TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Eva-Maria Pedosk

Programmeerimise algkursuste ülesannete kategoriseerimine Bloomi taksonoomia alusel

Bakalaureusetöö (9 EAP)

Juhendajad: Eno Tõnisson, PhD
Marina Lepp, PhD

Tartu 2019

Programmeerimise algkursuste ülesannete kategoriseerimine Bloomi taksonoomia alusel

Lühikokkuvõte:

Käesoleva bakalaureusetöö raames anti ülevaade Bloomi taksonoomia-alasest kirjandusest ja prooviti kategoriseerida statsionaarsete kursuste ning MOOCide "Programmeerimise alused" ja "Programmeerimise alused II" arvestustööde ülesandeid. Ülesannete kategoriseerimine viidi läbi Bloomi taksonoomia alusel ja kirja pandi kategoriseerimisel tekkinud raskused ning kitsaskohad. Lisaks kontrolliti kategoriseerimise abil, kas arvestustööde ülesannete sisu on vastavuses kursuste õpiväljunditega.

Võtmesõnad: Bloomi taksonoomia, programmeerimisülesannete kategoriseerimine, õpiväljundid

CERCS: P175 – Informaatika, süsteemiteooria; S270 – Pedagoogika ja didaktika

Categorization of tasks of introductory programming courses using Bloom's taxonomy

Abstract:

This Bachelor's thesis gave an overview of Bloom's taxonomy related literature and attempted to categorize the tasks of stationary courses and MOOCs "Introduction to Programming" and "Introduction to Programming II". The categorization of the tasks was carried out on the basis of Bloom's taxonomy. After categorizing the tasks, the difficulties of categorization were found and listed and it was analyzed, if the content of the tasks corresponded to the learning outcomes of the courses.

Keywords: Bloom's taxonomy, categorization of programming tasks, learning outcomes

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

Sisukord

Sissejuhatus

Programmeerimine hobina ja elukutsena on väga populaarne – programmeerimise õppimiseks on loodud erinevaid võimalusi (nt õpikud ja veebikursused [1]) ja igal aastal alustab suur hulk inimesi oma õpinguid just programmeerimist käsitlevatel erialadel. Põhiliseks programmeerimisoskuse omandamise ja arendamise viisiks on ülesannete lahendamine. Ülesannete lahendamine nõuab peamiselt probleemilahendamise oskuse rakendamist, mis on reaalteadustes oluline omadus, ja loovat lähenemist [2]. Selleks, et õppijad õpiksid programmeerima efektiivselt ja valdaksid vajalikku materjali, peaksid ülesanded kontrollima õpitavaid teadmisi võimalikult täpselt ja süstemaatiliselt. Kuna ülesandeid on võimalik koostada ja õppekavadesse sobitada väga erinevalt, saab abi otsida erinevatest kategoriseerimissüsteemidest, näiteks taksonoomiatest. Taksonoomia kõrval on kategoriseerimissüsteemideks veel näiteks ka ontoloogia (olemisõpetus) [3] ja tüpoloogia (liigitus tüüpide järgi) [4], kuid taksonoomia eelis seisneb eeskätt liigitamises mõõdetavate omaduste järgi [4].

Taksonoomia on abivahend süstemaatika loomiseks. Üheks mitmekülgseks hariduslikuks taksonoomiaks võib lugeda Bloomi taksonoomiat, mis hõlmab endas nii kognitiivse, afektiivse kui ka psühhomotoorse arengu ning võimekuse analüüsi. Bloomi taksonoomia jaotub kolmeks eelmainitud suuremaks kategooriaks ehk sfääriks, kus kognitiivne sfäär sisaldab omakorda alakategooriad [5]. Bloomi taksonoomia kõrval on õppija võimete ja oskuste hindamiseks levinud ka näiteks SOLO (ingl *Structure of the Observed Learning Outcome*) taksonoomia ja Finksi taksonoomia [6].

Antud bakalaureusetöö eesmärk on kirjanduse põhjal anda ülevaade Bloomi taksonoomiast ja proovida kategoriseerida Tartu Ülikoolis õpetatavate programmeerimise algkursuste arvestusülesandeid. Uurimistöö püüab välja selgitada, kui lihtne on antud ülesandeid kategoriseerida (sh toob autor välja kategoriseerimise raskused) ja hinnata antud ülesannete vastavust kursuste õpiväljunditele.

Töö esimeses osas viiakse lugeja esmalt kurssi taksonoomia üldmõistega, tutvustatakse Bloomi taksonoomiat ja antakse lühikene ülevaate informaatikaalastest teadustöödest Bloomi taksonoomia kohta, millele töö praktiline osa ka tugineb. Esimeses osas on ka alapeatükk kriitikast Bloomi taksonoomia kohta.

Töö teises osas kategoriseeritakse tuginedes Bloomi taksonoomiale programmeerimise algkursuste ehk statsionaarsete kursuste ja MOOCide (veebipõhiste kursuste [7])

“Programmeerimise alused” [7, 8] ning “Programmeerimise alused II” [9, 10] kasutatavaid kirjalikke ja arvutiülesandeid. Kategoriseerimise analüüsi põhjal leiab autor liigitamise raskused ja hindab ülesannete vastavust õpiväljunditele.

Käesolev bakalaureusetöö keskendub vaid modifitseeritud Bloomi taksonoomiale ja selle põhikategooriatele. Töö teises osas ehk ülesannete kategoriseerimisel lähtub autor vaid Bloomi taksonoomia kognitiivsest sfäärist. Põhjuseks on selle sfääri levinud rakendamine ja sobivus informaatikaalaste kursustega [2]. Samuti vastanduvad kognitiivse sfääri kategooriad õppeainete “Programmeerimise alused” ja “Programmeerimise alused II” õpiväljundite sisule paremini kui taksonoomia afektiivne või psühhomotoorne sfäär.

1. Taksonoomia

Antud peatükk tutvustab edasise töö paremaks mõistmiseks taksonoomia üldist mõistet ja antud bakalaureusetöö aluseks olevat Bloomi taksonoomiat. Lugejale tutvustatakse teemakohaseid teadustöid informaatikaõppes, seejuures ka kriitikat Bloomi taksonoomia kohta.

