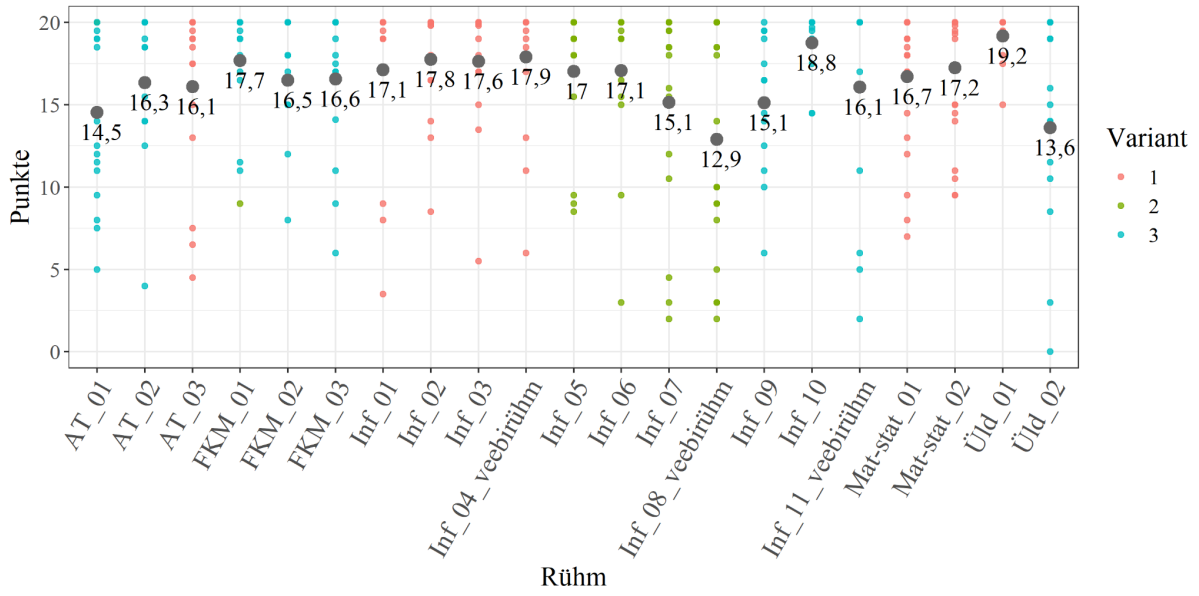


Joonis 3. I kontrolltöö programmeerimise osa eest saadud punktide jaotumine varianditi

Programmeerimise osa lahendasid üliõpilased iga variandi puhul keskmiselt sama ajaga. I ja III variandi puhul esitati esimesed lahendused ~ 35 minutit pärast programmeerimise osa lahendamise alustamist ning II variandi puhul ~ 45 minutit pärast lahendamise alustamist. See erinevus, aga ei ole nii suur, et sellest saaks midagi järeldada.

Praktikumirühmade lõikes jaotusid programmeerimise osa eest saadud punktid nagu näha joonisel 4. On näha, et rühmadel, kes lahendasid I varianti, läks hästi. II varianti lahendanud rühmadest läks pooltel tegelikult sama hästi kui enamikel I varianti lahendanutel, kuid teisel kahel rühmal veidi kehvemini, eriti Inf_08_veebirühmal. Samas Inf_08_veebirühma üliõpilaste arvestusliku osa tulemused ei erinenud nõnda märgatavalt teistest, seega ei ole selge, millest on tingitud selle rühma keskmisest madalam programmeerimise osa tulemus.

III varianti lahendanud rühmadest, läks Inf_10 rühmal väga hästi, mis on vastavuses ka arvestusliku osa tulemustega ning ülejäänud rühmade tulemused on võrreldavad II varianti lahendanud rühmade omadega.



Joonis 4. I kontrolltöö programmeerimise osa eest saadud punktide jaotumine praktikumirühmade kaupa.

Järeltööga proovis oma programmeerimise osa tulemust parandada 42 üliõpilast. Nendest kolmel punktide arv hoopis vähenes, kuid mitte rohkem kui 1,5 punkti võrra. Ülejäänutest pea kõik, kes said enne alla 10 punkti, said järeltööl siiski rohkem kui 10 punkti, välja arvatud 2 tudengit, kellel tõusis punktide arv 2-lt 7-le ja 6,5-lt 9-le. Arvestades, et järeltöödel oli mitu neid, kellel punktide arv vähenes, jäi ligikaudu samaks või tõusis ainult paari punkti võrra, kuid saadi ka väga häid tulemusi, võib arvata, et järeltööde ülesanded ei olnud liiga lihtsad ega ka rasked.

Pärast kontrolltööd hindasid 61,5% 130-st tagasiside küsimustikule vastanud üliõpilasest programmeerimise osa ülesanded pigem lihtsaks, lihtsaks või väga lihtsaks, ning ülejäänud pigem keeruliseks, keeruliseks või väga keeruliseks, kusjuures väga keeruline oli töö ainult 1,5% vastanute arvates. Tähelepanuväärne on ka see, et 39,5% vastanutest märkisid, et tunnevad end ebakindlalt just failide lugemisel ja kirjutamisel. Eraldi toodi veel välja, et ülesannete sõnastus ei olnud kõigi jaoks arusaadav või tekitas segadust.

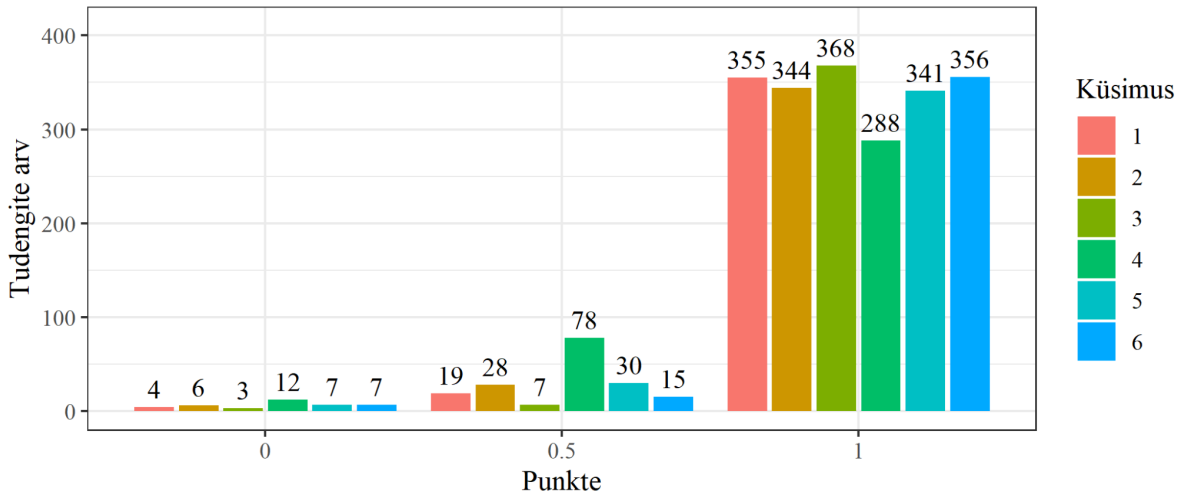
Programmeerimise osa erinevate variantide ülesandeid omavahel võrreldes paistis välja sama, mis tulemuste analüüsist – I variandi ülesanded olid kõige lihtsamad. Esimese ülesande puhul

avaldus I variandi lihtsus selles, et ülesande teema oli kõige triviaalsem ning lihtsamini hoomatavam ehk selle tõttu ka arvutused, mis pidi muutujatel läbi viima, olid kõige lihtsamad. Kusjuures ka muutujaid oli kõige vähem. Teise ülesande puhul oli näha, et II ja III variandi ülesanded olid justkui I variandi ülesande edasiarendused, sest kõike, mida pidi tegema I variandi II ülesandes, pidi tegema ka teistes variantides, kuid teistes variantides oli veel mingeid tehteid või nüansse juures, mis tegid need keerulisemaks või vähemalt ajamahukamaks. II ja III variandi ülesanded olid küll erinevad, kuid ülesannete raskusaste tundus kokkuvõtlikult olevat sarnane.

3.1.2 II kontrolltöö analüüs

II kontrolltöö arvestuslikku osa lahendas kokku 330 üliõpilast. Kaks korda lahendas arvestuslikku osa 44 üliõpilast, 3 korda 2 üliõpilast ning 4 korda üks üliõpilane. Esimesel katsel said positiivse tulemuse ehk 5,5 või 6 punkti 284 üliõpilast ehk 86% kõigist lahendanutest. Lõpuks jäi 3 üliõpilase tulemuseks mittearvestatud.

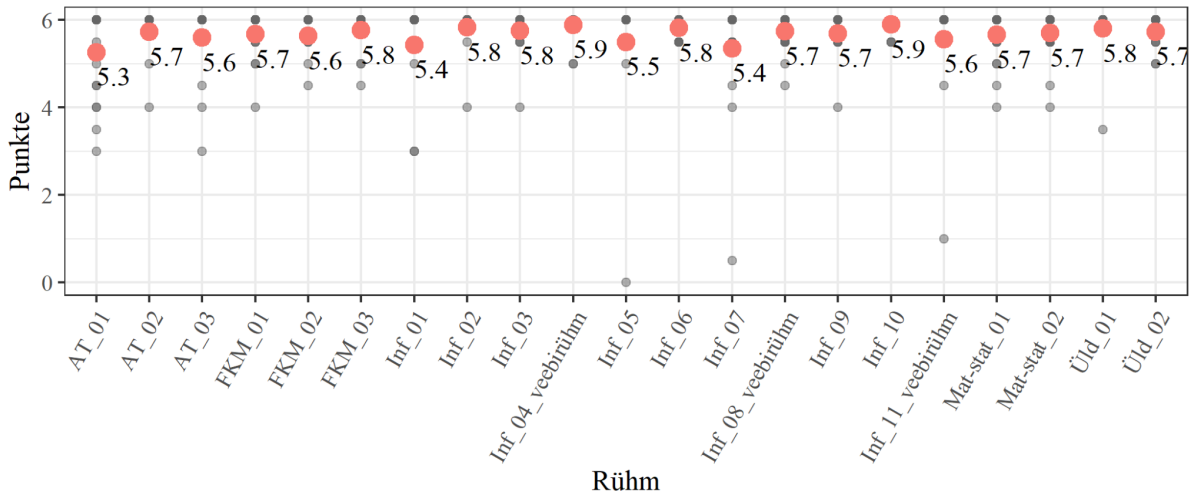
Arvestuslikus osas eksiti enim 4. küsimusega nagu näha joonisel 5. Tegu oli küsimusega, kus üliõpilane pidi leidma, mitu korda läbitakse kahekordse tsükli sisemise tsükli keha ning muutuja väärtused tsüklite esimestel läbimistel või kus üliõpilane pidi täitma kahemõõtmelist andmestruktuuri läbiva programmilõigu lüngad.



Joonis 5. II kontrolltöö arvestusliku osa eest saadud punktide jaotumine küsimuste kaupa.

Praktikumirühmade lõikes jaotusid arvestusliku osa eest esimesel katsel saadud punktid nagu näha joonisel 6. Keskmiselt kõige kehvemini läks AT_01 rühmal. Väga hästi läks rühmadel

Inf_06 ja Inf_10, kus said kõik üliõpilased I katsel arvestusliku osa arvestatud. Palju on rühmi, kus ebaõnnestusid ainult üksikud üliõpilased. Võrreldes I kontrolltöö arvestusliku osaga ei ole jaotused samasugused, seega ei pruugi konkreetse rühma puhul punktide säärane jaotumine olla tingitud just sellest, millises praktikumirühmas nad õpivad.



