TARTU ÜLIKOOL

Arvutiteaduste instituut

Informaatika õppekava

Kadi Rõmmel

E-kursuse „Programmeerimise alused II“ kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs

Bakalaureusetöö (9 EAP)

Juhendaja: Eno Tõnisson

Tartu 2017

E-kursuse „Programmeerimise alused II“ kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs

Lühikokkuvõte:

Käesoleva bakalaureusetöö tulemusena valmisid kahemõõtmelise järjendi teemade esialgsed materjalid e-kursusele „Programmeerimise alused II“. Bakalaureusetöö käigus analüüsiti ka „Infotehnoloogia mitteinformaatikutele“ magistriõppekava üliõpilaste tagasisidet neile materjalidele. Antud bakalaureusetöö keskendub kahemõõtmelise järjendi peatükkide tagasisidele, et teha ettepanekuid materjalide täiustamiseks.

Võtmesõnad:

E-kursuse loomine, õppematerjalid, kahemõõtmeline järjend, kahekordne tsükkel

CERCS: P175 – Informaatika, süsteemiteooria

Two-Dimensional Array Initial Materials and Analysis for „Introduction to Programming II“ e-course

Abstract:

The purpose of this thesis is to create initial materials about two-dimensional array for e-course „Introduction to Programming II“. During this thesis the feedback of the master’s students from „Conversion Master in IT“ curriculum was analysed. This thesis is focused on the feedback of the two-dimensional array to improve the materials for MOOC.

Keywords:

Creating e-course, educational material, two-dimensional arrays, nested loops

CERCS: P175 – Informatics, system theory

Sisukord

1. Sissejuhatus

Tartu ülikooli arvutiteaduse instituudis on korraldatud vaba juurdepääsuga programmeerimise e-kursuseid alates aastast 2014, mil toimus esimest korda kursus „Programmeerimisest maalähedaselt“. 2016. aasta alguses loodi jätkukursus „Programmeerimise alused“ [1]. Mõlemad kursused on jätkuvalt väga populaarsed ning osalejate arv on püsinud kõrge, ületades mitmetel juhtudel 1000 piiri. Kursustel kasutatakse programmeerimiskeelt Python. Kursuse „Programmeerimise alused“ lõpus olnud küsitluses tuli välja, et väga paljud inimesed oleksid huvitatud ka jätkukursusest „Programmeerimise alused II“. Seega otsustatigi see kursus välja töötada. „Programmeerimise alused II“ kursusel on kolm suurt teemat, kahemõõtmelised järjendid, rekursioon ja andmestruktuurid. Antud bakalaureusetöö eesmärk on luua ning analüüsida ka teiste valmistatud materjale, mida kasutada „Programmeerimise alused II“ e-kursuse kahemõõtmelise järjendi ning kahekordse tsükli temaatika juures. Kahemõõtmelise järjendi ning kahekordse tsükli teema on antud kursusel üks tähtsamaid ning ka programmeerimise seisukohalt väga vajalik teema, mida peaks iga programmeerija hästi mõistma. Kahemõõtmelise järjendi peatükk, mida antud bakalaureusetöö raames käsitletakse, on antud kursusel esimene teema, seega on osalejatel vaja kiiresti põhitõed ning varasemalt õpitu meelde tuletada.

Bakalaureusetöö autor on Tartu ülikooli e-kursustega seotud alates 2016. aasta algusest, mil liitus kursuste korraldustiimiga. Autori põhiliseks ülesandeks oli kursustel „Programmeerimisest maalähedaselt“ ning „Programmeerimise alused“ meili teel inimeste abistamine ning ülesannetega hätta jäänud osalejate suunamine õigele teele. „Programmeerimise alused II“ kursuse materjalid valmisid suuresti koostöös teiste kursuse korraldajatega. Bakalaureusetöö autor koostas:

1. 18 kontrollküsimust;
2. seitse eksami harjutusülesannet;
3. kaks Moodle’i testi, mis koosnesid mõlemad viiest küsimusest;

Kontrollküsimused on materjalidesse põimitud ning aitavad materjale parmini mõista. Küsimused on valikvastustega ning teksti sees. Vajutades vastusele on koheselt näha ka tagasisidet ning seletust. Koostatud seitsmest eksami harjutusülesandest läksid kasutusele neli. Lisaks eksamiülesannetele andis bakalaureusetöö autor ka ideid arvestusülesannete loomiseks, kuid ülesannete püstituse vormistasid teised kursuse korraldajad. Bakalaureusetöö autor

koostas ka kaks Moodle'i testi, milles kummaski oli viis küsimust. Samuti proovis bakalaureusetöö autor koostada ka ise esialgset teoreetilist õppematerjali ning andis ka tagasisidet teiste loodud materjalidele. Näited koostatud materjalidest on ka lisades välja toodud.

Loodud ülesandeid ning materjale said kasutada „Infotehnoloogia mitteinformaatikutele" magistriõppe õppekava üliõpilased 2016. aasta sügissemestril. Nende kursus on segu statsionaarsest ning e-kursusest, mis tähendab, et materjalid ning ülesanded kursustel kattuvad e-kursuse materjalidega. Lisaks olemasolevatele materjalidele toimusid neil ka üle nädala loengud ning lisapraktikumid, kus ka kursuse autor kohal käis, et vahetut tagasisidet saada.

Antud bakalaureusetöö on jagatud neljaks osaks. Esimeses osas räägitakse lähemalt kahekordsest tsüklist ning kahemõõtmelisest järjendist, milleks seda on vaja ja kuidas seda teemat on erinevates õpikutes käsitletud. Teises osas antakse ülevaade olemasolevatest programmeerimise vaba juurdepääsuga e-kursustest Tartu ülikooli arvutiteaduse instituudis ning nende ülesehitusest. Tutvustatakse keskkondi, mida kasutatakse nii eelnevate kui ka käesoleva kursuse raames. Kolmandas osas kirjeldatakse materjalide valmimise protsessi. Neljandas osas analüüsitakse loodud materjale, mille järgi õppisid „Infotehnoloogia mitteinformaatikutele" üliõpilased. Analüüsimisel võeti vaatluse alla üliõpilaste tagasiside, arvestusülesannete lahendused ning Moodle'i testide lahendused.

2. Kahemõõtmeline järjend

2.1 Kahemõõtmelise järjendi olulisus

Antud jaotises antakse põgus ülevaade kahemõõtmelisest järjendist ning selle olulisusest. Seletatakse ka maatriksi mõistet. Tegemist ei ole õppematerjaliga ning seda lühitutvustust kursustel ei kasutata.

Pythonis on järjendis võimalik hoida erinevat tüüpi andmeid, näiteks võivad järjendi elementideks olla ka omakorda järjendid. Kui järjendi elementideks on järjendid, siis on tegemist kahemõõtmelise järjendiga. Järjendid ei pea olema ainult ühe või kahemõõtmelised, võimalik on kasutada ka kolme ja rohkema mõõtmelisi järjendeid [2]. Selliseid järjendeid nimetatakse mitmemõõtmelisteks järjenditeks. Kahemõõtmeline järjend, mida antud kursusel käsitletakse, on kõige lihtsam vorm mitmemõõtmelisest järjendist. Nende abil on võimalik kujutada tabeleid ja maatrikseid. Maatriksid on ristkülikukujulised järjendid, mis koosnevad kindlast arvust ridadest n ja veergudest m (n x m maatriks). Maatriksite abil on võimalik lahendada keerukaid võrrandisüsteeme [3]. Üks näide maatriksite abil keerukate võrrandsüsteemide lahendamisest on Netflixi filmisoovituste andmine kasutajale. Kasutatakse kasutaja-filmi maatrikseid, kus elementideks on kasutajate poolt filmidele antud skoorid. Antud maatriksis võib olla ka tundmatuid, mis tuleb olemasolevate andmete põhjal täita [4]. Selliseid ülesandeid lahendatakse masinõppe meetodeid kasutades. Kahemõõtmelised järjendid ning tabelid on masinõppe ülesannete juures väga tihti kasutusel. Antud kursusel nii keerulisi ülesandeid ei lahendata. Maatriksi definitsioonile kursusel väga palju rõhku ei pöörata ning antud kursusel kasutatakse neid näiteks, et kujutada sudoku tabelit.