1.1. Taksonoomia mõiste

Taksonoomia on klassifitseerimissüsteem ja organiseerimise meetod [2, 11]. Taksonoomiaid on võimalik rakendada mitmekülgselt erinevates valdkondades - üks levinumaid valdkondi on näiteks bioloogia, kus taksonoomiaid rakendatakse elusorganismide liigitamisel [2].

Antud töö käsitleb hariduslikke ehk akadeemilisi taksonoomiaid, mida on võimalik rakendada informaatika valdkonnas. Hariduslikud taksonoomiad on raamistikud, mille eesmärgiks on ühtlustada õpetajate nägemust õppematerjalide omandamise võimalikest tasemetest [5]. Need aitavad detailsemalt kujundada arusaama õpiväljundite sisumõõtmetest [2, 6] ja organiseerivad mõtlemisoskusi nende keerukuse tasemel [11].

Hariduslikke taksonoomiaid kasutatakse eelkõige õppeaine struktuuri täiustamiseks [2]. Struktuuri täiustamise alla kuuluvad näiteks õppematerjalide ja kodutööde koostamine, õppeetappide kirjeldamine või juba eelnevalt mainitud õpiväljundite kujundamine [2, 11]. Lisaks saab hariduslikke taksonoomiaid rakendada ka haridusalastel uuringutel [2].

1.2. Bloomi taksonoomia

Bloomi taksonoomiat tutvustati esimest korda 1956. aastal Benjamin Bloomi ja tema mõttekaaslaste Max Engleharti, Edward Fursti, Walter Hilli ja David Krahtwohli poolt kui hariduslike õppe-eesmärkide kategoriseerimise raamistikku [12]. Taksonoomia jaotub tänase seisuga kaheks erinevaks versiooniks – originaalseks Bloomi taksonoomiaks ja modifitseeritud Bloomi taksonoomiaks (ingl *Revised Bloom's taxonomy*) [2, 5]. Antud bakalaureusetöö põhineb modifitseeritud (erinevates allikates nimetatud ka "moderniseeritud" ja "täpsustatud") Bloomi taksonoomiale.

Selleks, et kirjeldada õppimise tulemusi väljendavaid tegevusi, analüüsisid Bloom ja tema kaastöölised [5] õpetajate kasutatavaid ülesandeid, õppijate õppimist ja sotsiaalse arengu

hindamise vahendeid. Analüüsi tulemusena jõuti järelduseni, et õppimise tulemusi võib jaotada kolmeks suuremaks sfääriks: kognitiivne, afektiivne ja psühhomotoorne sfäär.

Iga sfäär sisaldab endas hierarhiliselt järjestatud kategooriaid, kus kehtib üldise ettekirjutusena kategooriate astmelise läbimise reegel – igasse järgmisesse kategooriasse edasi liikumine eeldab madalama kategooria sooritusoskusi.

1.2.1. Kognitiivne sfäär

Kognitiivne ehk tunnetuslik sfäär sisaldab endas kuut kategooriat, kus iga kategooria kirjeldab õppeprotsessi [6]. Antud sfäär viitab kognitiivsele arengule, mis tähendab inimese teadvuses kujunevate tunnetuslike, sõnaliste ja käitumuslike konstruktsioonide arendamist, diferentseerimist ja integreerimist [5].

Tänapäeval laialt kasutuses olev modifitseeritud Bloomi taksonoomia erinebki originaalsest taksonoomiast just käesoleva sfääri poolest [5]. Modifitseeritud taksonoomias on võrreldes originaalse taksonoomiaga kognitiivse sfääri kategooriad [5]:

1) väljendatud käskivas kõneviisis (tabel 1);
2) uut viisi järjestatud (tabel 2);
3) täiendatud alakategooriatega (tabel 3).

Modifitseeritud taksonoomia avaldati Lorin Andersoni ja David Krahtwohli poolt 2001. aastal [13] ning see andis võimaluse õppe-eesmärkide kahemõõtmeliseks sõnastamiseks [5]. Kahemõõtmelisus väljendub kognitiivse sfääri kirjeldamises maatriksina – vertikaalsel teljel kognitiivse sfääri põhikategooriad ja horisontaalsel teljel teadmiste alakategooriad (tabel 3), millest tuleb käesolevas alapeatükis ka juttu [2]. Järgnev tabel (tabel 1) toob aga välja originaalse [2] ja modifitseeritud [5] taksonoomia kognitiivsete kategooriate võrdluse.

Tabel 1. Originaalse [2] ja modifitseeritud [5] Bloomi taksonoomia põhikategooriate nimetuste võrdlus.

	Originaalne taksonoomia	**Modifitseeritud taksonoomia**
1	Teadmine (ingl *Knowledge*)	Pea meeles (ingl *Remember*)
2	Mõistmine (ingl *Comprehension*)	Saa aru (ingl *Understand*)
3	Rakendamine (ingl *Application*)	Rakenda (ingl *Apply*)
4	Analüüs (ingl *Analysis*)	Analüüsi (ingl *Analyze*)
5	Süntees (ingl *Synthesis*)	Hinda (ingl *Evaluate*)
6	Hindamine (ingl *Evaluation*)	Loo (ingl *Create*)

Kuna antud töö keskendub just modifitseeritud Bloomi taksonoomiale, seletab kognitiivse sfääri iga kategooria tähendust kõige paremini kognitiivsete protsesside mõõtmete tabel 13.3 Edgar Krulli raamatust “Pedagoogilise psühholoogia käsiraamat” (tabel 2).

Tabel 2. Kognitiivsete protsesside mõõtmed [5].