Joonis 6. II kontrolltöö arvestusliku osa eest saadud punktide jaotumine praktikumirühmade kaupa.

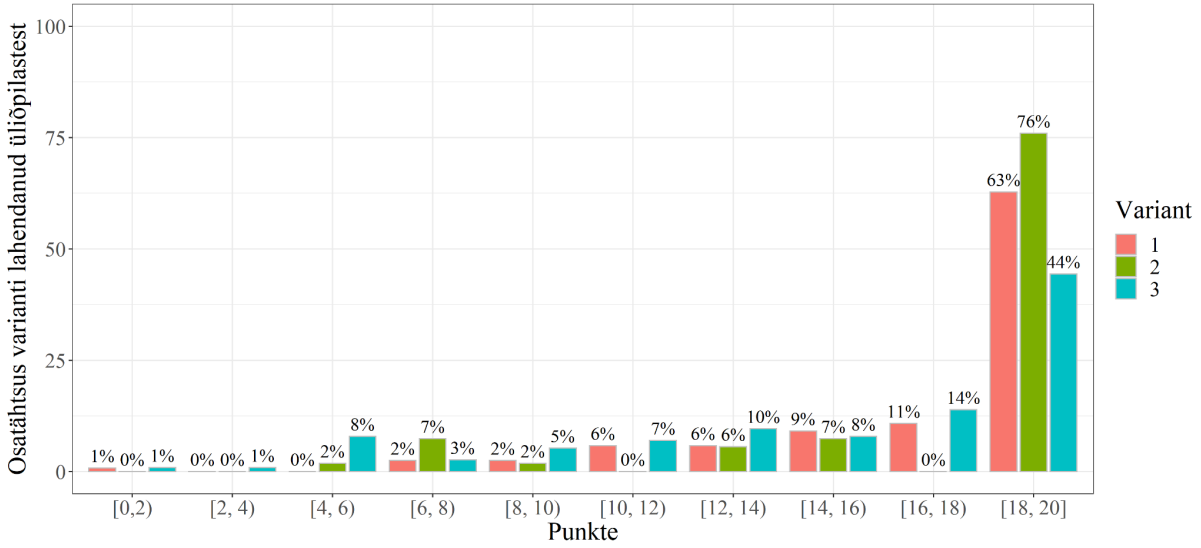
Pärast kontrolltööd tagasisideküsitlusele vastanud 66-st üliõpilasest arvas 42%, et küsimused olid pigem keerulised, keerulised või väga keerulised ning umbes sama palju ehk 41% arvas, et arvestuslik osa oli pigem lihtne. Seega neid, kes arvasid, et küsimused olid lihtsad või väga lihtsad, oli vähemuses, mis tähendab, et küsimused olid keerulisemad, kui I kontrolltöö puhul, samas olid ka teemad keerulisemad ning ka üliõpilaste üleüldine õppekoormus tõenäoliselt suurem. Kuid toodi välja, et küsimuste sõnastus oli I kontrolltöö arvestusliku osaga võrreldes selgem.

II kontrolltöö puhul on arvestuslikus osas ebaõnnestunud üliõpilaste programmeerimise osa eest saadud punktide jaotus võrdlemisi sarnane I kontrolltöö omale, samamoodi ka arvestuslikus osas õnnestunud üliõpilaste puhul. Suurim erinevus on see, et II kontrolltööl ei olnud üliõpilasi, kes oleksid saanud arvestuse, kuid programmeerimise osa eest vähem kui 5 punkti. Seega ei ole ka siin alust arvata, et arvestusliku osa ülesanded oleksid olnud liiga lihtsad või liiga keerulised.

II kontrolltöö programmeerimise osa puhul selgus kontrolltööd läbi viies, et I ülesanne oli liiga ajamahukas ning paljud üliõpilased jäid ajahätta. Selleks, et seda kompenseerida, muudeti hindamismeetodit ehk I ülesande eest oli võimalik kokku saada, kas I ja II variandi

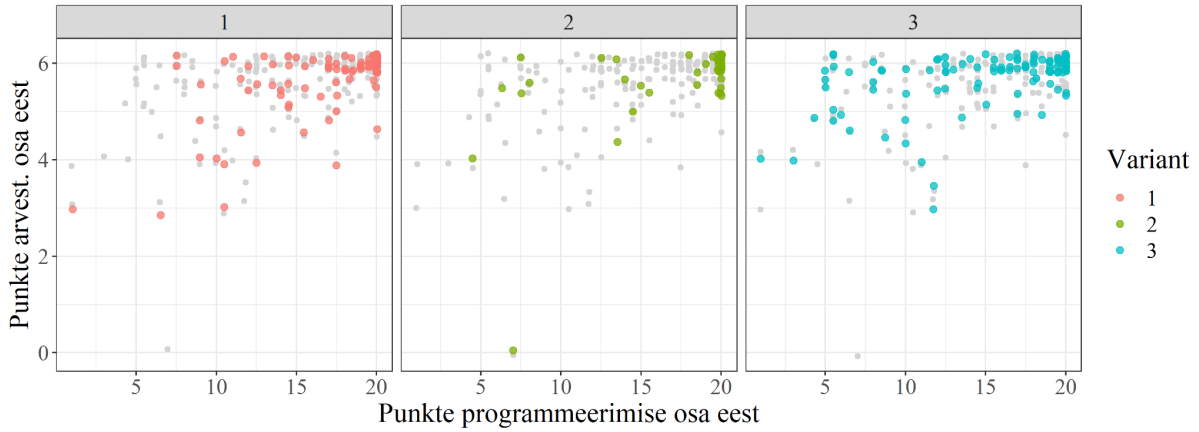
puhul 12 punkti või III variandi puhul 12,5 punkti, kuid arvesse läks üliõpilastel siiski 10 punkti, isegi kui teeniti üle 10 punkti. Kuid üliõpilased lahendamise ajal seda ei teadnud. Seega mõjus see nende üliõpilaste tulemustele negatiivselt, kes nägid palju vaeva ja lahendasid 1. ülesande suurepäraselt, kuid selle arvelt said vähem aega 2. ülesande lahendamiseks – nad said 1. ülesande eest sama palju punkte, kui need, kes tegid vigu või jätsid midagi kuni 2 või 2,5 punkti jagu lahendamata.

Joonisel 7 on näha programmeerimise osa eest saadud punktide jaotumine varianditi. Paistab silma, et III varianti lahendanute hulgas oli kõige vähem üliõpilasi, kes said 18 või rohkem punkti ning kõige rohkem neid, kes said alla 10 punkti. Seda, et III variandi tulemuste jaotumine erineb oluliselt I ja II variandi omast, kinnitab ka Kruskal-Wallise test. Kusjuures I ülesande puhul on jaotused küllaltki sarnased, kuid erinevus tuleb sisse II ülesande juures, kus III variandi puhul said 30 üliõpilast 115-st alla 5 punkti, sealhulgas 14 mitte ühtegi punkti.



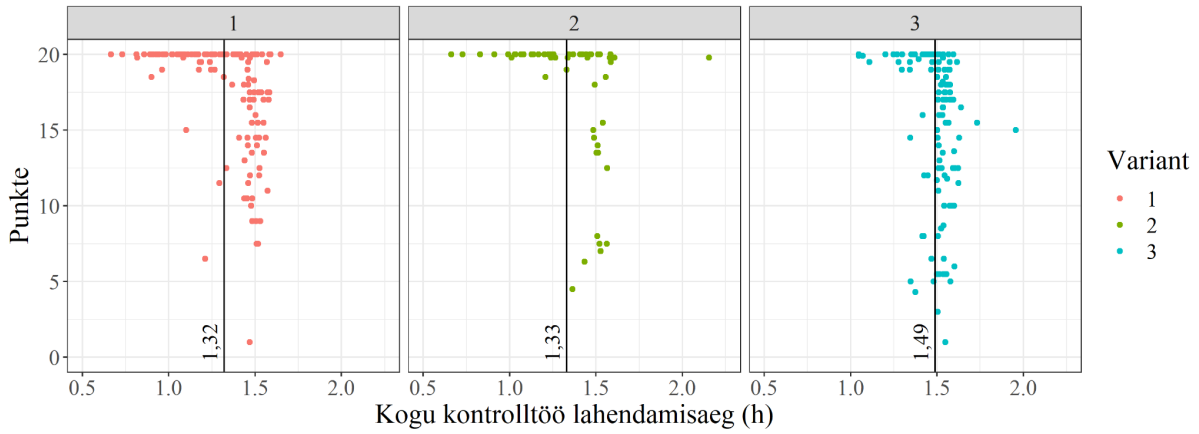
Joonis 7. II kontrolltöö programmeerimise osa eest saadud punktide jaotumine varianditi.

Samas ei olnud III varianti lahendanud üliõpilaste puhul ilmtingimata tegu nõrgemate üliõpilastega, sest nagu ka jooniselt 8 näha, oli nende hulgas, kes said programmeerimise osa eest 5–10 punkti, arvukalt ka neid, kellel läks arvestuslik osa hästi. Teiste variantide puhul oli sellised üliõpilasi vähem. Seega on tõenäoline, et III variandi ülesanded, või vähemalt I ülesanne, olid keerulisemad kui teiste variantide ülesanded.



Joonis 8. II kontrolltöö arvestusliku ja programmeerimise osa eest saadud punktide vaheline seos varianditi.

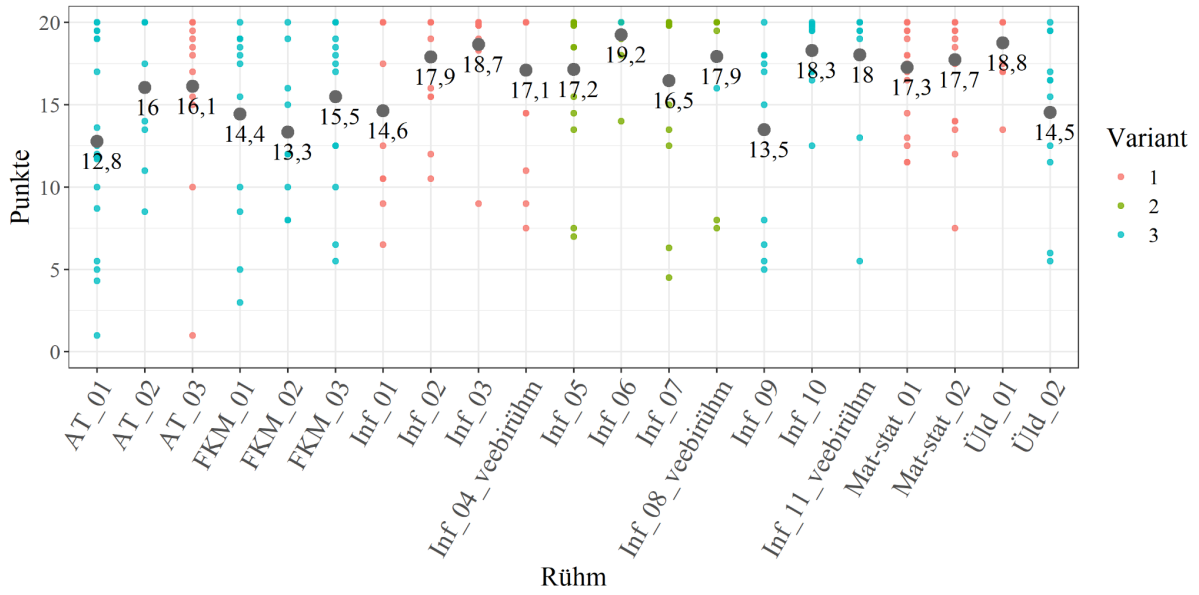
II kontrolltöö lahendamisaegade puhul aga esineb variantide vahel erinevusi. Joonisel 9 on näha, et I ja II variandi ülesandeid lahendanud üliõpilastest kiiremad said töö valmis ligikaudu 45 minutiga. III varinadi puhul aga läks ka kõige kiirematel oskajatel kogu kontrolltöö lahendamisega kauem kui 1 tund.