2.2 Kahemõõtmelise järjendi teema käsitlemine programmeerimisõpikutes

Antud bakalaureusetöös sai vaadeldud mõningaid programmeerimiskeelt Python õpetavaid õpikuid, mis sisaldasid kahemõõtmelise järjendi temaatikat. Kahemõõtmelisus tuleb õpikutes sisse väga erinevas järjekorras ning erineval moel. Paljudes algajatele mõeldud õpikutes ei ole kahemõõtmelisest järjendist juttu, kuid leidub ka selliseid, kus on kahemõõtmeline järjend üsna vara sisse toodud. Näiteks raamatus „Python 3 for Absolute Beginners“ [5] on viies peatükk järjendite kasutamisest, milles on ka väike alateema tutvustamaks mitmemõõtmelisi järjendeid. Antud alateema sisaldab väga lühikest kahemõõtmelise järjendi ning maatriksi tutvustust ning antud alateema juures koodinäited puuduvad. Sellest, kuidas mitmemõõtmelisest järjendist

elemente kätte saada, kirjutatakse peatükis, mis räägib andmete töötlemisest ning seal koos ühemõõtmeliste järjenditega.

Tony Gaddis kirjutab oma raamatus „Starting Out with Python. Second Edition“ [6] kahemõõtmelisest järjendist järjendite ja ennikute peatükis, mis on raamatus kaheksas peatükk. Antud raamatus on kahemõõtmelisi järjendeid õpetatud veidi põhjalikumalt. Seletatud on kahemõõtmelise järjendi mõiste, toodud on palju näiteid. Põhjalikult on seletatud näidetega, kuidas indeksite abil elemente kätte saada. Välja on toodud ka tabel, kus on hea näha, mis indeksitega millised elemendid kätte saab. Lisaks on veel näide, kuidas juhuslike arvudega kahemõõtmeline järjend ära täita, kasutades selleks kahekordset tsüklit. Kõik näites olevad read on põhjalikult ära seletatud. Kahemõõtmelise järjendi teema lõpus on kolm kordavat küsimust, mis on jäetud lugejale endale nuputamiseks ja järele proovimiseks. Lisaks alateema lõpus olevatele küsimustele on ka peatüki lõpus erinevaid küsimusi teema kohta, näiteks valikvastustega küsimused, õige või vale küsimused, lühivastustega küsimused ning programmeerimisülesanded.

D. Y. Liangi õpikus „Introduction to Programming Using Python“ [7] on terve 11. peatükk pühendatud mitmemõõtmelisele järjendile. Põhiliselt käsitletakse siiski kahemõõtmelist järjendit. Peatükk algab sissejuhatusega, kus tabeli näitena on toodud linnade vahelised kaugused. Teise alapealkirja alt on võimalik leida kahemõõtmelise järjendi elementide kättesaamist indeksite abil. Lisaks on veel näiteid, kuidas kasutajalt küsitud andmetest koostada kahemõõtmeline järjend. Kahemõõtmelise järjendi peatükist leiab veel, kuidas printida välja kõik elemendid. Lisaks on veel erinevad näited, kuidas liita kokku kõik kahemõõtmelise järjendi elemendid ning ka veergude summa leidmise näide on toodud. Veeru summa leidmise ülesanne on väga hea näide, kuidas kasutada kahekordset tsüklit ka teises järjekorras, kus välimine tsükkel tähistab veergu ning sisemine rida. Ühe näitena on toodud veel kood, kuidas leida rida, mille summa on suurim. Antud peatükis räägitakse veel sorteerimisest ning selgitatakse, kuidas töötab *sort* funktsioon kahemõõtmelise järjendi puhul. Peatüki vahepeal on kordavad küsimused, mida saab lugeja ise läbi proovida. Järgmise alateemana käsitletakse, kuidas kahemõõtmelist järjendit funktsioonile argumendina ette anda. Edasi tulevad ülesanded, mis on esitatud probleemidena ning millel on ka lahendus koos seletustega. Esimeseks probleemülesandeks on valikvastustega testi hindamise programm, kus õpilaste vastuseid hoitakse kahemõõtmelises järjendis ning õiged lahendused on ühemõõtmelises järjendis. Programm peab iga õpilase kohta printima välja õpilase indeksi ning õigete vastuste arvu. Teiseks probleemiks on leida koordinaatteljestikul olevate punktide seast kaks üksteisele kõige

lähemal asuvat punkti. Viimaseks probleemülesandeks on sudoku lahenduse kontrollimine. Kahele viimasele ülesandele on lisaks lahendusele, lisatud ka näide graafilisest kasutajaliidesest. Kui enamik näiteid ning ülesandeid antud peatükis käsitlesid kahemõõtmelist järjendit, siis peatüki lõpu osas on räägitud ka kolmemõõtmelisest järjendist. Kolmemõõtmelise järjendi näitena on kasutatud kuue õpilase viie eksami tulemusi, kus iga eksam koosneb kahest osast. Ka kolmemõõtmelise järjendi alateema all on välja toodud paar probleemülesannet, neist esimene on seotud päevaste temperatuuri ja õhuniiskuse keskmise leidmisega. Failis on viimase kümne päeva andmed, kus on ühel real neli erinevat väärtust: päev, tund, temperatuur ja õhuniiskus. Failis olevad andmed ei pruugi olla õiges järjekorras. Ülesandeks on kirjutada programm, mis leiaks kümne päeva keskmise temperatuuri ja õhuniiskuse. Teiseks probleemülesandeks on sünnipäevade arvamine. Arvuti peab ära arvama, millal on kasutaja sünnipäev. Ette on antud kolmemõõtmeline järjend, mille seas on üks õige kuupäev ning kasutajalt võib küsida vaid jah-ei küsimusi. Peatüki lõpus on toodud peatüki kokkuvõte, link testile ning programmeerimisülesanded.

„Fundamentals of Python: Data Structures“ õpikus [8] ei ole päris terve peatükk kahemõõtmelistele järjenditele pühendatud. Antud õpikus jääb kahemõõtmelise järjendi temaatika järjendite peatüki alla, mis on õpikus neljas peatükk. Kõik olulisemad punktid on kahemõõtmelise järjendi alateema all ära selgitatud. Teema algab indeksitega elementide leidmisest kahemõõtmelisest järjendist. Järgmiseks on kohe kahekordne for-tsükkel, et läbida elementhaaval kahemõõtmelist järjendit. Lisaks on veel lõik sellest, kuidas ise kahemõõtmelist järjendit luua.

Kahemõõtmelise järjendi temaatika on käsitletud õpikutes väga erinevas järjekorras. Antud õpikutes on enne kahemõõtmelist järjendit läbitud alati tingimuslause temaatika ning muutujad ja andmetüübid ning kõigis on enne läbitud ka ühemõõtmelise järjendi temaatika. Funktsioonide temaatika järjekord on erinev, mõnes õpikus on see teema enne kahemõõtmelist ning mõnes pärast samuti ka failist lugemise temaatika. Ka graafika on käsitletud mõnes õpikus enne kahemõõtmelisust ning mõnes pärast. Kui rekursiooni on käsitletud, siis see on tavaliselt hiljem ning üsna õpikute lõpus.

Üldiselt, kui õpikutes või kursustel on kahemõõtmelisusest kirjutatud, siis on kindlasti seletatud kaks põhilist teemat:

1. Kahemõõtmelise järjendi elementide kättesaamine indeksite abil;
2. Kahemõõtmelise järjendi läbimine kahekordse tsükliga;

Need kaks teemat on kahemõõtmelise järjendi peatükkides tavaliselt seletatud, kuid see, kuidas ning mis näidetega neid seletatakse on väga erinev. Indeksite selgitamiseks kasutatakse tihti tabelit, mis aitab teemat paremini mõista (vt joonis 1).

	Column 0	Column 1	Column 2
Row 0	scores[0][0]	scores[0][1]	scores[0][2]
Row 1	scores[1][0]	scores[1][1]	scores[1][2]
Row 2	scores[2][0]	scores[2][1]	scores[2][2]

Joonis 1. Kahemõõtmelise järjendi indeksite mõistmiseks tehtud tabel [6].

Õpikutes ning erinevatel kursustel käsitletavatest ülesannetest oli üsna levinud sudoku lahenduse kontrollimise ülesanne.