	Soorituse ehk kognitiivse protsessi põhikategooria	**Alakategooriad**
1	**Pea meeles** (olulise info reprodutseerimine pikaajalisest mälust)	1.1 Äratundmine 1.2 Meenutamine
2	**Saa aru** (õppeotstarbelise sõnumi tähenduse kindlakstegemine)	2.1 Interpreteerimine 2.2 Näitlikustamine 2.3 Klassifitseerimine 2.4 Summeerimine 2.5 Järeldamine 2.6 Võrdlemine 2.7 Seletamine
3	**Rakenda**	3.1 Sooritamine 3.2 Täideviimine
4	**Analüüsi** (materjali jaotamine koostisosadeks, seoste ja strukuuri äratundmine)	4.1 Eristamine 4.2 Organiseerimine 4.3 Omistamine
5	**Hinda** (otsuste langetamine kriteeriumide ja standardite põhjal)	5.1 Kontrollimine 5.2 Kritiseerimine
6	**Loo** (elementide kokkupanemine kooskõlalise terviku või originaalse produkti loomiseks)	6.1 Tekitamine (genereerimine) 6.2 Planeerimine 6.3 Produtseerimine

Omakorda täpsustavaks sisumõõtmeks ehk kategooriateks jaotub ka kognitiivse sfääri esimene tase “Teadmine” (ingl *Knowledge*) ehk “Pea meeles” (ingl *Remember*) [2]. Nende kategooriate seletused on illustreeritud Edgar Krulli eelmainitud teoses tabelis 13.2 “Moderniseeritud kognitiivse taksonoomia sisumõõde” (tabel 3).

Tabel 3. Modifitseeritud kognitiivse taksonoomia sisumõõde [5].

	Sisuelemendi põhikategooriad	**Alakategooriad**
A	**Faktiteadmine** (põhielementide teadmine)	A1. Terminoloogia teadmine A2. Spetsiifiliste detailide ja elementide teadmine
B	**Kontseptuaalne teadmine** (põhielementide vaheliste seoste teadmine)	B1. Klassifikatsioonide ja kategooriate teadmine B2. Printsiipide ja üldistuste teadmine B3. Teooriate, mudelite ja struktuuride teadmine
C	**Protseduuriline teadmine** (kuidas midagi teha)	C1. Ainespetsiifiliste oskuste ja tegevusmallide teadmine C2. Ainespetsiifiliste võtete ja meetodite teadmine C3. Protseduuri kasutamise sobivuse üle otsustamise kriteeriumide teadmine
D	**Metakognitiivne teadmine** (üldine teadmine tunnetustegevusest, sealhulgas teadlikkus oma tunnetustegevusest)	D1. Strateegiline teadmine D2. Tunnetuslike ülesannete teadmine, sealhulgas kontekstuaalne (olustikuline) ja tingimuslik teadmine D3. Teadmine iseenda kohta

Kognitiivse sfääri puhul saame rääkida eraldi ka õpiväljundite kategoriseerimiset. Nimelt on Biggs ja Tang oma raamatus "Õppimist väärtustav õpetamine ülikoolis" [14] esitanud Bloomi taksonoomia alusel (kognitiivse sfääri raames) laiendatud loetelu verbidest, mis aitavad täpsustada õpitegevusi, nimetades neid väljundiverbideks. Antud väljundiverbide (tabel 4) abil on võimalik kursuste õpiväljundeid lugedes indentifiseerida lauseehitusest kasutatavad verbid ning sobitada need Biggsi ja Tangi poolt kirja pandud verbidega, leides igale õpiväljundile oma kategooria.

Tabel 4. Modifitseeritud Bloomi taksonoomia väljundiverbid [14].

Meeldejätmine	Defineerida, kirjeldada, joonistada, leida, pealkirjastada, leida sobivad paarid, nimetada, tsiteerida, meelde tuletada, järele korrata, öelda, kirjutada.
Mõistmine	Klassifitseerida, võrrelda, näitlikustada, järeldada, demonstreerida, arutada, selgitada, ära tunda, illustreerida, tõlgendada, ümber sõnastada, prognoosida, teha kokkuvõte.
Rakendamine	Rakendada, muuta, valida, kalkuleerida, esitada draama vormis, ellu viia, ette valmistada, luua, esitada rollimänguna, välja valida, näidata, üle kanda, kasutada.
Analüüsimine	Analüüsida, iseloomustada, klassifitseerida, võrrelda, vastandada, vaidlustada, dekonstrueerida, tuletada, eristada, vahet teha, eritleda, uurida, organiseerida, anda ülevaade, seostada, lahutada tervikust, kasutada.
Hinnangu andmine	Anda hinnang, väita, hinnata, teha valik, järeldada, kritiseerida, otsustada, vaagida, langetada otsus, õigustada, progrnoosida, leida prioriteedid, tõestada, panna tähtsuse järjekorda, koostada andmestik, selekteerida, teostada järelvalvet.
Loomine	Ehitada, kavandada, arendada, tekitada, hüpoteesi välja töötada, avastada, välja mõelda, planeerida, toota, koostada, komponeerida, luua, teha, esitada.

Hariduslike taksonoomiate seast on just Bloomi taksonoomia kognitiivse sfääri mõju õppekavade koostamisele olnud märkimisväärne [2].

1.2.2. Afektiivne sfäär

Afektiivne ehk emotsioonidel põhinev sfäär kirjeldab õppijas vastuvõetava informatsiooniga emotsionaalse sideme loomist ja julgustamist informatsiooni hindama, organiseerima ning rakendama lähtudes õppija isikupärast [15].

Antud sfäär jaotub järgnevalt [15]:

1) **Vastuvõtmine** (ingl *Receiving*) – ideede, materjalide või nähtuste olemasolust teadlik olemine ja nendega arvestamine.

   Märksõnad: eristamine, aktsepteerimine, kuulamine, vastamine.

2) **Vastamine** (ingl *Responding*) – vähesel määral ideedele, materjalidele või nähtustele reageerimine/vastamine.

Märksõnad: järgimine, kiitmine, tunnustamine.

3) **Väärtustamine** (ingl *Valuing*) – ideede, materjalide või nähtuste tähtsustamine, hindamine.

   Märksõnad: loobumine, toetamine, debateerimine.

4) **Organiseerimine** (ingl *Organization*) – juba omandatud väärtuste seostamine (isiku) sisemise filosoofiaga.

   Märksõnad: arutlemine, sõnastamine, uurimine, tasakaalustamine, teoretiseerimine.

5) **Iseloomustamine** (ingl *Characterization*) – enesele seatud väärtuste järgimine.

   Märksõnad: (väärtuste) uuesti hindamine, vältimine, vastupanu, lahendamine.