Joonis 9. II kontrolltöö programmeerimise osa eest saadud punktide ja kontrolltöö lahendamisaja vaheline seos varianditi.

Vaadates programmeerimise osa eest saadud punktide jaotumist rühmade kaupa jooniselt 10, on näha, et II varianti lahendanud rühmadel läks kõigil üsna sarnaselt ning pigem hästi. I varianti lahendanutest läks ka enamikel hästi, veidi madalam on Inf_01 rühma tulemus, kuid nagu oli näha joonisel 6, sai see rühm ka arvestusliku osa eest keskmiselt vähem punkte. III varianti lahendanud rühmade hulgas on nii rühmasid, kellel läks suurepäraselt, rühmasid, kellel läks hästi, kui ka neid, kellel ei läinud nii hästi. Samas ei olnud programmeerimise osa eest keskmisest kehvema tulemuse saanud rühmadel arvestusliku osa tulemused küll

keskmisest kehvemad. Seega on võimalik, et III variandi programmeerimise osa ülesanded olid teistest keerulisemad.



Joonis 10. II kontrolltöö programmeerimise osa eest saadud punktide jaotumine praktikumirühmade kaupa

Järeltööga proovis oma programmeerimise osa tulemust parandada 36 üliõpilast. Nendest viiel punktide arv hoopis vähenes 0,5 kuni 4 punkti võrra. Ülejäänutest oli 6 neid, kellel ei õnnestunud saada vähemalt 10 punkti. Kuid lausa üheksal õnnestus saada 18 või rohkem punkti. Seega arvestades, et järeltöödel oli mitu neid, kellel punktide arv vähenes, jäi ligikaudu samaks või tõusis ainult paari punkti võrra, kuid saadi ka väga häid tulemusi, võib arvata, et järeltööde ülesanded ei olnud liiga lihtsad ega ka rasked.

Kontrolltöö järel 66-lt üliõpilastelt saadud tagasisidest tuli välja, et suurem osa ehk 61% vastanutest, hindasid programmeerimise osa ülesandeid pigem lihtsaks, lihtsaks või väga lihtsaks. Kuid varianditi see jaotus erines: I ja II varianti lahendanutest oli 75% või enam neid, kes arvasid, et ülesanded olid pigem lihtsad või veelgi lihtsamad, kuid III varianti lahendanutest arvasid 45% vastanutest, et need ülesanded olid pigem lihtsad või lihtsad. Pea kõik III varianti lahendanud tagasiside küsitlusele vastanutest tõid välja, et III variandi ülesanded olid nende jaoks ülesande teksti killustatuse ja mahu tõttu raskesti arusaadavad ja neid oli raske etteantud aja jooksul lahendada.

Programmeerimise osa erinevate variantide ülesandeid omavahel võrreldes oli näha, et III variandi ülesannete teemad olid kõige spetsiifilisematel teemadel, mis ilmselt raskendas

ülesande tekstist aru saamist. I ülesanne oli kõige lihtsam II variandil ning kõige mahukam III variandil. II ülesanne oli samuti kõige lihtsam II variandil. III variandi II ülesanne oli väga erinev I ja II variandist ning seal küll puudus failist lugemine, kuid tuli defineerida, erinevalt I ja II variandist, 3 funktsiooni, mis olid tunduvalt mahukamad kui I ja II variandi omad.

3.2 Kontrolltööde koostamise juhendid

3.2.1 I kontrolltöö juhend

I kontrolltöö koostamise juhend koosneb eraldi arvestusliku osa (vt lisa I) ja programmeerimise osa koostamise juhendist (vt lisa II). Arvestusliku osa koostamise juhendis on esmalt toodud välja, mis teemasid peavad 6 arvestusliku osa küsimust katma. Teemad ja nende järjekord on fikseeritud lähtuvalt 2022/2023 õppeaasta sügissemestril plaanitavast teemade läbimise järjekorrast. Seejärel on kirjas suunised, millest lähtuda kontrolltööks küsimusi Moodle'i küsimustepangast valides ja valmis pannes. Lähtuvalt 2021/2022 õppeaasta sügissemestri tulemustest ja üliõpilaste tagasisidest on eraldi välja toodud, et küsimusi valides tuleb kindlasti kontrollida, et need oleksid üheselt mõistetavad ning ühe teema küsimused sarnase raskusastmega. Viimaks on kirjeldatud hindamise põhimõtteid, mis olid samad ka 2021/2022 õppeaastal.

I kontrolltöö programmeerimise osa ülesannete koostamise juhend on jaotatud viieks osaks:

1. üldised suunised ülesannete koostamise protsessiks ja ülesannete ülesehituseks,
2. I ülesande kohustuslikud komponendid ja punktide jaotumise põhimõtted,
3. I ülesande näide koos vastava hindamisjuhendiga,
4. II ülesande kohustuslikud komponendid ja punktide jaotumise põhimõtted,
5. II ülesande näide koos vastava hindamisjuhendiga.

Üldiste suuniste juures on esmalt kirjas, et ülesanded peavad koosnema probleemi püstitusest, konkreetsete funktsioonide ja põhiprogrammi töö kirjeldusest ning kindlasti peavad juures olema ka näited sisendiga ja väljundiga/tagastusega, et lühikese aja jooksul üliõpilased võimalikult kiiresti aru saaks, mida programm tegema peab. Tähelepanu on juhitud ka sellele, et ülesannete koostajad jälgiks, et erinevate variantide ülesannetes ei oleks sama tüüpi arvutused, kuid ülesannete raskusaste oleks sama. Kusjuures ka sisendi-väljundi näited peaksid olema samal tasemel abistavad. Sarnaselt arvestuslikule osale, tuleb ka neid ülesandeid koostades suurt tähelepanu pöörata sellele, et ülesannete tekstid oleksid üheselt

mõistetavad ega eeldaks mingi muu eriala spetsiifilisi teadmisi, sest fookus peab olema siiski programmeerimisel. Selle tagamiseks on toodud ka nõue, et kui ülesanded valmis, peavad teised õppejõud need enne kontrolltöö toimumist nii sisuliselt kui ka keeleliselt üle kontrollima. Ning, kuna ülesandeid hindavad tõenäoliselt erinevad õppejõud, peab hinnete ühtluse tagamiseks ülesannetega kaasa andma teistele õppejõududele detailsed hindamisjuhendid koos näidislahendustega.

Ülesannete nõudeid ja punktide jaotumise põhimõtteid koondavas osas võeti eeskujuks 2021/2022 sügissemestri I kontrolltöö II variant, sest tulemuste ja ülesannete analüüsi põhjal tundus see kõige paremini kõiki nõutud teadmisi kontrollivat. Selle variandi ülesanded lisati juhendi juurde ka näidetena, kuid parandati ülesannete sõnastust. I variandi eest saadi küll kõige rohkem positiivseid tulemusi, kuid selle ülesanded olid tunduvalt lihtsama ülesehitusega, seega poleks selle põhjal koostatud juhend olnud piisavalt universaalne. II ja III variant olid hoolimata tulemuste erinevast jaotumisest tegelikult väga sarnased, kuid III variandi ülesannete sisendi ja väljundi näited olid veidi keerukamad, seega sobis II variant eeskujuks ja juhendi koostamise näiteks paremini.

Kummagi ülesande kohustuslikke komponente fikseerides lähtuti sellest, et kahe ülesande peale kaetaks kõik teemad, mis peavad olema omandatud ning et esimene ülesanne oleks veidi lihtsamate teemade peale, kui teine ülesanne, et üliõpilased ei kaotaks asjata punkte esimesse ülesandesse kinni jäädes. Juhendi järgi ülesandeid koostades peaksid erinevad variandid ülesehituselt tulema sarnased, sest 70–80% ulatuses on fikseeritud, mida ülesanne peab sisaldama ning kui palju mingi komponent punkte peab andma. Variantide erinevus tuleneb seega teemast, sisendist, kasutatavatest andmetüüpidest, nõutud arvutustest (sh aritmeetilised tehted, tingimuslaused) ja väljundist. Lisaks sellele, kui palju mingi komponent punkte peaks andma, on toodud ka põhimõtted, mille järgi selle komponendi eest punkte maha võtta, et kontrolltööle üheselt mõistetava hindamisjuhendi koostamine oleks võimalikult lihtne, aga ka ülesande koostamisel oleks selgem, mida tuleb üliõpilastelt nõuda. Suure tõenäosusega on juhendis mõne komponendi hindamisega seoses veel aspekte, mille peale töö autor ei tulnud ning mis selguvad kontrolltööde koostamise või hindamise käigus lähtuvalt konkreetsetest ülesannetest ning mille põhjal on võimalik kursuse õppejõududel kontrolltööde koostamise juhendeid täiendada.

Kontrolltööde koostamise juhendite esialgsetele versioonidele andsid kaks kursuse õppejõudu ka vabas vormis tagasisidet. I kontrolltöö arvestuslikule osale antud konstruktiivse tagasiside

põhjal täpsustati juhendis, kuidas valitakse küsimused töösse ning parandati detaile teemade nimekirjas. Positiivse poole pealt tõid õppejõud välja, et juhend on ülevaatlik ning kõik olulised teemad on kaetud.

I kontrolltöö programmeerimise osale antud tagasiside põhjal parandati juhendi tekstis olevad vormistusvead, üldiste suuniste osas rõhutati olulisi asju üle ning täiendati punktide jaotumise osa arvestades juhte, mille peale töö autor esialgu ei tulnud. Juhendi juures meeldis õppejõududele juhendi universaalsus, mistõttu sobib see hästi objektiivseks aluseks erinevatel aastatel ülesandeid koostades, aga ka juhendi detailsus, eriti punktide jaotumise osas.

Juhendi punktide jaotumise osa rakendatavuse ja üheselt mõistetavuse hindamiseks läbi viidud ülesannete näidishindamisest selgus kummagi ülesande puhul kaks komponenti, mille hindamisel oli võimalik juhendit kaheti tõlgendada. Need ning ka taolised kohad II kontrolltöö juhendis täpsustati. II ülesande puhul tekkis ka olukord, kus kaks hindajat viiest ei märganud ühte viga lahenduses, seega lisati juhenditesse ka punkt, mis peaks tagama selle, et sellised vead ei jää töö hindajal kahe silma vahele. Kokkuvõttes erinesid I ülesande hindajate, keda oli 3, antud punktid maksimaalselt 0,5 punkti võrra ning erinevus oli tingitud ühe komponendi (6-st) erinevalt hindamisest. II ülesande viiest hindajast 2 andsid 9 punkti ning ülejäänud kolm 7,5 punkti. Need, kes andsid 9 punkti ei märganud lahenduses eelmainitud viga. Kuid lisaks sellele komponendile olid hindajad veel 2 komponendi hindamist (11-st) erinevalt tõlgendanud.