2.3 Vajalikud eelteadmised antud kursuse edukaks sooritamiseks

„Programmeerimise alused II“ kursuse läbimiseks on vaja osalejal eelteadmisi programmeerimisest ja Pythoni põhitõdedest. Kahemõõtmelise järjendi teema antud kursusel nõuab tingimuslause, tsükli, järjendite ja funktsioonide mõistmist. Antud teemad tuletatakse põgusalt meelde kursuse alguses kordamise osas, kuid kui need oskused on puudulikud või nõrgad, siis on üleminek üsna keeruline. Ühemõõtmelise järjendi mõistmine on samuti oluline, paljudes õpikutes ning materjalides ning ka antud kursusel tuuakse võrdluseks ühemõõtmeline järjend. Seda näiteks indeksitega elementide võtmise juures kui ka tsüklitega läbimise juures. Olgugi, et ühemõõtmelisest järjendist on kursuse materjalides juttu ja näiteid, võiks see olla siiski enne tuttav teema ning üsna arusaadav.

3. Ülevaade Tartu ülikooli arvutiteaduse instituudi korraldatavatest MOOCidest

3.1 Programmeerimisest maalähedaselt

„Programmeerimisest maalähedaselt“ oli esimene Tartu ülikooli arvutiteaduse instituudi poolt pakutav vaba juurdepääsuga e-kursus ehk MOOC (ingl k. *Massive Open Online Course*). Kursus toimus esimest korda 2014. aasta lõpus [9]. Kursus on 2017. aastaks toimunud juba 5 korda ning kokku on osalenud 5426 inimest. Nagu MOOCidele tavaks, siis suur osa alustajatest kursust ei lõpeta. Suuremas osas MOOCides on lõpetanuid alla 10% osalejatest [10]. „Programmeerimisest maalähedaselt“ kursuse teeb eriliseks ka see, et lõpetanute osakaal on väga suur. 5426 registreerunust on antud kursuse suutnud edukalt lõpetada 3465, mis teeb ligikaudu 64 protsenti.

Programmeerimisest maalähedaselt kursus kestab 4 nädalat. Kursus algab algoritmi mõiste tutvustamisest väga maalähedasel viisil, tuues näiteid igapäevaelust ning suunates osalejat mõtlema ka ise elulisi algoritme. Selle seletuse juurest liigutakse sujuvalt programmi mõisteni. Räägitakse ka muutujatest ja erinevatest andmetüüpidest. Teisel nädalal tuleb kursuse programmi valikulause temaatika. Kursusel käsitletakse veel tsüklit, kasutades näidetes ja ülesannetes vaid *while*-tsüklit. Õpetatakse funktsioone kasutama ning andmeid lugema nii veebist kui failist, lisaks ka andmete faili kirjutamist. Igal nädalal on kaks kohustuslikku ülesannet (v.a viimasel nädalal, siis on vaid 1) ning valikus on ka lisaülesanded, mida võib, aga ei pea lahendama. Igal nädalal tuleb lahendada ka Moodle’i test. Testid koosnevad kümnest küsimusest.

Nagu nimigi ütleb, on kursus esitatud maalähedases võtmes, mis tähendab, et keerukad programmeerimise mõisted ning struktuurid on seostatud igapäevaeluga. Näiteks võrreldakse muutujaid kursusel maitseaine topsidega [11]. Lihtsat ning elulist lähenemist programmeerimisele on üritatud hoida ka jätkukursustes. Püütud on mõelda ülesandeid, mis oleksid võimalikult elulised ning mida oleks huvitav lahendada.

3.2 Programmeerimise alused

„Programmeerimise alused“ kursus kestab kaheksa nädalat. Antud kursusel on kokku osalenud 3893 inimest, kellest 2102 on kursuse edukalt lõpetanud. Soovituslikult võiks enne kursusel osalemist olla läbitud „Programmeerimisest maalähedaselt“, kuid see ei ole kohustuslik ning

piisava pealehakkamise ja soovi korral on antud kursus võimalik edukalt läbida ka ilma eelneva programmeerimise kogemuseta. „Programmeerimise alused“ kursusel käsitletakse uuesti kõik teemasid, mis olid ka „Programmeerimisest maalähedaselt“ kursusel, kuid kõik veidi põhjalikumalt. Näiteks tsüklitest on nüüd kasutusel ka *for*-tsükkel. Lisaks on veel järjendi teema ja graafika Tkinteri abil. Ülesanded ja lahendused on veidi keerulisemad võrreldes „Programmeerimisest maalähedaselt“ kursusega. Kursuse lõpus on osalejatel võimalus teha kohapeal ka arvestustöö, mille eduka sooritusе korral on võimalik ülikoolis õppivatel või ülikooli astuvatel üliõpilastel ainepunktid kanda valikainete alla. 2017. aastal „Programmeerimise alused“ õpilastele suunatud kursusel arvestustöö edukas sooritamine tagab õpilastele koha Tartu ülikooli informaatika õppekavale.

3.3 Kursustel kasutatavad keskkonnad ning vahendid

Kursused on üles ehitatud Moodle’i ning Courses.cs.ut.ee keskkondi kasutades. Moodle on vabavaraline kursuse haldamise süsteem (ingl k. *Course Management System ehk CMS*), mida kasutavad üle 30 000 haridusasutuse üle maailma, et hallata e-kursuseid või toetada tavakursuseid (*face-to-face courses*)[12]. Antud e-kursustel tuleb Moodle’i keskkonnas kursuse läbimiseks esitada iganädalaseid arvestusülesandeid ning lahendada teste. Arvestusülesannete näol on tegemist programmeerimisülesannetega, mis tuleb kursusel osalejal Moodle’i keskkonda esitada. Ülesandeid võib esitada mitu korda, esitamise arv ei ole piiratud. Oluline on vaid tähtaeg, millal peavad kõik nädala ülesanded saama arvestatud. Ülesannete kontrollimiseks on automaatkontrollid, mis annavad osalejale tagasisidet, kas ülesanne oli õige või vale ning samuti on võimalik saada automaatkontrollilt infot, milles võib viga olla. Moodle keskkonnas olevad automaatkontrollid võimaldavad ülesandeid kontrollida, ilma, et korraldajad peaksid kõiki töid käsitsi üle vaatama.
Lisaks arvestuslesannetele on Moodle’is veel iganädalased testid. Test loetakse sooritatuks, kui õigeid vastuseid on vähemalt 90%. Moodle’i testi võib lahendada mitu korda ning puudub ka ajaline piirang, mille jooksul test tuleks ära lahendada.
Moodle’i keskkonnas on osalejatel võimalus näha oma tulemusi, ehk hoida pilku peal, kas kõik kohustuslikud arvestusülesanded ja testid on arvestatud.
Moodle’i keskkonnas on ka foorum, mille eesmärk on pakkuda digitaalset keskkonda sotsiaalseks suhtluseks ning, mis on abiks ja toeks õppe sisu paremaks mõistmiseks [13]. Foorumis on teemad jaotatud nädalate kaupa ning iga nädala all on erinevad alateemad. Foorum ei ole ainus võimalus abi küsimiseks, kursustel „Programmeerimisest maalähedaselt“ ja

„Programmeerimise alused“ on kasutatud ka abiliini. Abiliini näol on tegemist kohaga, kuhu inimesed saavad kirjutada, saata oma koodi ning küsida individuaalset tagasisidet. Abiliinil vastasid kursuse korraldajad ning nende töö oli jagatud valvetesse. Valved olid ööpäeva ringselt vastavalt 10.00-22.00 päevane ja 22.00-10.00 öine vahetus. Suure koormuse ning vastamise ajakulu tõttu on püütud abiliini võimaluse pakkumist vähendada ning „Programmeerimise alused II“ kursusel seda võimalust enam ei ole. Abiliini tööd aitas vähendada ka Vello Vaherpuu bakalaureusetööna valminud murelahendaja [14]. Murelahendaja näol on tegemist programmiga, kus on arvestusülesannete enamlevinud probleemid välja toodud ning antud vihjed nende lahendamiseks. Murelahendaja eesmärk on suunata hätta jäänud kasutajat õigele teele.