Õpingutel panustatakse valdavalt õpetamise ja õppimise kognitiivsetele aspektidele, mistõttu jääb afektiivsus tavaliselt tähelepanuta [15]. Afektiivse sfääri puhul on tegu vähe uuritud, ebamäärase ja raskesti hinnatava sfääriga Bloomi taksonoomias [15].

1.2.3. Psühhomotoorne sfäär

Psühhomotoorne sfäär ei olnud osa originaalsest Bloomi taksonoomiast ja see lisandus taksonoomiasse 1970. aastatel doktor Elizabeth Simpsoni eestvedamisel [16]. Sfäär hõlmab endas teadud määral kognitiivseid ja afektiivseid protsesse, millele lisandub uue vaatenurgana motoorne aktiivsus ja tegevused [16]. Psühhomotoorsuse all täheldataksegi motoorse tegevuse läbiviimist [16].

Sfäär jaotub järgmisteks tasemeteks [16]:

1) **Tajumine** (ingl *Perception*) – meeleelunditega asjade, tunnuste ja seoste tundmine.
2) **Valmisolek** (ingl *Set*) – tegutsemis- ja kogemisvalmidus (eraldi mentaalne, füüsiline ja emotsionaalne valmidus).
3) **Juhitud vastus** (ingl *Guided response*) – tegutsemine ja reageerimine juhendaja jälgimise all.
4) **Mehhanism** (ingl *Mechanism*) – enesekindel tegutsemine/reageerimine ja väljakujunenud harjumuspärased vastused.
5) **Keeruline reageering** (ingl *Complex over response*) – oskuslik, sujuv ja efektiivne tegutsemine minimaalse aja- ja energiakuluga.

1.3. Bloomi taksonoomia informaatikas

Bloomi taksonoomia on üks enimkasutatavaid hariduslikke taksonoomiaid [17]. Oluliseks teadustööks Bloomi taksonoomiast informaatikas võib lugeda ITiCSE uurimisrühma tööd [2] informaatikaspetsiifilise taksonoomia arendamisest (ingl *"Developing a computer science-specific learning taxonomy*"). Töö arutleb olemasolevate taksonoomiate (nt SOLO, Niemierko, Tollingerova taksonoomiad) kasutamisest informaatikas, keskendudes enim Bloomi taksonoomiale, ja pakub välja uue taksonoomia, mis on kokku pandud spetsiaalselt informaatikaõppes kasutamiseks.

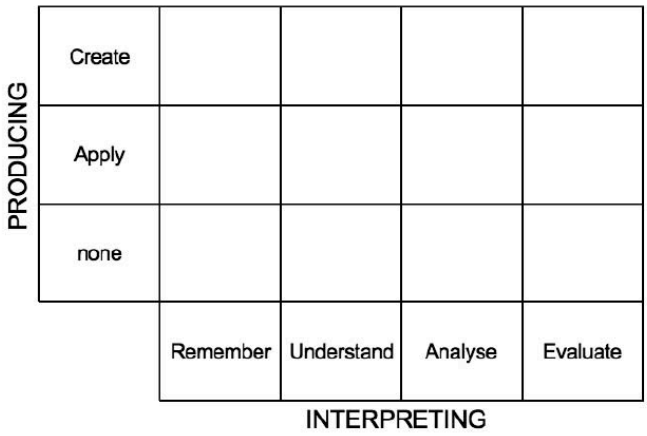
Antud töö põhjal on taksonoomiaid informaatikaõppes tavaliselt kasutatud kolmel viisil [2]:

- kursuse loomiseks või hindamiseks;
- õppestrateegia ja õppematerjalide koostamiseks;
- õppijate vastuste analüüsimiseks ja arengu hindamiseks.

Selle teadustöö suurimaks väärtuseks antud bakalaureusetööle võib lugeda peatükki informaatika eripärast (*"5. What is so specific about computer science?"*). Peatükist leiab aspektid ja omadused, mida olemasolevad taksonoomiad antud töögrupi arvates informaatikavallas hetkel ei käsitle. Välja on toodud näiteks informaatika keerukus. Informaatika põhiolemuseks on efektiivse ja tõhusa lahenduse väljatöötamine ning sellise lahenduse saavutamiseks on oluline liigendada probleemid väiksemateks alaprobleemideks ja mooduliteks. Kokkuvõtvate märksõnadega toodi välja loomupärased omadused, mis peaksid ühel informaatikaala professionaalil olema. Nendeks on probleemilahendamise oskus, domeeni modelleerimisoskus, teadmiste kirjeldamise oskus, efektiivsus, abstraktsus, loovus, kategoriseerimisoskus, suhtlemise oskus ja tarkvaraarenduse heade tavade tundmine. Just neile eelmainitud omadustele peaks uurimisgrupi arvates saama õppetöö raames hinnangut anda (ehk liigitada omadused ja/või oskused kategooriatesse sarnaselt Bloomi taksonoomia sisule). Töögrupi arvates on informaatikaõppes eesmärgiks alati intuitiivselt elegantse koodi kirjutamine, mitte vaid õppetöö käigus.

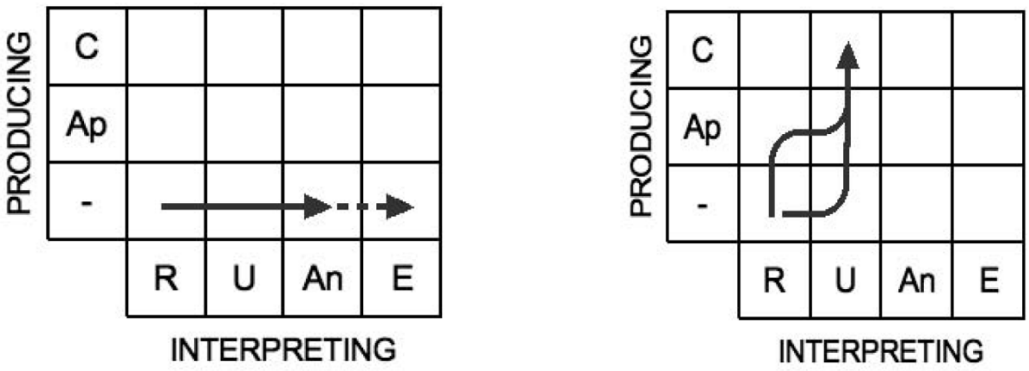
Antud teadustöö on käesolevast bakalaureusetööst lähtudes oluline just seetõttu, et läbi uue taksonoomia ("*The Matrix Taxonomy*") koostamise ja välja pakkumise annab ITiCSE uurimisrühm selged direktiivid, kuidas oleks õigem informaatika õppijate võimekust hinnata ja millised omadused on olulised. Uus taksonoomia on üles ehitatud modifitseeritud Bloomi taksonoomia kognitiivse sfääri alusel ja tegu on praktilise raamistikuga, mis aitab