3.2.2 II kontrolltöö juhend

II kontrolltöö koostamise juhend koosneb samuti arvestusliku osa (vt lisa III) ja programmeerimise osa (vt lisa IV) koostamise juhenditest. II kontrolltöö arvestusliku osa koostamise juhend on väga sarnane I kontrolltöö arvestusliku osa koostamisele juhendile, sest arvestusliku osa struktuur ja põhimõte on sama. Ainus erinevus on teemade nimekiri, mida arvestuslikus osas kontrollitakse. Kusjuures II kontrolltöö arvestuslikus osas kontrollitakse ainult neid teadmisi, mis omandatakse peale I kontrolltöö toimumist.

II kontrolltöö programmeerimise osa ülesannete koostamise juhend on sarnaselt I kontrolltöö programmeerimise osa juhendile jaotatud viieks osaks:

1. üldised suunised ülesannete koostamise protsessiks ja ülesannete ülesehituseks,
2. I ülesande kohustuslikud komponendid ja punktide jaotumise põhimõtted,

3. I ülesande näide koos vastava hindamisjuhendiga,
4. II ülesande kohustuslikud komponendid ja punktide jaotumise põhimõtted,
5. II ülesande näide koos vastava hindamisjuhendiga.

Juhendi I osa on mõlema kontrolltöö juhendi puhul sama, sest üldised detailid kontrolltööde puhul ei erine.

II kontrolltöö ülesannete nõudeid ja punktide jaotumise põhimõtteid koondavas osas võeti eeskujuks 2021/2022 sügissemestri II kontrolltöö I variant, sest kontrolltöö tulemuste ja ka ülesannete sisulisest analüüsist paistis, et II variant oli kolmest kindlasti kõige lihtsam ning III variandi puhul oli üliõpilaste raskusi, mille taga ei paistnud ükski muu faktor olevat, mistõttu tundus I variant olevat hea suunis universaalse juhendi koostamiseks.

Juhendi koostamise põhimõtted olid samad, mis I kontrolltöö puhul – ülesannete kohustuslikke komponente fikseerides lähtuti sellest, et kaks ülesannet kataksid kõik kontrollitavad teemad, et esimene ülesanne oleks veidi lihtsamate teemade peale, kui teine ülesanne. Selle kontrolltöö puhul tagavad variantide erinevused ülesande teema, sisend, kasutatavad andmestruktuurid, nõutud arvutused (sh aritmeetilised tehted, tehted ja operatsioonid andmestruktuuridel) ja väljund. Nagu ka I kontrolltöö puhul on ka selle kontrolltöö koostamise juhendis suure tõenäosusega mõne komponendi hindamisega seoses veel aspekte, mille peale töö autor ei tulnud ning mis selguvad kontrolltööde koostamise või hindamise käigus lähtuvalt konkreetsetest ülesannetest ning mille põhjal on võimalik kursuse õppejõududel kontrolltööde koostamise juhendeid täiendada.

Kursuse õppejõudude II kontrolltöö arvestuslikule osale antud tagasiside oli sarnane I kontrolltöö arvestusliku osa omale ning selle põhjal viidigi sisse samad muudatused, mis I kontrolltöö arvestusliku osa juhendisse, ning lisaks täpsustati ka teemade nimekirja, et see vastaks kindlasti kursusel käsitletavatele teemadele. Toodi välja ka idee ühendada mõlema kontrolltöö arvestusliku osa koostamise juhendid, sest ainus erinevus ongi teemade nimekiri. Kui kursust läbi viies osutub mugavamaks neid dokumente koos hoida, on kursuse õppejõududel alati võimalik seda teha, kuid võib ka osutuda, et on mugavam, et erinevate kontrolltöödega seotud dokumendid on eraldi ja sellisel juhul on mugavam ka kui arvestusliku osa koostamise juhendid on eraldi.

II kontrolltöö programmeerimise osa koostamise juhendile antud tagasiside kattus samuti suures osas I kontrolltöö programmeerimise osale antud tagasisidele. Eraldi toodi välja, et juhendis peaks andma rohkem vabadust sellele, mitu funktsiooni on ülesandes vaja luua,

mille kohane muudatus viidi juhendisse ka sisse. Samuti käidi välja idee programmeerimise osa juhendite ühendamise kohta, kuid kuna juba ühe juhendi maht on väga suur ning ainus ühisosa on üldiste suuniste osa, ei tundunud töö autorile nende juhendite ühendamine otstarbekas, kuid kursuse õppejõududel on alati võimalus seda teha, kui see nende jaoks otstarbekamaks osutub.

Juhendi punktide jaotumise osa hindamiseks läbi viidud ülesannete näidishindamisest selgus selle kontrolltöö puhul I ülesande juures 2 ja II ülesande juures 1 komponent, mille hindamisel oli võimalik juhendit kaheti tõlgendada. Need ning ka taolised kohad I kontrolltöö juhendis täpsustati. Kokkuvõttes erinesid I ülesande hindajate, keda oli 2, antud punktid maksimaalselt 0,75 punkti võrra ning erinevus oli tingitud kahe komponendi (13-st) erinevalt hindamisest. II ülesannet hindasid 3 praktikumijuhendajat ning nendest 2 hindasid ühtemoodi ning kolmas andis 1 punkti rohkem. Erinevus oli tingitud ühe komponendi (11-st) erinevalt tõlgendamises. Kõigi näidishindamises selgunud mitmetimõistetavuste põhjus oli selles, et töö autor ei tulnud ise kõigi erinevate lahendusvõimaluste peale ja seega ei osanud koostatud hindamisjuhendisse kõiki erijuhte täpsustada.

4. Arutelu

Selles peatükis on toodud välja töö autori soovitused juhendite rakendamiseks ning ideed võimalikeks töö edasiarendusteks.

Juhendite rakendamisega seoses kursuse 2022/2023 õppeaasta sügissemestril on töö autoril kaks soovitust. Esiteks pakub töö autor välja, et loodud juhendite abil võiksid kursuse alguses kõik praktikumijuhendajad luua ühe variandi I kontrolltööst ning peale I kontrolltöö toimumist kohe ka ühe variandi II kontrolltööst. Nii on loodud variante piisavalt palju selleks, et nende hulgast valida parimad nii põhitöö kui ka järeltöö jaoks. Samuti on piisavalt aega nende viimistlemiseks ja ühtlustamiseks. Kindlasti leiaksid variandid, mida ei võetud kasutusele, rakendust järgmisel aastal või saaks I kontrolltöö kasutamata variantide teemasid ära kasutada II kontrolltöö ülesannete loomisel. Lisaks on soovituslik kursusel tööde hindamiseks kasutada Moodle'i hindamismaatriksi funktsiooni, mis võimaldab üliõpilastel näha, kui palju ja mis põhjusel nad iga komponendi eest punkte said. See muudab kontrolltöö tagasiside üliõpilaste jaoks konstruktiivsemaks ning hindamise läbipaistvamaks.

Kui töö raames loodud juhendid kontrolltööde koostamiseks osutuvad mugavaks abivahendiks kontrolltööde koostamisel, oleks mõistlik taoline juhend koostada ka kursuse eksami koostamiseks. Praegu seda ei loodud, sest eksami teemad ja struktuur ei olnud juhendite koostamise ajaks veel piisavalt selged.

Arvestades, et loodud kontrolltööde koostamise juhendites on väga detailselt lahti kirjutatud ülesannete hindamise põhimõtted, on võimalik töö edasiarendus ka kontrolltööde automaatse hindamise süsteemi väljaarendamine. Nii sõltuks kontrolltöö eest saadud hinne veel vähem konkreetsest hindajast, kuid kindlasti on ülesannete automaatsel hindamisel ka oma väljakutsed ja kitsaskohad.

Tulevikus võiks ka luua rakenduse, mis võimaldab võimalikult väikese vaevaga kursuse kontrolltööde tulemusi visualiseerida. Nii oleks võimalik kursuse õppejõududel vahetult saada tagasisidet kontrolltööde kvaliteedi ja üliõpilaste edenemise kohta. Töö autoril valmis paralleelselt selle tööga ka veebirakendus, kus on võimalik näha 2021/2022 sügissemestri kontrolltööde soovitud tulemusi iseloomustavaid graafikuid[2], kuid see eeldab, et andmed on kindlal kujul, mille loomine võtab küllalt palju aega.

[2] https://0k4q6w-triinuaug.shinyapps.io/prog_kt/

Kokkuvõte

Bakalaureusetöö käigus valmisid Tartu Ülikooli kursuse „Programmeerimine“ jaoks juhendid õppejõududele kontrolltööde koostamiseks ja hindamiseks, et kahe kokkuvõtliku hindamise eesmärki täitva kontrolltöö erinevate variantide koostamine oleks õppejõudude jaoks üheselt arusaadav ja mugav ning oleks tagatud erinevate variantide raskusastmete võrdsus ja lahenduste ühtlane hindamine.

Esmalt tutvuti erinevate viisidega programmeerimise teadmiste kontrollimiseks ning saadi väga põhjalik ülevaate kursuse „Programmeerimine“ sisust ja senisest kontrolltööde läbiviimisest. Seejärel analüüsiti kursuse kontrolltööde tulemusi, kontrolltöödele antud tagasisidet ja kontrolltööde ülesandeid. Analüüsist leitu põhjal koostati juhendid kontrolltööde koostamiseks, läbiviimiseks ja hindamiseks. Juhendite mustanditele küsiti tagasisidet kursuse kogenumatelt õppejõududelt ja viidi läbi näidishindamine loodule tagasiside saamiseks. Lõpuks valmisid universaalsed ja detailsed juhendid, mis sobivad hästi objektiivseks aluseks ülesannete koostamisel ning vähendavad kontrolltööde hindamisel hindamisjuhendi mitmetimõistetavusest tingitud erinevusi.

Juhendid loodi kummagi kontrolltöö arvestuslikule osale ja programmeerimise osale. Arvestuslike osade juhendites on toodud välja, mis teemasid peavad küsimused katma, suunised küsimuste loomiseks ja tööks valmis panemiseks ning hindamise põhimõtted. Programmeerimise osa juhendites on kirjas üldised suunised ülesannete koostamise protsessiks ja ülesannete ülesehituseks ning kahe ülesande kohustuslikud komponendid, punktide jaotumise põhimõtted ja näited koos vastavate hindamisjuhenditega.

Suure tõenäosusega on juhendis mõne komponendi hindamisega seoses veel aspekte, mille peale töö autor ei tulnud ning mis selguvad kontrolltööde koostamise või hindamise käigus lähtuvalt konkreetsetest ülesannetest ning mille põhjal on võimalik kursuse õppejõududel kontrolltööde koostamise juhendeid täiendada.