Kursuse õppematerjalid asuvad Courses.ut.ee keskkonnas, kus igal nädalal ilmuvad nähtavale vastava nädala õppematerjalid. Materjalide sisse on põimitud kontrollküsimused, mis kaasavad üliõpilast rohkem materjalidesse süvenema ning annavad tagasisidet, kas materjalidest on aru saadud. Kontrollküsimused on klikitavad ning igale vastusevariandile on lisatud ka tagasiside, kas vastus oli õige või vale, mille juurde kuulub ka põhjendus. Kui vastatud on õigesti, siis ilmub rohelist värvi tagasiside ning vale vastuse korral on tagasiside punane. Kontrollküsimused teksti vahel, teevad teksti paremini jälgitavaks ning küsitlused on näidanud, et osalejatele need väga meeldivad. Courses.ut.ee keskkonnast leiab ka lisamaterjale lugemiseks.

4. „Programmeerimise alused II“ materjalide valmimise protsess

4.1 Programmeerimise alused II

„Programmeerimise alused II“ kursusest on 3 erinevat varianti:

1. Statsionaarne kursus,
2. Segu statsionaarsest kursusest ning e-kursusest,
3. Vaba juurdepääsuga e-kursus ehk MOOC,

Kõik kolm varianti on oma ülesehituselt veidi erinevad, kuid teemad, mida käsitletakse on üldjoontes samad. 2016. aasta kevadsemestril toimunud statsionaarsel kursusel olid nii loengud kui ka praktikumid ja õppematerjalidena kasutati Aivar Annamaa programmeerimise õpikut [15].

Käesoleva bakalaureusetöö raames on analüüsitud teist varianti, ehk kursust, mis oli segu statsionaarsest kursusest ning e-kursusest. Kursusel kasutati e-kursuse materjale ning üle nädala toimusid ka auditoorsed loengu ning lisapraktikumid, kus oli võimalik saada täiendavat infot, kui materjalides oli puudusi või teemad olid jäänud segaseks. Kursus oli jaotatud perioodideks ning iga perioodi lõpus tuli üliõpilastel esitada lisaks kohustuslikele programmeerimisülesannetele ja testidele ka tagasiside ülesannete, testide ja materjalide kohta. Kursus algas 20. oktoobril 2016 ning kursusel osalesid „Infotehnoloogia mitteinformaatikutele“ õppekava üliõpilased, kes olid osalenud eelnevalt „Programmeerimise alused“ e-kursusel. „Infotehnoloogia mitteinformaatikutele“ on magistriõppekava, mis loodi 2016. aastal suurenenud huvi tõttu infotehnoloogia valdkonna vastu. Õppekava eesmärk on arendada vajalikke teadmisi ja oskusi inimestes, kellel on huvi ja soov töötada IT-valdkonnas või toetada infotehnoloogiliste lahendustega oma olemasolevaid tööülesandeid, kuid varasemad teoreetilised teadmised puuduvad [16]. Õppekavas oli kohustuslik kursus „Programmeerimine“, mis sisu poolest on „Programmeerimise alused“ ja „Programmeerimise alused II“ kursused kokkuvõetult. Kuna 17 inimest olid „Programmeerimise alused“ e-kursusena juba läbinud, siis neil oli võimalus võtta „Programmeerimine“ (6 EAP) asemel Programmeerimise alused II“ (3 EAP). Antud kursus oli suurepärane võimalus analüüsida e-kursuse tarbeks loodavaid materjale, kuna osalejateks on „Programmeerimise alused“ e-kursusena läbinud üliõpilased, seega neil on sarnane programmeerimise kogemus nagu suuremal osal e-kursuse „Programmeerimise alused II“ eeldatavatest osalejatest. Samuti võimaldab sellist tüüpi kurus ka ise kohapeal näha, millega võib probleeme tulla e-kursuslastel, kellel pole võimalik individuaalselt küsida abi.

„Programmeerimise alused II“ vaba juurdepääsuga e-kursus toimus esimest korda 3. aprill – 28. mai 2017, kus osalejaid oli üle 900 [17]. Kursusel on kasutusel samad keskkonnad ning vahendid nagu eelnevatel e-kursustel. Kursuse sisuline pool koosneb kolmest põhilisest teemast:

1. Kahemõõtmeline järjend
2. Rekursioon
3. Andmestruktuurid

Esimeseks uueks teemaks kursusel on kahemõõtmelise järjendi temaatika, kus alustatakse üldise selgitusega, mis on kahemõõtmeline järjend ning seejärel kuidas sealt vajalikku infot kätte saada. Selgitatakse, kuidas kahemõõtmelisest järjendist infot kätte saada nii indeksitega kui ka kahekordse tsükliga. Seletatakse ka maatriksi mõiste. Antud temaatika all on käsitletud ka CSV failist andmete lugemine ning nende töötlemine. Materjalid koosnevad paljudest näidetest ning ka selles kursuses, nagu eelnevateski e-kurustes, on materjalide vahele põimitud kontrollküsimused. Kursusel on ka Moodle’i testid, mille läbimise tingimused on jällegi samad nagu eelnevatel kursustel ehk arvestuse saamiseks tuleb 90% õigesti vastata. Igal nädalal tuleb lahendada arvestusülesanded.

4.2 Materjalide loomine ning testimine

Kursuse materjalid valmisid suuresti koostöö käigus. Bakalaureusetöö autori põhiline roll oli mõelda välja kontrollküsimusi kahemõõtmelise järjendi materjalide toetamiseks, mõelda välja antud teema kohta Moodle’i testide küsimused, pakkuda ideid arvestusülesannete jaoks ning mõelda eksami harjutusülesandeid. Samuti sai pisut kätt proovitud materjalide loomisel.

Kontrollküsimuste loomise juures tuli jälgida materjale ning püüda mõelda materjalidele tuginedes selliseid küsimusi, mis toetaksid ning annaksid materjalidele lisandväärtust. Kontrollküsimusi tuli mõelda päris palju, kahemõõtmelise järjendi teema toetamiseks sai loodud 18 kontrollküsimust. Antud küsimuste vastusevariante mõeldes tuli lähtuda üliõpilase mõtlemisest, kellel ei ole teema veel päris selge, et valed vastusevariandid tunduksid loogilised. Kõikidele vastusevariantidele tuli mõelda asjalik tagasiside, kus vale vastuse juures ei oleks kirjas lihtsalt „Vale!“, vaid millel oleks juures ka põhjendus, miks see vastus on vale.

Bakalaureusetöö autor koostas kaks kahemõõtmelise järjendi teemalist Moodle’i testi. Mõlemas testis oli viis küsimust. Nende loomise juures oli keeruline see, et kuna suur hulk

kahemõõtmelise järjendi teemalisi küsimusi oli juba loodud kontrollküsimuste näol, siis tuli mõelda välja midagi uut. Tuli mõelda, mida ei ole enne küsitud ning, mis oleks veidi keerulisemad, kui materjalides esinevad küsimused.

Ka ülesannete loomine osutus üsna keeruliseks. Kuna eelnevatel kursustel on ülesanded olnud üsna elulised, siis oli ka soov mõelda selles kursuses päris probleeme, mis on elulised ning millega inimesed võiksid suhestuda. Samuti tahtis bakalaureusetöö autor mõelda välja ülesandeid, mis ei oleks teistel kursustel juba kasutuses. Bingoloto tundus teema, mis võiks sobida, kuna võimalik on luua väga erinevaid selleteemalisi ülesandeid. See tundus teema, mida inimesed teavad ning on kasvõi korra mänginud või kokku puutunud. Kuid nagu analüüsist selgus, siis see teema ning ülesanded ei olnud just kõigile väga meeltmööda ning selleteemalisi ülesandeid oli natuke liiga palju. Kasutatud sai ka juba statsionaarses kursuses kasutusel olnud ülesandeid, üheks näiteks sudoku lahenduse kontroll, mis on üsna levinud kahemõõtmelise järjendi ülesanne. Ülesannete loomise juures oli kõige keerulisem oma idee kirja panna nii, et see oleks üheselt mõistetav ja arusaadav kõigile. Kui isegi tundus, et ülesanne on arusaadav ja sellega ei tohiks probleeme tulla, siis analüüsides ja tagasisidesid lugedes tuli ikka välja palju ülesandeid, kus oleks püstitust vaja täiendada.

Antud materjalide analüüsimiseks saadud tagasiside pärines „Infotehnoloogia mitteinformaatikutele“ õppekava üliõpilastelt. Neil oli võimalus kirjutada tagasisidet nii materjalide arusaadavuse, ülesannete kui ka testide kohta.