hinnata õppija võimekust informaatikas (k.a tehnikateaduses) täpsemini, kui seda on seni võimaldanud muud hariduslikud taksonoomiad. Taksonoomia koostamisel on lähtutud käsitlusest, et koodist arusaamine ja koodi kirjutamine (interpretatsioon) on eraldiseisvad võimekused. Need kaks suunda on esitatud kahemõõtmelise maatriksina – loomise tasemed (kategooriad "Rakenda", "Loo") maatriksi vertikaalsel teljel ja interpretatsiooni tasemed (kategooriad "Pea meeles", "Saa aru", "Analüüsi", "Hinda") horisontaalsel (joonis 1). Antud maatriks võimaldab juhendajal või õppejõul võimekust hinnata mugavalt, tehes õppija poolt omandatud tasemetesse (lahtritesse) märke.

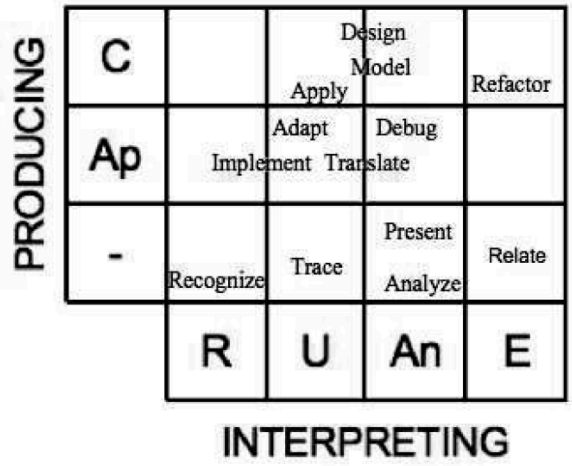


Joonis 1. ITiCSE töögrupi moodustatud kahemõõtmeline maatriks [2].

Antud kahemõõtmelise maatriksi peal on ITiCSE töögrupi teadustöös illustreeritud ka näiteks mitmed erinevad "teekonnad" (joonis 2), mis näitavad õppijate võimalikke kursusel saavutatud oskusi. Teekondade koostamiseks on loodud ka abimaatriks (joonis 3), millele on teekondade märkimisel paremini orienteerumiseks kirja pandud programmeerimisalased tegevused.



Joonis 2. Vaid teoreetilise (vasak) ja praktilise (parem) kompetentsusega õppijate teekonnad [2].



Joonis 3. Programmeerimisalased tegevused kahemõõtmelises maatriksis [2].

Küll aga on tegu kohati pooliku lahendusega, sest välja pakutud taksonoomia käsitleb vaid õppija abstraktseid oskuseid. Käesolevas bakalaureusetöös seda maatriksit ülesannete kategoriseerimisel ja õpiväljundite vastavuse kontrollis ei kasutata.

Aktuaalseks teadusartikliks on ka "*Bloom's Taxonomy for CS Assessment*" [18], mis käsitleb samuti modifitseeritud Bloomi taksonoomia kasutamist informaatikaalase pädevuse hindamiseks. Selles töös keskendub uurimisrühm aga konkreetselt programmeerimisülesannete kategoriseerimisele. Teadustöös on modifitseeritud Bloomi taksonoomia kognitiivse sfääri iga kategooria kohta alapeatükk ja neis peatükkides ühe kuni kahe näiteülesande kategoriseerimine. Näiteliigitamised viiakse läbi ülesannetel, mis on kirjutatud programmeerimiskeeles Java, kuid sarnast kategoriseerimist on võimalik eeskujuks võtta ka antud bakalaureusetöös käsitletud kursuste "Programmeerimise alused" ja "Programmeerimise alused II" Pythonis kirjutatud ülesannete kategoriseerimiseks.

Antud teadustöö omistab igale Bloomi taksonoomia kategooriale teatud kriteeriumid/omadused, mis kergendavad Bloomi taksonoomia kasutamist informaatikas ja ülesannete liigitamisel (tabel 5).

Tabel 5. Autori kokkuvõte teadustöö "*Bloom's Taxonomy for CS Assessment*" kategooriate kirjeldustest.

Pea meeles	Koodi põhjal teema või kontseptsiooni implementeerimise või kirjelduse äratundmine, kursusel õpitud materjali meeldetuletamine (kus ülesanne ja materjal omavad sama konteksti)
Saa aru	Algoritmi/kontseptsiooni tõlgendamine, esitlemine, kirjeldamine, nimetamine, defineerimine
Rakenda	Kas tuttavale (kuid võõra konteksti, andmete jms) probleemile või täiesti tundmatule probleemile algoritm/kontseptsioon rakendamine
Analüüsi	Ülesande osadeks jaotamine, organiseerimine, tähtsate ja vähemtähtsate osade identifitseerimine, koodis seoste leidmine
Hinda	Nõuetele vastavuse kontrollimine, koodi testimine ja selle hindamine
Loo	Uute lahenduste loomine kasutades nt kas õpitud algoritme või algoritmide kombinatsioone, mis on õppijatele uus lahendus

Näiteks liigitatakse avaldise 2 + 4 / 7 * 5 % 3 == 7 lahendamise ülesanne kategooriasse "Rakenda". Kui tegu oleks lihtsalt avaldisega 1 + 2, piisaks õpilasel vaid õpitud teadmiste meenutamisest (kategooria "Pea meeles"), kuid kuna avaldise lahendamiseks on oluline tehete järjekord, peab vastuse saamiseks tehtele rakendama algoritmi.