Viidatud kirjandus

[1] Rootalu, K. Kui suurt palka teenitakse Eesti kõige nõutumatel ametikohtadel? *Statistikaameti blogi*, 2021. https://www.stat.ee/et/uudised/kui-suurt-palka-teenitakse-eesti-koige-noutumatel-ametikohtadel (06.02.2021)

[2] Eickelmann, N. ACM Fellow Profile: David Lorge Parnas. *Software Engineering Notes*, 1999, Vol. 24, No. 3. https://web.archive.org/web/20060712074630/http://www.sigsoft.org/SEN/parnas.html (06.12.2021)

[3] Flórez, F. B., Casallas, R., Hernández, M., Reyes A., Restrepo, S., Danies, G. Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, Vol. 87, No. 4, 2017, p. 834–860. https://www.jstor.org/stable/44667676 (06.12.2021)

[4] Pedaste, M., Palts, T., Kraav, T. Orav-Puurand, K. Komplekssete probleemide lahendamise oskus ning selle hindamine ja arendamine gümnaasiumis. *Eesti Haridusteaduste Ajakiri,* nr 9 (1), 2021. https://doi.org/10.12697/eha.2021.9.1.06 (06.12.2021)

[5] Halapuu, V., Valk, A. Täiskasvanute oskused Eestis ja maailmas: PIAAC uuringu esmased tulemused. Tartu: Haridus- ja Teadusministeerium, 2013. https://www.oecd.org/skills/piaac/Estonia.pdf (06.12.2021)

[6] Pruulmann-Vengerfeldt, P., Roots, A., Strenze, T., Ainsaar, M. Tehnoloogiarikkas keskkonnas probleemilahendusoskuse tase ja IKT kasutus Eesti elanike hulgas. PIAAC uuringu temaatiline aruanne nr 5. Tartu: Haridus- ja Teadusministeerium, 2015. https://www.hm.ee/sites/default/files/ikt_final_parandatud.pdf (25.04.2022)

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

[8] Sheard J. Simon, Carbone, A. et al. How difficult are exams? A framework for assessing the complexity of introductory programming exams. *Australasian Computing Education conference*, 2013, p. 145–154. https://openrepository.aut.ac.nz/bitstream/handle/10292/5151/Pages%20145ff%20from%20Vol136.pdf?sequence=2&isAllowed=y (06.03.2022)

[9] Bennedsen J, Caspersen ME. Assessing Process and Product: A Practical Lab Exam for an Introductory Programming Course. *Innovation in Teaching and Learning in Information and Computer Sciences*, 2007, Vol 6, Ed. 4, p. 182–202. https://www.tandfonline.com/doi/full/10.11120/ital.2007.06040183?scroll=top&needAccess=true (06.03.2022)

[10] Haghighi PD, Sheard J. Summative Computer Programming Assessment Using Both Paper and Computer. *Proceedings of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation*, 2005, p. 67–75. https://dl.acm.org/doi/10.5555/1563334.1563345 (06.03.2022)

[11] Kotkas, K. Programmeerimise algkursuste eksamid. Tartu Ülikooli arvutiteaduse instituut, Bakalaureusetöö. 2018. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=61909&year=2018 (06.03.2022)

[12] Tartu Ülikooli õppeinfosüsteem. Programmeerimine (6 EAP) LTAT.03.001 2021/2022 sügis. https://ois2.ut.ee/#/courses/LTAT.03.001/version/37ca53ac-440c-032f-1c5e-125c983bb04b/details (06.12.2021)

[13] Tartu Ülikooli arvutiteaduse instituudi kursused. Programmeerimine 2021/22 sügis. https://courses.cs.ut.ee/2021/programmeerimine/fall (06.12.2021)

[14] Tartu Ülikooli Moodle. Programmeerimine (LTAT.03.001). https://moodle.ut.ee/course/view.php?id=500 (06.12.2021)

[15] TÜ Arvutiteaduse instituudi programmeerimise algkursuse õpik. http://progeopik.ut.ee (06.12.2021)

[16] R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing, 2021, version 4.1.2. https://www.R-project.org/ (01.05.2022)

[17] Schauberger, P., Walker, A. openxlsx: Read, Write and Edit xlsx Files. 2021, package version 4.2.5. https://CRAN.R-project.org/package=openxlsx (01.05.2022)

[18] Wickham, H. et al. Welcome to the tidyverse. *Journal of Open Source Software*, 2019, Vol. 4, No. 43, p. 1686. https://doi.org/10.21105/joss.01686 (01.05.2022)

[19] Wickham, H. Reshaping Data with the reshape Package. *Journal of Statistical Software*, 2007, Vol. 21, No. 12, p. 1–20. http://www.jstatsoft.org/v21/i12 (01.05.2022)

[20] Xie, Y. knitr: A General-Purpose Package for Dynamic Report Generation in R. 2022, package version 1.38. https://yihui.org/knitr/ (01.05.2022)

[21] Wickham, H., Seidel, D. scales: Scale Functions for Visualization. 2022, package version 1.2.0. https://CRAN.R-project.org/package=scales (01.05.2022)

[22] Kassambara, A. ggpubr: 'ggplot2' Based Publication Ready Plots. 2020, package version 0.4.0. https://CRAN.R-project.org/package=ggpubr (01.05.2022)

[23] Pihlak, M. Klassikaline ja mitteparameetriline matemaatiline statistika: õpik kõrgkoolidele. Tallinn: TTÜ Kirjastus, 2018, lk 160. https://www.digar.ee/viewer/et/nlib-digar:408712/348821/page/161 (31.03.2022)

Lisad

I. Juhend I kontrolltöö arvestusliku osa koostamiseks

Juhend TÜ kursuse “Programmeerimine” I kontrolltöö arvestusliku osa koostamiseks

Arvestuslik osa koosneb kuuest küsimusest. Iga küsimus kontrollib ühe suurema teema teadmisi:

1. **avaldised ja lihtlaused** (muutujad, avaldised, omistuslause, andmetüübid, sisend ja väljund)
2. **tingimuslaused** (if, if-else, if-elif-else, tingimuslaused üksteise sees, tõeväärtustehted)
3. **funktsioonid** (funktsioonide defineerimine, rakendamine, lokaalsed ja globaalsed muutujad, parameetrid, väärtuse tagastamine)
4. **korduslaused** (while, for-tsükkel)
5. **sõnetöötlus** (tehted sõnedega)
6. **lihtsam failitöötlus** (faili kirjutamine, failist lugemine, tsükliga, ilma järjenditeta)

Küsimused valitakse juhuslikult konkreetse teema küsimuste hulgast Moodle’i küsimuste-pangast. See, kui palju peab küsimusi iga teema kohta küsimustepangas olema ja kui palju erinevaid ühe teema küsimusi läheb ühte tööse, oleneb töö tegijate arvust ja on kursuse õppejõudude otsustada.

Järeltöös ei tohi olla samad küsimused, mis olid põhitöös.

Soovituslik on järjestikustel aastatel kasutada erinevaid küsimusi ning iga aasta luua iga teema kohta vähemalt 1 uus küsimus.

Töösse küsimusi valides tuleb kontrollida, et kõik küsimused oleksid üheselt mõistetavad ning ühe teema küsimused oleksid sarnase raskusastmega. Kui ei ole, tuleks neid siis muuta, need kustutada või nende asemel luua uued.

Küsimused võiksid arvestuslikus testis olla järjestatud kursusel läbimise järjekorras (nagu üleval teemad), siis saab pärast statistikat tehes hea ülevaate ning tudengitel on ka loogiline vastata (st. ei kuluta alguses raskemale asjale liiga palju aega).

Iga küsimuse eest on võimalik saada 0 (vale vastus), 0,5 (50% või rohkem ulatuses õige vastus, oleneb täpsemalt küsimusest) või 1 punkti (täiesti õige vastus).

Töö saab arvestatud, kui saadakse 90% punktidest ehk 5,5 või 6 punkti.

II. Juhend I kontrolltöö programmeerimise osa koostamiseks koos ülesannete näidete ja vastavate hindamisjuhenditega

Juhend TÜ kursuse “Programmeerimine” I kontrolltöö programmeerimise osa koostamiseks

Üldised suunised

Programmeerimise osa koosneb kahest 10-punktilisest ülesandest. Ülesannetes on püstitatud mingi probleem/eluline olukord, kirjeldatud täpsemalt konkreetsete funktsioonide tööd (kui neid on) ning põhiprogrammi ülesannet.

Funktsioonide ja programmi tööst peavad olema ka näited sisendi ja väljundiga/tagastusega. Näited ei pea hõlmama kõiki võimalusi, kuid ülesande tekstis peavad kõik nõutud võimalused olema kirjeldatud. Erinevate variantide näited peavad olema samal tasemel abistavad (nt aritmeetilisi tehteid tegeva programmi puhul ei tohi olla nii, et ühes variandis on näide arvudega 1, 2, 3, kuid teises hoopis arvudega 25.7, 40.2, 1234 – teise variandi puhul tõenäoliselt ei ole ju näite põhjal programmi tegevus nii lihtsalt hoomatav kui esimese puhul).

Erinevaid variante koostades peab jälgima seda, et igas variandis ei oleks sama tüüpi arvutused (sh samad tingimuslaused, tõeväärtustehted, sõnetöötluse elemendid), kuid ülesannete raskusaste oleks sama.

Ülesannete tekstid peavad olema üheselt mõistetavad ja ei tohi eeldada mingi muu eriala spetsiifilisi teadmisi.

Kui ülesanded on valmis, peab vähemalt kaks õppejõudu need läbi lugema, lahendama ning vajadusel parandussoovitusi andma, et need oleksid üheselt mõistetavad ja ei eeldaks mingi muu eriala spetsiifilisi teadmisi. Üks inimene peab kontrollima ülesande teksti üle ka keeleliselt.

Ülesandega koos tuleb koostada ka ülesandele vastav hindamisjuhend võimalikult detailselt järgides juhendis toodud põhimõtteid.

Ülesande ja hindamisjuhendiga tuleb teistele õppejõududele kaasa panna ka näidislahendus.

1. ülesanne (10p)

1. ülesanne peab kindlasti katma järgmised teemad: avaldised ja lihtlaused, tingimuslaused, korduslaused, failitöötlus, kasutajaga suhtlus (sisend ja väljund).

Kohustuslikud komponendid ja punktide jaotumine:

- loetava faili avamine ja sulgemine (**0,5 p**)
    - “with”-konstruktsiooniga avamine –> 0,5 p
    - ainult avamine –> 0 p
    - faili avamine puudu –> 0 p
- kirjutatava faili avamine ja sulgemine (**0,5 p**)
    - “with”-konstruktsiooniga avamine –> 0,5 p
    - ainult avamine –> 0 p
    - faili avamine puudu –> 0 p
- failist lugemine tsükli abil (**1 p**)
    - kui lõpmatu tsükkel (programm jookseb kokku) –> 0 p
    - kui failist loetud ilma tsüklita –> 0 p
    - failist lugemine puudu –> 0 p
- tüübiteisendused (**1 p**)
    - kui on 1 teisendus ja see on tegemata või valesti –> 0 p
    - kui on mitu teisendust ja nendest 1 tegemata või valesti –> 0,5 p
    - kui on mitu teisendust ja nendest rohkem kui 1 tegemata või valesti –> 0 p
- faili kirjutamine (**1 p**)
    - õige informatsioon failis, kuid vales vormistuses –> 0,5 p
    - mingi informatsioon failist puudu –> 0 p
    - vale informatsioon failis –> 0 p
    - faili kirjutamine puudu –> 0 p
    - kui failis puudub informatsioon seetõttu, et fail on sulgemata, siis faili kirjutamise eest lisaks mitte punkte maha võtta

- tingimuslaused (**1–2 p**, olenevalt mahust, raskusest)
    - iga tingimuslause 0,5–1 p
        - kui vale loogika, siis selle tingimuslause eest 0 p
            - kui loogika käitub valesti ühel juhul mitmest , siis selle tingimuslause eest pooled punktid
        - kui mingi vajalik tingimuslause puudu, siis selle eest 0 p
- mingi(d) arvutus(ed) (summa, suurima, vähima, keskmise leidmine või mingi muu arvutus, sh ümardamised jne) (**2–3 p**, olenevalt mahust, raskusest)
    - hindamisjuhendisse täpsustada konkreetse ülesande puhul nende punktide jaotumine võimalikult detailselt
- kasutajalt sisendi küsimine (**0,5 p**)
    - sisendit pole programmi töös kasutatud –> 0 p
        - erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
    - kasutajalt sisendi küsimine puudu või valesti –> 0 p
- kasutajale info väljastamine (**1,5 p**)
    - jaotada 1,5 p erinevate väljundite vahel (nt vaata näidet), iga komponendi kohta:
        - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> pooled punktid
        - väljundis vale informatsioon –> 0 p
        - väljund puudu –> 0 p

Märkused:

1. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
2. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

1. ülesande näide

Juku vanaema tahab veenda vanaisa suitsetamisest loobuma ja näidata, kui palju sigarette on ostetud ja kui palju raha on asjatult kulutatud. Juku vanaema palus Jukul panna kirja, kui palju vanaisa kulutab igas kuus raha sigarettide ostmiseks. Kõik andmed on salvestatud faili ***kulud.txt***.