4.3 Kursuse korraldamise keerukus võrreldes eelnevate kursustega

„Programmeerimise alused II“ kursusel käsitletavad teemad on oluliselt keerukamad võrreldes „Programmeerimisest maalähedaselt“ või „Programmeerimise alused“ kursusega. Ülesannete lahendused on pikemad ning ei ole enam üheselt lahendatavad. Ühe ülesande lahendamiseks on mitu erinevat võimalust, mis muudab keeruliseks automaatkontrollide loomise. Samuti muutub oluliselt keerulisemaks eelnevates kursustes suurt rolli mänginud murelahendaja loomine. Murelahendaja ülesandeks on suunata hätta jäänud üliõpilasi õigele teele, aga kui ülesannet saab lahendada mitut erinevat viisi, mis on kõik õiged, siis võib suunamine hoopis segadusse ajada.

Mida raskem on kursus, seda väiksemaks muutub kursuse lõpetanute protsent. Kui „Programmeerimisest maalähedaselt“ kursusel läbis 64% osalejatest, siis „Programmeerimise alused“ kursusel oli juba veidi vähem ehk 54%. On üsna tõenäoline, et antud kursuse

„Programmeerimise alused II“ läbimisprotsent on veel väiksem, seda enam, et kursusel puudub abiliini võimalus. Osalejad ei saa enam oma koodi saata ega hätta jäädes küsida individuaalset tagasisidet.

5. Kahemõõtmeliste järjendite materjalide analüüs põhinedes „Infotehnoloogia mitteinformaatikutele“ tagasisidel

Antud jaotises analüüsitakse magistriõppekava „Informaatika mitteinformaatikutele“ üliõpilaste tagasisidet. Vaatluse all on:

1. Ülesannete lahendused,
2. Lahendatud testid ning lahendamiseks kulunud aeg,
3. Üliõpilaste endi kirjutatud tagasiside materjalide kohta,

Iga perioodi lõpuks tuli üliõpilastel anda tagasiside arvestusülesannete, kontrollküsimuste ning õppematerjalide arusaadavuse kohta. Tagasiside esitati Moodle’i kaudu. Esimesel perioodil esitasid kõik 17 kursusel osalenud üliõpilast tagasiside, kuid teisel perioodil esitati 16 tagasisidet. Käesolevas peatükis analüüsitakse materjale koos kordamisküsimustega, programmeerimisülesandeid ning Moodle’i teste.

5.1 Materjalid/Kontrollküsimused

Materjalide kohta kirjutatud tagasiside oli üsna positiivne. Kirjutati, et materjalid on toetavad ja igati arusaadavad. Lisaks toodi välja, et meeldib väga materjalide esitamise viis ning kontrollküsimused materjalide sees. Enamus tõid välja, et kontrollküsimused on väga head materjalide toetamiseks ning neid võiks olla ka edaspidi palju. Need, kellele jäi antud materjalidest väheks, meeldis lugeda lisaks ka Aivar Annamaa programmeerimise õpikut [15].

5.2 Arvestusülesanded

5.2.1 Esimese perioodi ülesanded

Esimese perioodi ülesannete läbivaks teemaks oli bingoloto. Perioodi lõpuks tuli osalejatel lahendada 3 ülesannet:

1. „Reeglite kontrollimine“;
2. „Kas on võitnud?“;
3. „Võiduks veel vaja“;

Reeglite kontrollimise ülesandes tuli vaadata üle kõikides veergudes olevad arvud ning kontrollida, kas arvud vastavad bingo mänguvälja reeglitele ehk esimeses tulbas olevad arvud peavad olema vahemikus 1-15, teises 16-30 ja nii edasi. Antud ülesannet oleks võimalik

lahendada mitut erinevat moodi. Üks võimalus on kasutada alampiiri, mis tsükli sees suureneb 15 võrra.

```
def on_legaalne_bingo_tabel(tabel):
    alampiir = 1  #Määrame alampiiri
    for j in range(5):  #Välimine tsükkel läbib veerge
        for i in range(5):  #Sisemine tsükkel läbib ridu
            if tabel[i][j] < alampiir or tabel[i][j] > alampiir+14: #Kas leidub element, mis ei jää antud vahemikku
                return False #Kui leidub tagastame kohe False ning edasi ei ole vaja vaadata
        alampiir += 15 #Suurendame alampiiri, et järgmises tulbas oleks vahemik 15 võrra suurem
    return True #Kui kõik elemendid on läbitud ning if lause sisu pole täidetud, siis tagastatakse True
```

Näide 1. „Reeglite kontrollimine“ lahendus

Tegelikult võib ka kohe ülempiiri ära määrata või hoopis lahendada ilma alampiiri ja ülempiiri muutujateta, kasutades nende leidmiseks tulba indeksit. Lahendati üsna erinevalt, leidus häid lahendusi, kuid leidus ka lahendusi, kus olid kõik vahemikud käsitsi välja kirjutatud.

```
if rida[0] > 15 or rida[1] > 30 or rida[1] < 16 or rida[2] > 45 or rida[2] < 31
or rida[3] > 60 or rida[4] < 46 or rida[4] > 75 or rida[4] < 61:
```

Näide 2. Ülesande „Reeglite kontrollimine“ if lause

Antud ülesande puhul kõige mõistlikum olnuks otsida juhtu, mis rikub reeglit, ehk arvu, mis jääb piiridest välja. Kui see tehtud, siis tagastada *False* ning rohkem ei peaks elemente läbima. Leidus lahendusi, kus oli tehtud vastupidi, *return* oli ainult kõige lõpus, kui kogu tabel oli läbi vaadatud. Samuti oli mõnel juhul lisatud koodi ridu, mis tegelikult olid ebavajalikud

```
def on_legaalne_bingo_tabel(numbrid):
    read = []

    for rida in numbrid:
        if rida[0] in range(1, 16):
            if rida[1] in range(16, 31):
                if rida[2] in range(31, 46):
                    if rida[3] in range(46, 61):
                        if rida[4] in range(61, 76):
                            read.append("korras")

    if read.count("korras") != 5:
        return False
    else:
        return True
```

Näide 3. „Reeglite kontrollimine“ ülesande üks esitatud lahendus

„Kas on võitnud“ ülesandes oli antud 5x5 tabel milles iga element on kas täisarv lõigust 1-75 või, kui arv on juba loositud, siis täht X. Ülesandeks on kontrollida, kas on võidetud

nurkademäng, diagonaalidemäng või täismäng. Nurkademängu leidmise osa oli kõige lihtsam ning sellega ei paistnud kellelgi probleeme olevat. Diagonaalidemängu osa hõlmas endas kolme funktsooni. Tuleb leida mitu X-i on peadiagonaalil, mitu kõrvaldiagonaalil ning seejärel kolmandas funktsioonis toimub kontroll, kas diagonaalidel oli võitmiseks vajaminev arv X-e.

Ülesandes „Võiduks veel vaja“ tuli leida nii nurkademängu, diagonaalidemängu kui ka täismängu võidust puuduolevad numbrid. Kui üldiselt oli ülesanne üsna hästi lahendatud, siis kõige keerulisemaks osutus diagonaalidemängu osa. Seal ilmnes väga palju ebaratsionaalseid lahendusi. Selle asemel, et ülesanne lahendada kahekordse tsükliga, lahendati ülesanne kõik diagonaalide indeksid käsitsi välja kirjutades (vt Näide 4).

```
def diagonaalidemänguks_vaja(tabel):
    diagonaal = []
    for i in range(1):
        if tabel[0][0] != "X":
            diagonaal.append(tabel[0][0])
        if tabel[1][1] != "X":
            diagonaal.append(tabel[1][1])
        if tabel[2][2] != "X":
            diagonaal.append(tabel[2][2])
        if tabel[3][3] != "X":
            diagonaal.append(tabel[3][3])
        if tabel[4][4] != "X":
            diagonaal.append(tabel[4][4])
        if tabel[4][0] != "X":
            diagonaal.append(tabel[4][0])
        if tabel[3][1] != "X":
            diagonaal.append(tabel[3][1])
        if tabel[1][3] != "X":
            diagonaal.append(tabel[1][3])
        if tabel[0][4] != "X":
            diagonaal.append(tabel[0][4])
    return diagonaal
```

Näide 4. „Võiduks veel vaja“ lahenduseks esitatud diagonaalidemängu funktsioon

Põhinedes üliõpilaste tagasisidele oli antud nädal enamuse jaoks üsna raske peamiselt kahel põhjusel. Esiteks, kuna kahemõõtmeline järjend oli kursuse esimene teema, ning paljudel üliõpilastel oli mitu kuud möödas viimasest programmeerimisest, olid Pythoni põhitõed läinud meelest ning kursus algas nende jaoks väga kiire tempoga. Teine probleem oli see, et kuna nädala põhiteemaks oli bingoloto ning kõik ülesanded olid sellega seotud, siis oli inimesi, kes ei teadnud bingoloto reegleid ning jäid selle taha toppama.