Veel ühe näitena esitatakse õppijatele programmeerimiskeeles Java kirjutatud klass, kus on deklareeritud muutujad *numbers* ja *used*:

```
private double numbers[] = new double[10];

private int used = 0;
```

Selles klassis on loodud arvude miinimumi arvutamiseks järgneva sisuga meetod:

```
for (int i = 0; i < used; i++) {
      min = Math.min(min, numbers[i]);}
```

Ülesandes esitatakse väide, et järgnev meetod on parem alternatiiv eelmainitud sisuga meetodile.

```
public double min() {
   double min = numbers[0];
   for (double number : numbers) {
         min = Math.min(min, number);
   }
   return min;
}
```

Ülesandeks on leida nende kahe lahenduse erinevused ja valida sobivam lahendus. Kuna ülesanne nõuab lahendajalt kahe tsükli kasutamise hindamist ja analüüsimist (kuid mitte koodi osadeks jaotamise ja organiseerimise ehk kategooria "Analüüsi" mõttes), kuulub ülesanne Bloomi taksonoomia "Hinda" kategooriasse. Antud teadustöö toob läbi ülesannete kategoriseerimise tähelepanu kursuse õpetamisele ja uuringu tulemustes tõdeti, et kursuse koostajate ühine arusaam Bloomi taksonoomiast on väärtuslik vahend eksamiküsimuste koostamiseks ning seda eriti õppeainetes, kus õppejõude on rohkem kui üks.

Sarnasel teemal kodumaiseid uurimistöid kirjutatud ei ole, kuid siinkohal saab välja tuua kolm Tartu Ülikooli bakalaureusetööd, mis on kaudselt seostatavad antud bakalaureusetöö teemaga. Üheks nendest töödest on Kadi Rõmmeli 2017. aasta bakalaureusetöö [19] "E-kursuse „Programmeerimise alused II" kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs". Kuna Rõmmeli töö käsitleb samuti arvestusülesandeid, võimaldaks Rõmmeli töö tulevikus võrrelda õppijate õppeedukust ülesannetega, mida on täiendatud ja analüüsitud lähtuvalt Bloomi taksonoomiast. Selleks oleks aga vaja käesoleva bakalaureusetöö edasi uurimist – uute näiteülesannete koostamist ja neid ülesandeid lahendanud õppijate lahenduste uurimist. Sarnasel teemal kirjutab ka Kirsti Tagam oma bakalaureusetöös [20] "E-kursuse "Programmeermise alused II" rekursiooni temaatika küsimuste ja ülesannete loomine". Küll aga on nii Rõmmeli kui ka Tagami tööd piiratud vaid järjendite ja rekursiooni teemadega.

Mainimist vajab ka Helen Hendriksoni 2018. aasta bakalaureusetöö [21] "MOOCi „Programmeerimise alused" ülesannete lahenduste analüüs". Kuna antud töö uurib õppijate

lahenduste ülesannete kitsaskohti ja annab asjaliku ülevaate õppijate vigadest ülesannete lahendamisel, leiab Hendriksoni töö üles kursuse teemad, mis vajavad rohkem tähelepanu. Koos Hendriksoni töö tulemuste ja Bloomi taksonoomia rakendamisega oleks võimalik veenduda, ja vajadusel parandada, kursuse ülesannete mitmekülgsuses. Samuti on aga ka Hendriksoni bakalaureusetöö piiranguks kitsendatus vaid tingimuslausete ja tsükli teemade käsitlemisele.

Käesolev bakalaureusetöö on eelmainitud teadustöödest eelkõige erinev põhjusel, et kategoriseerimine viiakse läbi programmeerimiskeeles Python kirjutatud ülesannetel (võrdluses välismaiste teadustöödega) ja töö on Tartu Ülikooli statsionaarsete kursuste ning MOOCide "Programmeerimise alused" ja "Programmeerimise alused II" spetsiifiline.

1.4. Bloomi taksonoomia kriitika

Bloomi taksonoomiat on kritiseeritud mitmest vaatenurgast. Taksonoomia nõrkusena on välja toodud näiteks kategooriate kattuvus, mis raskendab antud taksonoomia rakendamist vajalikule materjalile ja vähene tähelepanu kriitilise mõtlemisele olulisusele [2]. Teadlaste poolt on täheldatud ka liigset keskendumist kognitiivsusele, millega kaasneb afektiivsuse ja psühhomotoorsuse fookusest kõrvale jäämine [22]. Sellega viidatakse, et taksonoomia on kasulik enim vaid keelelisele ning matemaatikaalasele arengule [22]. Läbi on viidud ka uuring [5], mille tulemused ei toeta taksonoomia astmelise läbimise põhimõtet väites, et kõrgema astme protsessid võivad madalama astmete protsessidele ja oskuste omandamisele hoopis kaasa aidata. See aga ei tähenda, et astmelisest sfääride läbimisest väär lähtuda oleks.

Informaatikaõppe seisukohast on välja toodud, et Bloomi taksonoomia ei ole kognitiivsete sfääri poolest vastavuses programmeerimise õppimisega alustanud õppijate õpingute suunaga [2]. Seda just protsesside järjestuse suhtes – nimelt on õppijad, kelle sooritus kognitiivse sfääri esimestes kategooriates saadab ebaedu, võimelised sfääri kõrgematel tasemetel olema väga edukad [23]. Samuti võib esineda ka olukord, et ülesanne, mis esitab väljakutse algaja programmeerijale analüüsi- ja sünteesimisoskuse rakendamise läbi, on klassikaline teadmiste rakendamine kogenud programmeerijale ehk taksonoomiat ei ole võimalik üheselt rakendada [2]. Seda toetab ka üks psühholoogia põhimõtteid, et kõige keerulisemad psüühilised protsessid on õpitavad tasemel, kus neid sooritatakse minimaalse või ilma vaimse pingutuseta [5].