Faili pikkus ei ole teada, kuid võib eeldada, et andmed on failis kolme rea kaupa. Esimesel real on kuu nimetus, teisel kulutatud summa ja kolmandal ostetud pakkide arv. Ühe kuu jaoks võib olla andmeid sisestatud mitmel korral.

Näide faili *kulud.txt* võimalikust sisust

```
jaanuar
15.20
4
märts
12.85
3
veebruar
13.39
3
märts
8.10
2
```

Juku võttis ülesannet tõsiselt ja arvutas lisaks, kui palju nikotiini ja tõrva ostetud sigaretid sisaldasid. Igas pakis on 20 sigaretti ja üks sigaret sisaldab keskmiselt 0,9 mg nikotiini ja 11 mg tõrva.

Kirjuta programm, mis küsib kasutajalt kuu nime ja väljastab ekraanile sellel kuul kulutatud summa ja ostetud pakkide arvu loetaval kujul. Lisaks tuleb väljastada kogu kulutatud summa, ostetud pakkide arv ning summaarne tarbitud nikotiini- ja tõrva kogus.

On teada, et ühekordsel tarbimisel võib olla surmav juba 50 mg nikotiini. Kui summaarne nikotiini kogus ületab 50 mg, siis arvutada välja, mitu korda (ümardatuna täisarvuks) summaarne tarbitud kogus ületab surmavat kogust ja anda vastav hoiatus. Kulutatud summa kokku ja ostetud pakkide arv tuleb kirjutada faili ***kulud_kokku.txt*** eraldi ridadele.

Programmi võimalik väljund faili *kulud.txt* korral (kasutaja sisend on paksus kirjas)

```
Sisesta kuu nimi: märts
Kuus 'märts' kulutati 20.95 eurot ja osteti 5 pakki.
Kokku kulutati 49.54 eurot ja osteti 12 pakki.
Vanaisa tarbis kokku 216.0 mg nikotiini ja 2640 mg tõrva.
Selline kogus nikotiini ületab 4-kordse surmava annuse ühekordsel
tarbimisel!
```

Näide faili *kulud_kokku.txt* võimalikust sisust

```
49.54
12
```

Vastav hindamisskeem:

- loetava faili avamine ja sulgemine (**0,5 p**)
    - “with”-konstruktsiooniga avamine –> 0,5 p
    - ainult avamine –> 0 p
    - faili avamine puudu –> 0 p
- kirjutatava faili avamine ja sulgemine (**0,5 p**)
    - “with”-konstruktsiooniga avamine –> 0,5 p
    - ainult avamine –> 0 p
    - faili avamine puudu –> 0 p
- failist lugemine while-tsükli ja break abil (või for-tsükli abil) (**1 p**)
    - kui lõpmatu tsükkel (programm jookseb kokku) –> 0 p
    - kui failist loetud ilma tsüklita –> 0 p
    - failist lugemine puudu –> 0 p
- tüübiteisendused (**1 p**)
    - kui on 1 teisendus tegemata või valesti –> 0,5 p
    - kui rohkem kui 1 tegemata või valesti –> 0 p
- faili kirjutamine (**1 p**)
    - õige informatsioon failis, kuid vales vormistuses –> 0,5 p
    - mingi informatsioon failist puudu –> 0 p
    - vale informatsioon failis –> 0 p
    - faili kirjutamine puudu –> 0 p
    - kui failis puudub informatsioon seetõttu, et fail on sulgemata, siis faili kirjutamise eest lisaks mitte punkte maha võtta
- tingimuslaused (**1,5 p**)
    - tingimuslause failist lugemise lõpetamiseks (**0,5 p**)
        - vale loogika –> 0 p
        - puudu –> 0 p
    - tingimuslause õige kuu leidmiseks (**0,5 p**)
        - vale loogika –> 0 p
        - puudu –> 0 p
    - tingimuslause hoiatuse väljastamiseks (**0,5 p**)
        - kui loogika käitub valesti ühel juhul mitmest –> 0,25 p
        - vale loogika –> 0 p
        - puudu –> 0 p

- arvutused (**2,5 p**)
    - kuu koguse ja maksumuse leidmine (**1 p**)
        - vale tulemus –> 0 p
        - puudu –> 0 p
    - summaarse koguse ja maksumuse leidmine (**0,5 p**)
        - vale tulemus –> 0 p
        - puudu –> 0 p
    - nikotiini ja tõrva koguse leidmine (**1 p**)
        - vale tulemus –> 0 p
        - puudu –> 0 p
- kasutajalt sisendi küsimine (**0,5 p**)
    - sisendit pole programmi töös kasutatud –> 0 p
        - erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
    - kasutajalt sisendi küsimine puudu või valesti –> 0 p
- kasutajale info väljastamine (**1,5 p**)
    - küsitud kuul kulutatud summa ja pakkide arv (**0,5 p**)
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0,25 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p
    - kogu kulutatud summa, pakkide arv, tarbitud nikotiini ja tõrva kogus (**0,5 p**)
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0,25 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p
    - hoiatus (**0,5 p**)
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0,25 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p

Märkused:

1. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
2. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

2. ülesanne (10 p)

2. ülesanne peab kindlasti katma järgmised teemad: avaldised ja lihtlaused, tingimuslaused, funktsioonid, sõnetöötlus, kasutajaga suhtlus (sisend ja väljund).

Kohustuslikud komponendid ja punktide jaotumine:

- kasutajalt sisendi(te) küsimine (**1 p**)
    - kui mitu sisendit ja neist 1 puudu või valesti –> 0,5 p
    - kui mitu sisendit ja neist rohkem kui 1 puudu või valesti–> 0 p
    - kui mitu sisendit ja neist 1 pole programmi töös kasutatud –> 0,5 p
    - kui mitu sisendit ja neist rohkem kui 1 pole programmi töös kasutatud –> 0 p
    - kui 1 sisend ja see on puudu või valesti või pole programmi töös kasutatud –> 0 p
    - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
- sisendi(te) tüübiteisendus (**1 p**)
    - kui on 1 teisendus ja see on tegemata või valesti –> 0 p
    - kui on mitu teisendust ja nendest 1 tegemata või valesti –> 0,5 p
    - kui on mitu teisendust ja nendest rohkem kui 1 tegemata või valesti –> 0 p
- funktsioon 1 (**3,5 p**)
    - funktsiooni välja kutsumine õigete argumentidega (**1 p**)
        - kui vähemalt 1 argument on vale või puudu –> 0 p
    - õiged arvutused, sh tõeväärtustehted/tingimuslaused, sõnetöötlus (**1,5 p**)
        - hindamisjuhendisse täpsustada konkreetse ülesande puhul nende punktide jaotumine võimalikult detailselt
    - vastuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p

    - globaalsete muutujate mittekasutamine (**0,5 p**)
        - kui on kasutatud kasvõi ühte globaalset muutujat –> 0 p
- funktsioon 2 (**3,5 p**)
    - funktsiooni välja kutsumine õigete argumentidega (**1 p**)
        - kui vähemalt 1 argument on vale või puudu –> 0 p
    - õiged arvutused, sh tõeväärtustehted/tingimuslaused, sõnetöötlus (**1,5 p**)
        - hindamisjuhendisse täpsustada konkreetse ülesande puhul nende punktide jaotumine võimalikult detailselt
    - vastuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p
    - globaalsete muutujate mittekasutamine (**0,5 p**)
        - kui on kasutatud kasvõi ühte globaalset muutujat –> 0 p
- kasutajale info väljastamine (**1 p**)
    - jaotada 1 p erinevate väljundite vahel (nt vaata näidet), iga komponendi kohta:
        - väljundis õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0 p
        - väljundis vale informatsioon –> 0 p
        - väljund puudu –> 0 p

Märkused:

1. Kahe funktsiooni asemel võib olla ka 1 või 3, kuid sellisel juhul tuleb punktid jaotada ümber proportsionaalselt.
2. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).
3. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.

4. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

2. ülesande näide

Lauamängu saab mängida erinev arv mängijaid. Koosta funktsioon `mitu_mängu`, mis võtab argumentideks lauamänguõhtul osalejate arvu ja maksimaalse üht mängu mängida saavate mängijate arvu, ning tagastab lauamängude arvu (täisarvuna), mida on vaja, et kõik osalejad saaks seda mängida.

Näited funktsiooni `mitu_mängu` tööst

```
>>> mitu_mängu(10, 4)
3
>>> mitu_mängu(6, 3)
2
```

Ühe mängu rendihind on 5 eurot. Koosta funktsioon `kas_raha_jätkub`, mis võtab argumentideks lauamängude arvu (täisarvuna), lauamänguõhtu osalustasu ehk mängude rentimiseks kasutatava rahasumma ühe inimese kohta (ujukomaarvuna) ja osalejate arvu (täisarvuna) ning tagastab tõeväärtuse `True`, kui mängude rentimiseks raha jätkub, ja `False`, kui raha ei jätku.

Näited funktsiooni `kas_raha_jätkub` tööst

```
>>> kas_raha_jätkub(1, 2.5, 2)
True
>>> kas_raha_jätkub(3, 1.5, 3)
False
```

Kirjuta põhiprogramm, mis

- küsib kasutajalt
    - osalejate arvu (täisarvuna),
    - ühte mängu mängida saavate inimeste arvu (täisarvuna),
    - lauamänguõhtul osalemise tasu ühe inimese kohta (ujukomaarvuna),
- leiab funktsioone `mitu_mängu` ja `kas_raha_jätkub` kasutades, mitu mängu tuleb rentida ja kas selleks on piisavalt raha ning väljastab selle informatsiooni ekraanile.