Vaatamata kirjeldatud probleemidele ning raskustele said esimese perioodi kohustuslikud ülesanded lahendatud kõigil osalejatel. Lahendusi oli üsna erinevaid. Kuna automaatkontroll ei seadnud piiranguid ülesannete ebaratsionaalsuse vältimiseks siis oli ka selliste ülesannetega võimalik saada täispunktid.

5.2.2 Teise perioodi ülesanded

Teisel perioodil oli jällegi 3 kahemõõtmelise järjendi teemalist ülesannet:

1. „Vähimatest suurim“;
2. „Sudoku lahenduse kontrollimine“;
3. „Juhuslik bingo tabel“;

Teise perioodi ülesannetena on kasutatud osaliselt „Programmeerimise alused II“ statsionaarses õppes olnud ülesandeid. Üheks selliseks on ülesanne „Vähimatest suurim“, kus tuleb täisarvude maatriksist kõikide ridade vähimatest elementidest leida suurim. Ka sudoku lahenduse ülesanne on olnud kasutusel statsionaarses kursuses. Käesoleval kursusel oli ülesanne tehtud veidi lihtsamaks ning funktsioon, mis kontrollib, et sudoku kastides ei oleks korduvaid numbreid, oli juba ette antud.

Ülesanne „Vähimatest suurim“ oli mõeldud lahendamiseks kahekordse tsükliga, kuid seda oli võimalik väga edukalt lahendada ka vaid ühte tsüklit kasutades. 10 inimest 17-st selle ka nii lahendasid. Sudoku ülesande lahendusi analüüsides tuli välja, et automaatkontroll vajas parandamist. Kuna ülesanne oli tehtud lihtsamaks ning kastide kontrollimise funktsioon oli antud, andis vaid selle esitamine juba automaatkontrollis täispunktid. Lühend „mat“, mida kasutati antud funktsioonis muutujana, tekitas paljudele segadust. Tagasisides kirjutasid mitu inimest, et üritasid seda alguses importida ja ei saanud alguses üldse aru, millega on tegemist. Antud perioodis oli ka üks bingo ülesanne ning tagasisides selgus, et bingo temaatika osalejatele väga ei meeldinud. Kuna terve esimene nädal oli bingoloto ülesandeid täis, siis teisel nädalal toodi välja, et hea oli vaheldust saada bingo ülesannetest.

Üldine tagasiside teise perioodi kohta oli juba märksa parem kui esimene. Teise perioodi tagasisidedes toodi välja, et vilumus kasvab ning antud ülesanded tundusid lihtsamad ning võtsid vähem aega, kui esimesel perioodi.

5.2.3 Eksamiülesanded ja harjutusülesanded

„Infotehnoloogia mitteinformaatikutele“ õppekava üliõpilastel tuli lahendada ka antud kursuse läbimiseks eksam. Eksam koosnes kahest osast, kus esimene osa tuli lahendada paberil ning

teine arvutis. Kursuse edukaks läbimiseks tuli mõlemas osas saada vähemalt pooled punktid. Eksamiülesanded koostas kursuse õppejõud Eno Tõnisson. Eksami arvutiosa koosnes kolmest ülesandest. Esimene oli kahemõõtmelise järjendi teemaline, teine rekursiooni käsitlev ning kolmandaks andmestruktuure puudutav ülesanne. Kahemõõtmelise järjendi ülesanne oli kõige aeganõudvam ja andis ka rohkem punkte kui teised.

Lisaks eksamil olnud ülesannetele tuli mõelda ka harjutusülesandeid. Kahemõõtmelise järjendi harjutusülesanded mõtles bakalaureusetöö autor. Mõeldud sai seitse harjutusülesannet, millest neli läks tudengitele lahendamiseks. Üliõpilaste kirjutatud tagasisides selgus, et neljast ülesandest kaks jäid püstituse poolest ebaselgeks ning vajaksid kindlasti põhjalikumat ning täpsemat sõnastust. Kuna neid ülesandeid ei pidanud esitama, siis ülesannete lahendusi ei olnud võimalik analüüsida. Lisaks neljale arvutiosa ülesandele mõtles bakalaureusetöö autor ka ühe paberosa ülesande kahemõõtmelise järjendi kohta. Ülesande idee oli võetud statsionaarse kursuse eksamist. Ette oli antud programm ning lisaks sellele ka programmijupp, milles on funktsioon välja kutsutud. Ülesandes tuleb kirjutada ise funktsiooni sisu nii, et programm töötaks ülaltooduga võrdväärselt.

5.3 Moodle'i testid

Kahemõõtmelise järendi teemalisi Moodle'i teste oli antud kursusel kaks. Mõlemas testis oli viis küsimust. Testi lahendamisel ei olnud ajalist piirangut ning testi võis lahendada piiramatu arv kordi. Testi läbimiseks oli vaja 90% õigeid vastuseid. Kuna testi lahendamisel ei olnud ajalist piirangut, siis oli näha statistikas ka väga ebarealistlikke testi lahendamiseks kulunud aegu. Näiteks esimese testi lahendamiseks kulunud pikim aeg oli 1 päev ja 2 tundi, mis ei saa olla 5 küsimuse vastamiseks reaalselt kulunud aeg. Testi on võimalik jätta pooleli ning hiljem ära lõpetada, seega keskmise ajakulu leidmiseks on eemaldatud üle kahe tunni kulunud ajad. 2 tundi sai valitud just seetõttu, et paistis silma, et sealt edasi läks hüpe suuremaks ning järgmine aeg oli juba 11 tundi. Aja arvestusse võeti vaid esimese katse aeg, kuna testi võis lahendada mitu korda ning küsimused olid teisel katsel samad.

1. perioodi testi statistika

Esimese katse kiireim aeg	22 minutit 49 sekundit
Esimese katse aeglaseim aeg	1 päev 2 tundi
Keskmine ajakulu, millest välja jäetud üle 2h kulunud ajad	47 minutit
Esimese korraga maksimaalse soorituse teinud inimeste arv	10 inimest
Küsimus, milles esines kõige rohkem vigu	1. küsimus

Esimese nädala testi lahendasid edukalt kõik 17 kursusel osalenud üliõpilast. Testi kõige probleemsemaks küsimuseks oli esimene küsimus, kus tuli koodis lünk täita. Keeruliseks tegi küsimuse see, et tegemist oli lünkteksti väljaga ning sellisel juhul on Moodle'is võimalik ette anda teatud arv vastuseid, mis peavad olema täpselt samad kasutaja sisestatud vastusega. Kasutajal võis olla vastuse idee õige, aga tühiku pärast loeti vastu testis valeks. Seega selle ülesande juures tuli olla eriti tähelepanelik. Esimese perioodi tagasisides toodi välja ka antud testi küsimuse üks murekoht, millele testi koostaja ei mõelnud: „Olin hädas |-märgi leidmisega, kuna kopeerida seda ei saa ja klaviatuurilt sain alles pärast tunniajast proovimist". Teise ning kolmanda küsimuse antud testis vastasid kõik osalejad esimese korraga õigesti. Antud küsimused olid koodipõhised ning neid oli võimalik lihtsa vaevaga ise proovida. Tuli märkida linnuke õige tulemuse andnud programmi või koodijupi ette.