Samuti on ITiCSE töögrupi poolt kirjutatud [2], et taksonoomia kasutamist informaatikaõppes loetakse kontekstipõhiseks. Õppija, kes lahendab näiteülesandele väga

sarnast programmeerimisülesannet (näiteks kontrolltööd või eksamit tehes), demonstreerib Bloomi taksonoomia sfääride madalamaid protsesse kui õpilane, kes lahendab struktuurilt tundmatut ülesannet. Uuringute tulemusel on ka märgitud, et programmeerimiskursuste õppe-eesmärkide ja ülesannete hindamisel peetakse arendamisväärseks eelkõige teadmiste rakendamise protsessi. Lisaks hõlmab uuringute kohaselt rakendamise protsess endas ka antud sfääri kõrgemaid protsesse.

2. Ülesannete kategoriseerimine

Käesolevas bakalaureusetöös kategoriseeritakse ülesandeid, mis on konkreetselt programeerimise algkursuste jaoks koostatud ja mille andsid autorile töö juhendajad. Kategoriseeritavaid ülesandeid, mida antud bakalaureusetöö käsitleb, on kahte tüüpi:

1) kirjalikud ülesanded (MOOCi kursustel nimetusega "paberosa", statsionaarsetel kursustel "loengu kontrolltöö");
2) arvutiülesanded (MOOCi kursustel nimetusega "arvutiosa", statsionaarsetel kursustel "arvutikontrolltöö").

Kuigi ülesanded jaotuvad kaheks osaks, loetakse ülesandeid siiski terviklikuks tööks ja töö sooritatakse kursuste lõpus. Ülesannete kategoriseerimisel on eelduseks, et õppijatel puudub eelnev kokkupuude programmeerimisega ja vajalikud teadmised õpitakse kursustelt "Programmeerimise alused" ja "Programmeerimise alused II". Kategoriseerimisel toetub autor täpsemini nii modifitseeritud Bloomi taksonoomiale kui ka 1.2 peatüki teadustööle "*Bloom's Taxonomy for CS Assessment*". Kirjalikkude ja arvutiülesannete kategoriseerimise järel on autor tulemusi illustreerinud kokkuvõtvate tabelitega, kus kategooriatesse sobivuse hindamiseks on kasutusel 3-palli skaala (1 – antud kategooria omadusi ei leidunud, 2 – leidus mõningaid antud kategooria omadusi ja 3 – kõige rohkem antud kategooria omadusi).

2.1. Õpiväljundid

Antud bakalaureusetöös on kategoriseeritavad ülesanded Tartu Ülikooli kursustelt "Programmeerimise alused" (statsionaarne), "Programmeerimise alused II" (statsionaarne), "MOOC Programmeerimise alused" ja "MOOC Programmeerimise alused II". Nii MOOCid kui ka statsionaarsed kursused on oma ülesehituselt ja materjalide poolest identsed. Erinevalt statsionaarsetest kursustest on MOOCi kursused aga veebipõhised ja neil puuduvad iganädalased loengud ja praktikumid [7, 9].

Kuigi mõlema õppeaine eesmärgid on pisut varieeruvad ("Programmeerimise alused" on eelduseks ainele "Programeerimise alused II" [10]), on üldine eesmärk üks – anda õppijatele baasteadmised programmeerimisest keeles Python. Kursuse "Programmeerimise alused" sihtgrupiks on kas vähese või olematu programmeerimise kokkupuutega inimesed [7] ja "Programmeerimise alused II" sihtgrupiks õppijad, kellel on kokkupuude programmeerimisega vähemalt eeldusaine "Programmeerimise alused" raames [9].

“Programmeerimise alused” statsionaarne kursus [8] ja MOOC [7] käsitlevad teemasid nagu andmetüübid, muutujad, algoritmid, tingimuslaused, tsüklid, järjendid, funktsioonid ja andmevahetus. Tartu Ülikooli õppeinfosüsteemis [24] on märgitud, et “Programmeerimise alused” statsionaarse kursuse läbinu on motiveeritud kasutama arvutit ja koostama programme oma edasiste õpingute vältel, oskab programmeerimise baaskonstruktsioone esitada nii plokkskeemidena kui ka programmilõikudena ja oskab tekstina püstitatud ülesande realiseerida arvutiprogrammina. MOOCi kursuse puhul on õpiväljundid sõnastatud aga mõnevõrra teisiti. “MOOC Programmeerimise alused” kursuse läbinu oskab keeles Python programmeerimise baaskonstruktsioone, ülesannete realiseerimist programmina (kasutades erinevaid andmetüüpe, operatsioone jms) ja programmeerimisülesanete leidmist, sõnastamist ning lahendamist. Ta omab esmast ülevaadet programmeerimise ajaloost ja valdkonnast ning on teadlik oma võimalustest programmeerimisalaste õpingute jätkamisest.

“Programmeerimise alused II” statsionaarne kursus [10] ja MOOC [9] tegelevad eeldusaine materjali kordamisega ja täiendamisega. Uuteks läbivateks teemadeks on kahemõõtmeline järjend, kahekordne tsükkel, uued andmestruktuurid (ennik, hulk, sõnastik), viitamine ja muteerimine, testimine ja rekursioon. Erinevalt “Programmeerimise alused” ainest, on antud õppeaine statsionaarse kursuse ja MOOCi õpiväljundid samad. Need loetlevad üles põhiliste programmeerimiskonstruktsioonide (omistuslause, tingimuslause, rekursioon...) kasutusoskuse, andmetüüpide ja -struktuuride tundmise ning vastavate standardoperatsioonide kasutusoskuse, oskuse programmi analüüsida, selgitada ja laiendada, oskuse luua algoritmi ning oskuse koostada, vormistada, siluda ja testida lahendusporgrammi.