Näited programmi tööst (kasutaja sisend on paksus kirjas)

```
Sisesta osalejate arv: 7
Sisesta ühte mängu mängida saavate inimeste arv: 3
Sisesta osalustasu ühe inimese kohta: 2.5
Tuleb rentida 3 mängu. Raha on selleks piisavalt.
```

```
Sisesta osalejate arv: 4
Sisesta ühte mängu mängida saavate inimeste arv: 4
Sisesta osalustasu ühe inimese kohta: 1
Tuleb rentida 1 mäng. Kahjuks raha ei jätku.
```

Vastav hindamisskeem:

- kasutajalt sisendite küsimine (**1 p**)
    - kui 1 sisendi küsimine on puudu või valesti –> 0,5 p
    - kui rohkem kui 1 sisendi küsimine on puudu või valesti –> 0 p
    - ühte sisendit pole programmi töös kasutatud –> 0,5 p
    - rohkem kui ühte sisendit pole programmi töös kasutatud –> 0 p
    - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
- sisendite tüübiteisendus (**1 p**)
    - kui kõigist teisendustest 1 tegemata või valesti–> 0,5 p
    - kui rohkem kui 1 teisendus tegemata või valesti –> 0 p
- funktsioon mitu_mängu() (**3,5 p**)
    - funktsiooni välja kutsumine õigete argumentidega (**1 p**)
        - kui vähemalt 1 argument on vale või puudu –> 0 p
    - mängude arvu leidmine ceil või tingimuslause abil (**1,5 p**)
        - vale arvutus –> 0 p
        - tingimuslause abil arvutades, kui loogika käitub valesti ühel juhul mitmest –> 0,5 p
    - vastuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p
    - globaalsete muutujate mittekasutamine (**0,5 p**)
        - kui on kasutatud kasvõi ühte globaalset muutujat –> 0 p
- funktsioon kas_raha_jätkub() (**3,5 p**)
    - funktsiooni välja kutsumine õigete argumentidega (**1 p**)
        - kui vähemalt 1 argument on vale või puudu –> 0 p
    - tingimuslause abil tõeväärtuse leidmine (**1,5 p**)
        - vale loogika –> 0 p
        - kui loogika käitub valesti ühel juhul mitmest –> 0,5 p
    - vastuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p
    - globaalsete muutujate mittekasutamine (**0,5 p**)

        - kui on kasutatud kasvõi ühte globaalset muutujat –> 0 p
- kasutajale info väljastamine (**1 p**)
    - mängude arv (**0,5 p**)
        - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> 0 p
        - väljundis vale informatsioon –> 0 p
        - väljund puudu –> 0 p
    - raha jätkumine (**0,5 p**)
        - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> 0 p
        - väljundis vale informatsioon –> 0 p
        - väljund puudu –> 0 p

Märkused:

1. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).
2. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
3. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

III. Juhend II kontrolltöö arvestusliku osa koostamiseks

Juhend TÜ kursuse “Programmeerimine” II kontrolltöö arvestusliku osa koostamiseks

Arvestuslik osa koosneb kuuest küsimusest. Iga küsimus kontrollib ühe suurema teema teadmisi:

1. **Järjendid I** (lihtsamad operatsioonid järjenditega, indeksid, järjendite töötlemine ilma tsüklita)
2. **Järjendid II** (järjendist otsimine, kokkuvõtte tegemine, koostamine elementhaaval, teisendamine, filtreerimine, kombineerimine tsükliga)
3. **Kahekordne tsükkel**
4. **Keerulisem failitöötlus** (failist lugemine järjenditesse)
5. **Teised andmestruktuurid** (ennikud, hulgad, sõnastikud)
6. **Mitmemõõtmelised andmestruktuurid** (mitmemõõtmelised järjendid ja sõnastikud, mis sisaldavad järjendeid, sõnastikke või ennikuid)

Küsimused valitakse juhuslikult konkreetse teema küsimuste hulgast Moodle’i küsimuste-pangast. See, kui palju peab küsimusi iga teema kohta küsimustepangas olema ja kui palju erinevaid ühe teema küsimusi läheb ühte tööse, oleneb töö tegijate arvust ja on kursuse õppejõudude otsustada.

Järeltöös ei tohi olla samad küsimused, mis olid põhitöös.

Soovituslik on järjestikustel aastatel kasutada erinevaid küsimusi ning iga aasta luua iga teema kohta vähemalt 1 uus küsimus.

Töösse küsimusi valides tuleb kontrollida, et kõik küsimused oleksid üheselt mõistetavad ning ühe teema küsimused oleksid sarnase raskusastmega. Kui ei ole, tuleks neid siis muuta, need kustutada või nende asemel luua uued.

Küsimused võiksid arvestuslikus testis olla järjestatud kursusel läbimise järjekorras (nagu üleval teemad), siis saab pärast statistikat tehes hea ülevaate ning tudengitel on ka loogiline vastata (st. ei kuluta alguses raskemale asjale liiga palju aega).

Iga küsimuse eest on võimalik saada 0 (vale vastus), 0,5 (50% või rohkem ulatuses õige vastus, oleneb täpsemalt küsimusest) või 1 punkti (täiesti õige vastus).

Töö saab arvestatud, kui saadakse 90% punktidest ehk 5,5 või 6 punkti.

IV. Juhend II kontrolltöö programmeerimise osa koostamiseks koos ülesannete näidete ja vastavate hindamisjuhenditega

Juhend TÜ kursuse “Programmeerimine” II kontrolltöö programmeerimise osa koostamiseks

Üldised suunised

Programmeerimise osa koosneb kahest 10-punktilisest ülesandest. Ülesannetes on püstitatud mingi probleem/eluline olukord, kirjeldatud täpsemalt konkreetsete funktsioonide tööd ning põhiprogrammi ülesannet.

Funktsioonide ja programmi tööst peavad olema ka näited sisendi ja väljundiga/tagastusega. Näited ei pea hõlmama kõiki võimalusi, kuid ülesande tekstis peavad kõik nõutud võimalused olema kirjeldatud. Erinevate variantide näited peavad olema samal tasemel abistavad (nt aritmeetilisi tehteid tegeva programmi puhul ei tohi olla nii, et ühes variandis on näide arvudega 1, 2, 3, kuid teises hoopis arvudega 25.7, 40.2, 1234 – teise variandi puhul tõenäoliselt ei ole ju näite põhjal programmi tegevus nii lihtsalt hoomatav kui esimese puhul).

Erinevaid variante koostades peab jälgima seda, et igas variandis ei oleks sama tüüpi arvutused (sh samad andmestruktuurid), kuid ülesannete raskusaste oleks sama.

Ülesannete tekstid peavad olema üheselt mõistetavad ja ei tohi eeldada mingi muu eriala spetsiifilisi teadmisi.

Kui ülesanded on valmis, peab vähemalt kaks õppejõudu need läbi lugema, lahendama ning vajadusel parandussoovitusi andma, et need oleksid üheselt mõistetavad ja ei eeldaks mingi muu eriala spetsiifilisi teadmisi. Üks inimene peab kontrollima ülesande teksti üle ka keeleliselt.

Ülesandega koos tuleb koostada ka ülesandele vastav hindamisjuhend võimalikult detailselt järgides juhendis toodud põhimõtteid.

Ülesande ja hindamisjuhendiga tuleb teistele õppejõududele kaasa panna ka näidislahendus.

1. ülesanne (10p)

1. ülesanne peab kindlasti katma järgmised teemad: failist lugemine funktsiooniga, mitmemõõtmelised andmestruktuurid.

Kohustuslikud komponendid ja punktide jaotumine:

- funktsioon failist lugemiseks (**4 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - faili avamine ja sulgemine (**0,5 p**)
        - “with”-konstruktsiooniga avamine –> 0,5 p
        - ainult avamine –> 0 p
        - faili avamine puudu –> 0 p
    - järjendi/sõnastiku loomine (**0,5 p**)
        - vale struktuuri loomine –> 0 p
        - puudu –> 0 p
    - sisemiste järjendite/sõnastike/ennikute/hulkade loomine (**1 p**)
        - vale struktuuri loomine –> 0 p
        - vale loogika/ülesehitus –> 0 p
        - puudu –> 0 p
    - tsükkel üle faili ridade kuni on ridasid (**1 p**)
        - tsükli lõpetamise tingimus ei ole see, et read said otsa –> 0 p
        - pole loetud tsükliga –> 0 p
    - mitmemõõtmelise järjendi/sõnastiku tagastamine (**0,5 p**)
        - ühemõõtmelise, valesti struktureeritud või vale informatsiooni sisaldava järjendi/sõnastiku tagastamine –> 0 p
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti –> 0 p

- kasutajalt sisendite küsimine lõpmatu tsükliga (**1,5 p**)
    - pole lahendatud lõpmatu tsükliga –> 0 p
    - kui 1 sisendi küsimine on puudu või valesti –> 0,5 p
    - kui rohkem kui 1 sisendi küsimine on puudu või valesti –> 0 p
    - kui sisendeid pole programmi töös kasutatud –> 0 p
        - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
    - kui mingi vajalik tüübiteisendus tegemata –> –0,5 p
- teated, kui sisend on vale ja uue sisendi küsimine (**1 p**)
    - ainult teade, aga uut sisendit ei küsita –> 0 p
    - uue sisendi küsimine ilma teateta –> 0 p
    - mitmest teatest üks valesti –> 0,5 p
- sisendi põhjal operatsioon/arvutus järjendil/sõnastikul (**2 p**)
    - võib selle jaoks nõuda ka eraldi funktsiooni
    - hindamisjuhendisse täpsustada konkreetse ülesande puhul nende punktide jaotumine võimalikult detailselt
- kasutajale info väljastamine (**1,5 p**)
    - jaotada 1,5 p erinevate väljundite vahel (nt vaata näidet), iga komponendi kohta:
        - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> 0 p
        - väljundis vale informatsioon –> 0 p
        - väljund puudu –> 0 p

Märkused:

1. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).

2. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
3. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuid seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

1. ülesande näide

103 aastat tagasi kuulutas Läti Vabariik end iseseisvaks. Sünnipäeva puhul ostetakse kauplustest palju pürotehnikat. Et arvepidamine segi ei läheks, koostas kauplus kogu pürotehnika kohta hinnakirja, mis asub failis *hinnakiri.txt*. Failis on igal real komadega eraldatult toote nimi sõnena, hind ujukomaarvuna ja toote värvus sõnena. Ridade arv failis pole ette teada.

Näide faili *hinnakiri.txt* võimalikust sisust

```
taevalatern,3.80,kuldne
raketid,14.10,roheline
taevalatern,2.70,punane
```

Kirjuta funktsioon `loe_failist`, mis võtab argumendiks faili nime ning tagastab sõnastiku, kus võtmeks on toote nimi ning väärtuseks on sõnastik, kus on kirjas toote värvused ja hinnad. Selle sõnastiku võtmeks on värvus ning väärtuseks toote hind ujukomaarvuna. Samanimeliste toodete värvused ja hinnad tuleb lisada sisemisse sõnastikku.

Näide funktsiooni `loe_failist` tööst

```
>>> loe_failist("hinnakiri.txt")
{'taevalatern': {'kuldne': 3.8, 'punane': 2.7}, 'raketid':
{'roheline': 14.1}}
```

Põhiprogrammis tuleb funktsiooni `loe_failist` abil sisse lugeda andmed failist *hinnakiri.txt*. Seejärel peab programm väljastama eraldi ridadele hinnakirjas olevad tooted koos hinna ja värvusega, mis on sulgudes (soovitav teha eraldi funktsioon). Järgmiseks küsib programm kasutajalt, millist toodet ta soovib osta. Kui selle nimega toode on olemas, tuleb küsida kasutajalt värvust. Kui toode või vastava värvusega toode puudub, siis antakse sellest kasutajale teada ning küsitakse uuesti toote nime. Pärast igat edukat toote ostukorvi lisamist väljastatakse kasutajale valitud kaupade jooksev summa. Tooteid küsitakse seni kuni kasutaja sisestab tühja sõne (""). Seejärel väljastab programm tasumisele kuuluva kogusumma ning lõpetab töö.