2. perioodi testi statistika

Esimese katse kiireim aeg	6 minutit 32 sekundit
Esimese katse aeglaseim aeg	1 tund 13 minutit
Keskmine ajakulu	30 minutit
Esimese korraga maksimaalse soorituse teinud inimeste arv	16 inimest
Küsimus, milles esines kõige rohkem vigu	1. küsimus

Teise perioodi kahemõõtmelise järjendi test tundus osalejatele lihtsam. Jällegi kõik 17 kursusel osalejat said testi edukalt lahendatud. Peaaegu kõik lahendasid esimese korraga testi maksimaalsele punktide arvule ning ka keskmine ajakulu oli lühem. Selle testi juures utoopilisi

11 tunnilisi lahendusi ja suuremaid ei olnud, kõige kauem lahendati testi 1 tund ja 13 minutit, seega nende andmete juures midagi välja ei jäänud. Ainuke viga tehti testi esimeses küsimuses, kus oli antud ette programm ning programmijupp, kus oli osa koodist asendatud funktsiooniga. Ülesandeks oli valikvastuse seast valida, missugune funktsioon sobiks programmilõiku, et töötaks samamoodi nagu esialgne programm. Tegemist oli jällegi ülesandega, mida oli võimalik lihtsa vaevaga ka ise järgi proovida.

6.Kokkuvõte

Antud bakalaureusetöö eesmärk oli luua ning analüüsida ka teiste poolt valmistatud materjale, mida kasutada „Programmeerimise alused II“ vaba juurdepääsuga e-kursuse kahemõõtmelise järjendi ning kahekordse tsükli temaatika juures. Materjalid valmisid koostöö käigus ning üksteise loodud mõtteid ning ideid täiendati ja parandati vastavalt vajadusele. Bakalaureusetöö autor koostas kaks Moodle'i testi, kus kummaski oli viis küsimust. Testi lahenduste analüüsimisel selgus, et testid olid jõukohased ning kõik osalejad said testi lahendatud. Kõige problemaatilisemaks ülesandeks oli esimese testi esimene küsimus, milles oli kõige rohkem eksimusi. Antud küsimus nõudis koodi täiendamist ning ainuüksi tühiku viga võis vastuse valeks lugeda. Seega tuli selle küsimuse juures olla eriti tähelepanelik. Antud küsimus on välja toodud ka lisades. Üldisest tagasisidest selgus, et testi küsimustega suuremaid probleeme ei esinenud.

Lisaks sellele koostas bakalaureusetöö autor veel ka kahemõõtmelise järjendi materjalide toetamiseks kontrollküsimused. Kontrollküsimused on materjalide sisse põimitud küsimused, millel on valikvastused ning nendele vajutamisel on võimalik näha, kas vastus oli õige või vale. Lisaks on veel tagasiside vastajale, miks antud vastus on vale. Kontrollküsimused on ennast juba tõestanud eelnevatel kursustel ning need on inimestele väga meeldinud. Antud kursusel ei õnnestunud saada põhjalikumat tagasisidet enda loodud kontrollküsimuste sisulisele poolele, kirjutati vaid, et kontrollküsimused on väga head ning toetavad materjale.

Kahemõõtmelise järjendi arvestusülesandeid oli kuus. Esimese perioodi arvestusülesannete idee, milleks oli bingoloto, oli bakalaureusetöö autori mõeldud, kuid need ülesanded olid vormistatud teiste korraldajate poolt. Kui esialgu tundus temaatika hea, siis tagasisides selgus, et bingolotoga ei ole väga paljud inimesed kursis ning ülesannete püstitused jäid arusaamatuks. Samuti selgus, et inimestele pigem ei meeldi, kui mitu ülesannet on ühe teemalised. Osalejad olid rõõmsad, kui teisel perioodil oli ka teistsuguse teemalisi ülesandeid. Teisel perioodi kahemõõtmelise järjendi ülesannetena sai kasutatud ka olemasoleva statsionaarse kursuse kahte ülesannet. Üheks oli „Sudoku lahenduse kontrollimine“, milles oli ette antud kastide kontrollimise funktsioon. Ülesandeid analüüsides selgus, et selle ülesande automaatkontroll vajab kindlasti täiendamist, sest läbi läksid lahendused, kus oli esitatud vaid etteantud kood. Üldiselt olid arvestusülesanded osalejatele jõukohased. Ajakulu oli esimesel perioodil suurem, kuna inimesed ei olnud veel sisse elanud ning Pythoni põhitõed olid läinud meelest. Antud bakalaureusetöö autor mõtles välja antud teema kohta eksami harjutusülesandeid. Neid mõtles bakalaureusetöö autor seitse, kuid kasutusele läks neist neli. Neid ülesandeid ei pidanud

esitama, seega lahendusi ei olnud võimalik analüüsida, kuid tagasisides tuli välja, et kahe ülesande tekst oli jäänud väga ebaselgeks ning vajaks kindlasti täpsustamist ning paremat sõnastamist.

„Programmeerimise alused II“ vaba juurdepääsuga e-kursus algas 3.aprill 2017 ning kursusel osalejaid oli üle 900. Sellel kursusel on materjale täiendatud ning parandatud.

7. Kasutatud materjalid

[1] Tartu ülikooli arvutiteaduse instituudi e-kursus. (2016). Programmeerimise alused. Pilootkursus 2016. URL https://courses.cs.ut.ee/2015/eprogalused/fall (3.05.2017)

[2] Data structures. (2017). URL http://www.bbc.co.uk/education/guides/z4tf9j6/revision/3 (3.05.2017)

[3] Serre D. (2010). Matrices. Springer New York. pp. 15

[4] Zhou Yunhong jt. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. URL https://link.springer.com/chapter/10.1007/978-3-540-68880-8_32 (3.05.2017)

[5] Hall T., Stacey, J-P. (2009). Python 3 for Absolute Beginners. Apress. 83-85

[6] Gaddis T. (2011). Starting Out with Python, 2nd edition. Pearson: 2 edition. 328-332

[7] Liang Y. D. (2013) Introduction to Programming Using Python. Pearson: 1 edition. 361-398

[8] Lambert K. A. (2014). Fundamentals of Python: Data Structures. Cengage Learning PRT; 1 edition. 104-107

[9] Tõnisson, E. (2015). Programmeerimisest maalähedaselt. Juured. URL http://etu.ut.ee/kevad-2015/programmeerimisest-maalahedaselt/ (3.05.2017)

[10] Jordan, K. (2014). Initial Trends in Enrolment and Completion of Massive Open Online Courses. URL http://www.irrodl.org/index.php/irrodl/article/view/1651/2774 (3.05.2017)

[11] Tartu ülikooli arvutiteaduse instituudi e-kursus. (2015). Programmeerimisest maalähedaselt. 2014/15 kevad. URL https://courses.cs.ut.ee/2015/progmaa/spring/Main/HomePage (3.05.2017)

[12] Cole, J., Foster, H. (2008). Using Moodle: Teaching with the Popular Open Source Course Management System. 2nd edition. O'Reilly Media, Inc.

[13] Schmidt, J. (2015). MOOC kursustel õppimise kogemus. URL http://andragoogika.tlu.ee/?page_id=796 (3.05.2017)

[14] Vaherpuu V. (2016). Murelahendaja loomise keskkond. Tartu Ülikooli arvutiteaduse instituut, bakalaureusetöö

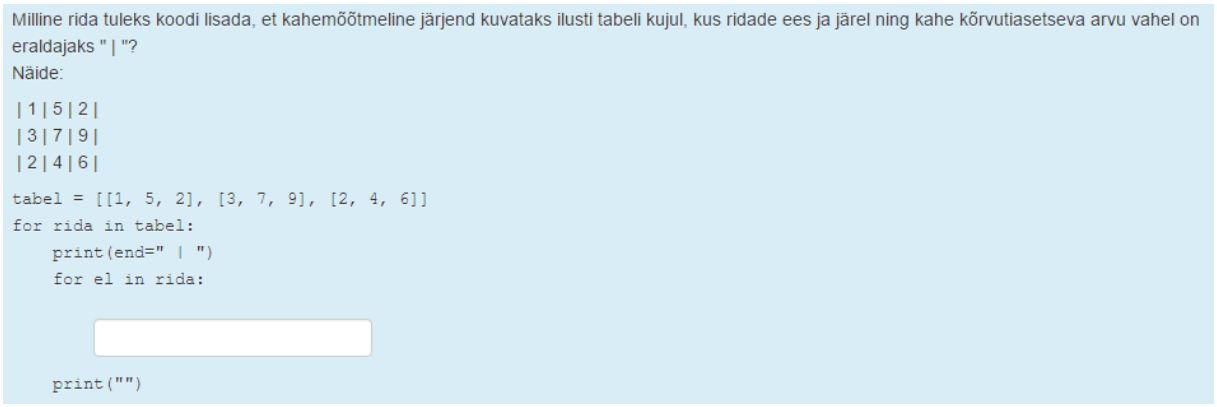
[15] Annamaa A. Programmeerimise õpik. URL http://progeopik.cs.ut.ee/ (3.05.2017)

[16] Kula, K. (2016). TÜ avab IT magistriõppekava mitteinformaatikutele. *Tartu Postimees, 24. märts*. URL http://tartu.postimees.ee/3630183/tu-avab-it-magistrioppekava-mitteinformaatikutele (3.05.2017)

[17] Tartu ülikooli arvutiteaduse instituudi e-kursus. 2017. Programmeerimise alused II. 2016/17 kevad URL https://courses.cs.ut.ee/2017/eprogalused2/spring/Main/HomePage (3.05.2017)

Lisad

Lisades on toodud näited Moodle'i testi küsimustest, eksami harjutusülesannetest ning materjalides olevatest kontrollküsimustest. Kõik toodud ülesanded on bakalaureusetöö autori koostatud.