Näide põhiprogrammi tööst (kasutaja sisend on paksus kirjas)

```
Kaupluses on müügil järgmised tooted:
taevalatern - 3.8€ (kuldne)
taevalatern - 2.7€ (punane)
raketid - 14.1€ (roheline)
```

```
Sisesta ostusoov: taevalatern
Sisesta toote värvus: sinine
Sellise värvusega toodet ei ole.
Sisesta ostusoov: taevalatern
Sisesta toote värvus: punane
Jooksev summa on 2.7€.
Sisesta ostusoov: raketid
Sisesta toote värvus: roheline
Jooksev summa on 16.8€.
Sisesta ostusoov:
Tasuda tuleb 16.8€.
```

Vastav hindamisskeem:

- funktsioon loe_failist() (**4 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - faili avamine ja sulgemine (**0,5 p**)
        - “with”-konstruktsiooniga avamine –> 0,5 p
        - ainult avamine –> 0 p
        - faili avamine puudu –> 0 p
    - sõnastiku loomine (**0,5 p**)
        - vale struktuuri loomine –> 0 p
        - puudu –> 0 p
    - sisemiste sõnastike loomine (**1 p**)
        - vale struktuuri loomine –> 0 p
        - vale loogika/ülesehitus –> 0 p
        - puudu –> 0 p
    - tsükkel üle faili ridade kuni on ridasid (**1 p**)
        - tsükli lõpetamise tingimus ei ole see, et read said otsa –> 0 p
        - pole loetud tsükliga –> 0 p
    - mitmemõõtmelise sõnastiku tagastamine (**0,5 p**)
        - ühemõõtmelise, valesti struktureeritud või vale informatsiooni sisaldava sõnastiku tagastamine –> 0 p
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti –> 0 p
- kasutajalt sisendite küsimine lõpmatu tsükliga (**1,5 p**)
    - pole lahendatud lõpmatu tsükliga –> 0 p
    - kui 1 sisendi küsimine on puudu või valesti –> 0,5 p
    - kui rohkem kui 1 sisendi küsimine on puudu või valesti –> 0 p
    - kui sisendeid pole programmi töös kasutatud –> 0 p

        - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
- teated, kui sisend on vale ja uue sisendi küsimine (**1 p**)
    - ainult teade, aga uut sisendit ei küsita –> 0 p
    - uue sisendi küsimine ilma teateta –> 0 p
    - mitmest teatest üks valesti –> 0,5 p
- sisendi põhjal operatsioonid sõnastikul (**2 p**)
    - sõnastikust õige toote leidmine ja õiget värvi toote hinna leidmine (**1,5 p**)
        - valesti –> 0 p
        - puudu –> 0 p
    - jooksva summa arvutamine (**0,5 p**)
        - valesti –> 0 p
        - puudu –> 0 p
- kasutajale info väljastamine (**1,5 p**)
    - müügil olevad tooted (**0,5 p**)
        - õige informatsioon, kuid mitte nõutud vormistuses –> 0,25 p
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p
    - jooksev summa (**0,5 p**)
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p
    - tasumisele kuuluv summa (**0,5 p**)
        - õige informatsioon, kuid mitte loetav (nt lihtsalt numbrid) –> 0 p
        - vale informatsioon –> 0 p
        - puudu –> 0 p

Märkused:

1. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).
2. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
3. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuid seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

2. ülesanne (10 p)

2. ülesanne peab kindlasti katma järgmised teemad: failist lugemine funktsiooniga, kahekordne tsükkel, andmestruktuurid, mida eelmises ülesandes ei käsitletud ja tehted nendel.

Kohustuslikud komponendid ja punktide jaotumine:

- sisendi küsimine (**0,5 p**)
    - kui sisendi küsimine on puudu või valesti –> 0 p
    - kui sisendit pole programmi töös kasutatud –> 0 p
        - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
    - kui mingi vajalik tüübiteisendus tegemata –> 0 p
- funktsioon failist lugemiseks (**4 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - faili avamine ja sulgemine (**0,5 p**)
        - “with”-konstruktsiooniga avamine –> 0,5 p
        - ainult avamine –> 0 p
        - faili avamine puudu –> 0 p
    - järjendi/sõnastiku loomine (**0,5 p**)
        - vale struktuuri loomine –> 0 p
        - puudu –> 0 p
    - sisemiste järjendite/sõnastike/ennikute/hulkade loomine (**1 p**)
        - vale struktuuri loomine –> 0 p
        - vale loogika/ülesehitus –> 0 p
        - puudu –> 0 p
    - tsükkel üle faili ridade kuni on ridasid (**1 p**)

        - tsükli lõpetamise tingimus ei ole see, et read said otsa –> 0 p
        - pole loetud tsükliga –> 0 p
    - mitmemõõtmelise järjendi/sõnastiku tagastamine (**0,5 p**)
        - ühemõõtmelise, valesti struktureeritud või vale informatsiooni sisaldava sõnastiku tagastamine –> 0 p
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti –> 0 p
- funktsioon tulemuse leidmiseks (**5 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - arvutused/tehted andmestruktuuriga (sh kahekordse tsükli rakendamine) (**4 p**)
        - hindamisjuhendisse täpsustada konkreetse ülesande puhul nende punktide jaotumine võimalikult detailselt
    - tulemuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p
- tulemuse väljastamine (**0,5 p**)
    - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> 0 p
    - väljundis vale informatsioon –> 0 p
    - väljund puudu –> 0 p

Märkused:

1. Ühe funktsiooni asemel võib olla ka 2 või 3, kuid sellisel juhul tuleb punktid jaotada ümber proportsionaalselt.

2. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).
3. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti. Kui veateade tekib mingi konkreetse sisendi puhul, tuleb vaadata, mis on valesti ja selle eest punkte maha võtta.
4. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

2. ülesande näide

Failis *hokiturniir.txt* on kirjas hokiturniiri tulemused võistkondade kaupa. Igal real on ühe hokivõistkonna mängude tulemused, kus `"V"` on võit ja `"K"` on kaotus (sellel turniiril viike ei ole). Tulemuste eraldajaks on semikoolon (`';'`).Võib eeldada, et igal võistkonnal on sama palju tulemusi. Ridade arv failis pole ette teada.

Näide faili *hokiturniir.txt* võimalikust sisust

```
K;K;K
V;V;K
V;V;V
K;K;V
```

Koostada funktsioon `loe_tulemused`, mis võtab argumendiks failinime ja tagastab kahemõõtmelise järjendi hokitulemuste kohta.

Näide funktsiooni `loe_tulemused` tööst

```
>>> loe_tulemused('hokiturniir.txt')
[['K', 'K', 'K'],
 ['V', 'V', 'K'],
 ['V', 'V', 'V'],
 ['K', 'K', 'V']]
```

Read on paigutatud näites ülevaatlikkuse mõttes üksteise alla.

Kirjuta funktsioon `kes_võitis`, mis võtab argumendiks kahemõõtmelise järjendi ja tagastab täisarvu, mitmes võistkond teenis kõige rohkem võite. Loendamine toimub selliselt, et kui kõige rohkem võite saavutas esimesel real olev võistkond, siis tagastab funktsioon täisarvu 1, kui teisel real olev võistkond, siis arvu 2 jne. Kui mitmel võistkonnal on võite sama palju, siis tagastatakse esimese sellise võistkonna järjekorranumber.

Näide funktsiooni `kes_võitis` tööst

```
>>> kes_võitis([['V', 'V', 'V'], ['K', 'K', 'K'], ['V', 'V', 'K'],
['K', 'K', 'V']])
```

```
1
>>> kes_võitis([['V', 'K', 'K'], ['K', 'V', 'V'], ['V', 'K', 'V'],
['V', 'K', 'K']])
2
```

Koosta programm, mis

- küsib kasutajalt failinime ja loeb andmed failist kahemõõtmelisse järjendisse, kasutades funktsiooni `loe_tulemused`,
- väljastab ekraanile, mitmes võistkond sai kõige rohkem võite, kasutades funktsiooni `kes_võitis`.

Näide programmi tööst (kasutaja sisend on paksus kirjas)

```
Sisestage failinimi: hokiturniir.txt
Turniiri võitis 3. võistkond.
```

Vastav hindamisskeem:

- sisendi küsimine (**0,5 p**)
    - kui sisendi küsimine on puudu või valesti –> 0 p
    - kui sisendit pole programmi töös kasutatud –> 0 p
    - NB! erandina ei võeta sisendi programmi töös mittekasutamise eest maha punkte, kui see ei ole võimalik, sest mingi muu osa pole implementeeritud, kuid sisendi küsimine on õigesti lahendatud
- funktsioon failist lugemiseks (**4 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - faili avamine ja sulgemine (**0,5 p**)
        - “with”-konstruktsiooniga avamine –> 0,5 p
        - ainult avamine –> 0 p
        - faili avamine puudu –> 0 p
    - järjendi loomine (**0,5 p**)
        - vale struktuuri loomine –> 0 p
        - puudu –> 0 p
    - sisemiste järjendite loomine (**1 p**)

        - vale struktuuri loomine –> 0 p
        - vale loogika/ülesehitus –> 0 p
        - puudu –> 0 p
    - tsükkel üle faili ridade kuni on ridasid (**1 p**)
        - tsükli lõpetamise tingimus ei ole see, et read said otsa –> 0 p
        - pole loetud tsükliga –> 0 p
    - mitmemõõtmelise järjendi/sõnastiku tagastamine (**0,5 p**)
        - ühemõõtmelise, valesti struktureeritud või vale informatsiooni sisaldava sõnastiku tagastamine –> 0 p
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti –> 0 p
- funktsioon tulemuse leidmiseks (**5 p**)
    - selle funktsiooni õige väljakutsumine ja rakendamine (**0,5 p**)
        - globaalsete muutujate kasutamine –> 0 p
        - väljakutsumine valede argumentidega –> 0 p
    - arvutused/tehted andmestruktuuriga (**4 p**)
        - kahekordse tsükli üle maatriksi õige ja loogiline rakendamine (1 p)
        - reas võitude arvu leidmine (1 p)
        - maksimaalse võitude arvu leidmine (1 p)
        - õige rea numbri leidmine (1 p)
    - tulemuse tagastamine (**0,5 p**)
        - vastuse väljastamine (tagastamise asemel või lisaks tagastamisele) –> 0 p
        - on puudu või valesti (sh vale andmetüüp) –> 0 p
- tulemuse väljastamine (**0,5 p**)
    - väljundis õige informatsioon, kuid mitteloetav (nt lihtsalt numbrid) –> 0 p
    - väljundis vale informatsioon –> 0 p

- väljund puudu –> 0 p

Märkused:

1. Kui ülesandes on nõutud, et mingit osa peab lahendama funktsiooniga, kuid on lahendatud ilma funktsiooni defineerimata, siis saab selle eest 0 punkti (punkte saab siis ainult teiste osade eest).
2. Kui programmi käivitamisel tuleb veateade, tuleb hinnata ainult neid komponente, mida täidetakse enne vea tekkimist. Vea põhjustanud komponendi ja sellele järgnevate eest tuleb anda 0 punkti.
3. Kui põhiprogrammi mingi osa lahendatakse funktsiooni abil, kuigi seda pole otseselt nõutud, hinnatakse komponente ikka samamoodi.

V. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Triinu Aug,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Kontrolltööde koostamise ja hindamise juhendite loomine Tartu Ülikooli kursusele „Programmeerimine“,

   mille juhendaja on Tauno Palts, 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 4.0, 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.

Triinu Aug
10.05.2022