Milline rida tuleks koodi lisada, et kahemõõtmeline järjend kuvataks ilusti tabeli kujul, kus ridade ees ja järel ning kahe kõrvutiasetseva arvu vahel on eraldajaks " | "?
Näide:

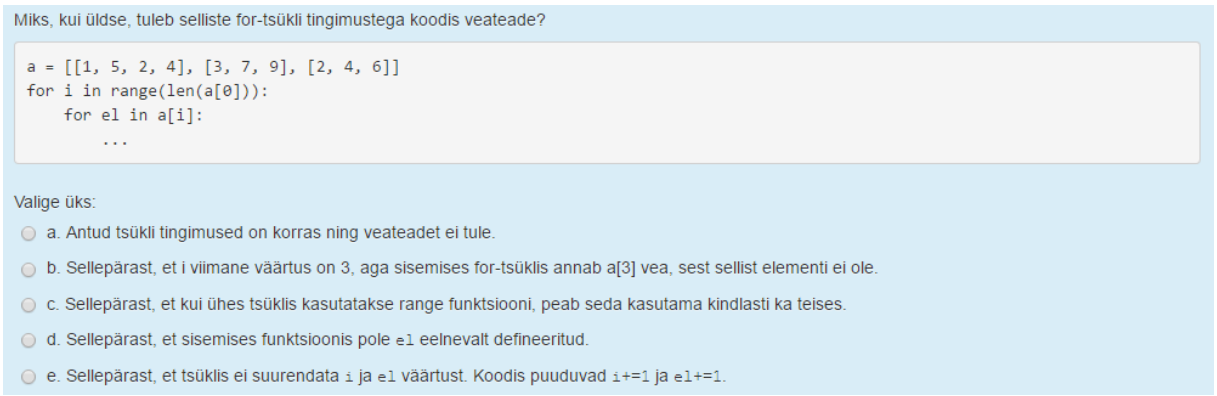
1	5	2
3	7	9
2	4	6

```
tabel = [[1, 5, 2], [3, 7, 9], [2, 4, 6]]
for rida in tabel:
    print(end=" | ")
    for el in rida:

    print("")
```

Esimese perioodi Moodle'i testi esimene küsimus


Miks, kui üldse, tuleb selliste for-tsükli tingimustega koodis veateade?

```
a = [[1, 5, 2, 4], [3, 7, 9], [2, 4, 6]]
for i in range(len(a[0])):
    for el in a[i]:
        ...
```

Valige üks:

- a. Antud tsükli tingimused on korras ning veateadet ei tule.
- b. Sellepärast, et i viimane väärtus on 3, aga sisemises for-tsüklis annab a[3] vea, sest sellist elementi ei ole.
- c. Sellepärast, et kui ühes tsüklis kasutatakse range funktsiooni, peab seda kasutama kindlasti ka teises.
- d. Sellepärast, et sisemises funktsioonis pole `el` eelnevalt defineeritud.
- e. Sellepärast, et tsüklis ei suurendata `i` ja `el` väärtust. Koodis puuduvad `i+=1` ja `el+=1`.

Esimese perioodi Moodle'i testi neljas küsimus

Mis ilmub ekraanile?

```
a = ["JavaScript", "Java", "PHP", "Python", "C++", "Ruby"]

for i in range(len(a)):
    keel = ""
    for j in range(len(a[i]) - 1, -1, -1):
        keel += a[i][j]
    a[i] = keel

print(a)
```

Valige üks:

- a. `['JavaScript', 'Java', 'PHP', 'Python', 'C++', 'Ruby']`
- b. `['tpircSavaJ', 'avaJ', 'PHP', 'nohtyP', '++C', 'ybuR']`
- c. `['Ruby']`
- d. Veateade

Teise perioodi Moodle'i testi neljas küsimus

Ülesanne A1 (? punkti)

Kirjutada funktsioon, mis võtab argumendiks 2 maatriksit ning kontrollib, kas üks on saadud teisest transponeerimise teel. (Maatriksid ei pruugi olla ruutmaatriksid.)
https://et.wikipedia.org/wiki/Transponeeritud_maatriks

Näide tööst:

Maatriksite
```
maatriks_1 = [[1, 2], [3, 4], [5, 6]]
maatriks_2 = [[1, 3, 5], [2, 4, 6]]
```
puhul peaks funktsioon tagastama **True**

Kutsudes välja funktsiooni maatriksitega
```
maatriks_3 = [[1, 2], [1, 1]]
maatriks_4 = [[1, 2, 3], [2, 2, 3]]
```
peaks funktsioon tagastama **False**.

Eksami arvutiosa esimene harjutusülesanne

Ülesanne A2 (? punkti)

Kirjutada funktsioon *tulba_pikimad*, mis võtab argumendiks kahemõõtmelise järjendi, mille elementideks on sõnad (sõned). Võib eeldada, et kõikides ridades on sama palju sõnu. Funktsioon leiab igas tulbas pikima sõna ning tagastab järjendi nendest sõnadest. Sama pikkusega sõnade puhul võetakse eespool olev sõna.

Kirjutada funktsioon *enim_taishaalikuid*, mis võtab argumendiks sõnade järjendi ning tagastab sõna, milles on kõige rohkem täishäälikuid. Täishäälikute leidmiseks võib kasutada järjendit:

```
taishaalikud = ["a","e","i","o","u","õ","ä","ö","ü"]
```

Kasutades neid funktsioone, leida kahemõõtmelise järjendi tulba pikimatest sõnadest sõna, milles on kõige rohkem täishäälikuid.

Näide tööst

```
sonade_tabel = [["piim", "kommikarp", "lillkapsas"], ["leib",
"hakkliha", "sink"], ["makaronid", "mandariinid", "tee"]]
```

Funktsioon *tulba_pikimad* tagastab sellise sõnade järjendi puhul järjendi

```
['makaronid', 'mandariinid', 'lillkapsas']
```

Funktsioon *enim_taishaalikuid* tagastab aga

```
mandariinid
```

Eksami arvutiosa teine harjutusülesanne

Mis väljastatakse ekraanile?

```
meeste_nimed = ["Aleksander", "Vladimir", "Sergei", "Andrei", "Aleksei", "Andres"]
naiste_nimed = ["Olga", "Jelena", "Tatjana", "Irina", "Svetlana", "Valentina"]
nimed = [meeste_nimed, naiste_nimed]
print(nimed[1][4])
```

Vali Andrei

Vali Svetlana

Vali Aleksei

Vali Irina

Õige vastus. See on tõesti nii.

Kahemõõtmelise järjendi kontrollküsimus

Mida teeb antud programm?

```
b = [1, 9, 1, 8]
korrutis = 1
for i in range(len(b)):
  korrutis *= i
print(korrutis)
```

Vali Leiab järjendi b kõigi elementide korrutise

Vali Korrutab kõik i väärtused kokku

Vali Veateade

Vale, listi elemendid korrutataks, kui oleks korrutis *= b[i].

Kahemõõtmelise järjendi kontrollküsimus

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kadi Rõmmel** (sünnikuupäev: 01.02.1995),

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **E-kursuse „Programmeerimise alused II“ kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs**,
   mille juhendaja on **Eno Tõnisson**,
   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace'i lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **08.05.2017**