Selleks, et töö hilisemas osas oleks võimalik kontrollida õpiväljundite vastavust arvestustööde sisule, tasub lisaks ülesannetele kategoriseerida Bloomi taksonoomia alusel ka kursuste õpiväljundid. Seda saab teha alapeatükis 1.2.1 oleva tabeli 4 abil, kus on igale taksonoomia kategooriale välja toodud väljundiverbid. Väljundiverbide tabeli järgi oleks võimalik kursuste õpiväljundid kategoriseerida näiteks nii, nagu on välja toodud allpool, tabelites 6 ja 7. Õpiväljundite kategoriseerimisel langevad mõned õpiväljundid tabel 4 verbide poolest mitmesse kategooriasse ning on seega ka nii kategoriseeritud tabelites 6 ja 7. Juhul, kui väljundiverbi tabelis puudus täpselt sama sõna, mida oli kasutatud kursuste õpiväljundite kirjeldamises, on leitud sellele sõnale väljundiverbide tabelist lähedaseim ning lisatud see õpiväljundi järel sulgudesse. Kuna “Programmeerimise alused II” õpiväljundid

on nii statsionaarsel kursusel kui ka MOOCil samad, on antud õpiväljundid kategoriseeritud tabelis 7 samasse tulpa ilma neil õpiväljunditel erinevust tegemata. "Programmeerimise alused" statsionaarse kursuse ja MOOCi õpiväljundites oli aga mõningaid erinevusi ja sellest tulenevalt on "Programmeerimise alused" statsionaarse kursuse õpiväljundid kategoriseeritud eraldi tulpadesse (tabel 6).

Tabel 6. "Programmeerimise alused" õpiväljundite kategoriseerimine.

Kategooriad	"Programmeerimise alused" õpiväljundid (MOOC)	"Programmeerimise alused" õpiväljundid (statsionaarne)
Pea meeles	õpiväljund puudub	õpiväljund puudub
Saa aru	- programmeerimise ajaloo ülevaate **omamine** ("demonstreerida")	õpiväljund puudub
Rakenda	- oskus **kasutada** erinevaid andmetüüpe, operatsioone jms	- oskab programmeerimise baaskonstruktsioone **esitada** nii plokkskeemidena kui ka programmilõikudena ("näidata") - **motiveeritud kasutama** arvutit
Analüüsi	- oskus **kasutada** erinevaid andmetüüpe, operatsioone jms	õpiväljund puudub
Hinda	- programmeerimisalaste õpingute jätkamise võimalustest **teadlik olemine** ("langetada otsus")	õpiväljund puudub
Loo	- oskus ülesandeid programmina **realiseerida** ("arendada") - oskus **leida ja sõnastada** programmeerimisülesandeid ("koostada") - oskus programmeerimisülesandeid **lahendada** ("arendada")	- oskab tekstina püstitatud ülesande **realiseerida** arvutiprogrammina ("koostada") - **motiveeritud koostama** programme ("välja mõelda")

Tabel 7. “Programmeerimise alused II” õpiväljundite kategoriseerimine.

Kategooriad	“Programmeerimise alused II” õpiväljundid (MOOC ja statsionaarne)
Pea meeles	- oskus **vormistada** lahendusprogramme (“kirjutada”)
Saa aru	- oskus **ära tunda** andmetüüpe ja –struktuure - oskus programme **selgitada**
Rakenda	- oskus **kasutada** programmeerimiskonstruktsioone - oskus standardoperatsioone **kasutada** - oskus **laiendada** programmi (“ellu viia”) - oskus **luua** algoritme
Analüüsi	- oskus **kasutada** programmeerimiskonstruktsioone - oskus standardoperatsioone **kasutada** - oskus programme **analüüsida** - oskus lahendusprogramme **siluda** (“dekonstrueerida”)
Hinda	- oskus lahendusprogramme **testida** (“hinnata”)
Loo	- oskus **luua** algoritme - oskus **koostada** lahendusprogramme

Autori koostatud tabelid 6 ja 7 on ülesannete kategoriseerimise järgselt aluseks kursuste õpiväljundite vastavuse kontrollile.

2.2. Kirjalikud ülesanded

Loengu kontrolltöö ja paberosa (arvestustöö kirjalikud osad) lahendatakse paberil. Õppijad on kohustatud kursuse edukaks läbimiseks lahendama ülesanded ilma arvuti, materjalide ja muu kõrvalise abita ning ületama etteantud punktilävendi.

2.2.1. “Programmeerimise alused”

Autor kategoriseerib kahe 2019. aasta arvestustöö variandi ülesandeid – MOOCi paberosa variant A (Lisa 1) ja statsionaarse kursuse loengu kontrolltöö variant 16 (Lisa 2). Mõlemad tööd koosnevad kolmest ülesandest. Esimesena kategoriseerib autor paberosa variant A ülesandeid.

Kõik kolm paberosa ülesannet on sisult natukene erinevad, kuid küsivad sama asja – etteantud programmi väljundit. Esimese ülesande keskmeks on kasutajalt sisendi küsimine ja tingimuslausete läbimine. Teine ülesanne seisneb *for*-tsükliga järjendi läbimises, milles sisaldub jäägiga jagamise tingimuslause ja kolmandas ülesandes on defineeritud meetod `poolitaja` ning kood sisuks on antud meetodi välja kutsumine *while*-tsüklis.

Ülesannete kontekstid ja koodid, mis õppijale ette antakse, ei ole kursuse materjalides sellisel viisil kasutatud, nagu antud paberosas, seega ei ole tegu lahendajale tuttavate ülesannetega. Ülesannete puhul tasub õppijal läheneda lahenduse leidmisele samm-sammult, rakendades ülesannetele õpitud kontseptsioone ja asendades muutujate kohale välja pakutud sisendeid (sobivus "Rakenda" kategooriaga). Tingimuslausete korral on asjakohane silmas pidada ka näiteks loogiliste tehete või avalduste õigesti lahendamist/järeldamist. Kategooriate omaduste poolest võiks ülesanded sobida kategooriasse "Saa aru", sest programmi väljundi leidmisega võiksid käia kokku Bloomi taksonoomia märksõnad nagu *summeerimine* ja *järeldamine*, kuid rõhk on siiski õpitu rakendamisel. Lisaks saab kategooria määramisel kasutada ka välistamise tehnikat. Kindel on see, et kuna paberosa ülesannetes koodimist ei toimu, on võimalik kõrvale jätta Bloomi taksonoomia "Loo" kategooria esinemise. Autor leiab kõigi kolme paberosa ülesande puhul, et kõige enam sobiksid need omaduste poolest kategooriasse "Rakenda" (tabel 8).

Tabel 8. MOOCi “Programmeerimise alused” paberosa A ülesannete kategooriad.

Kategooriad	Ülesanne 1, 2 ja 3
Pea meeles	1 – antud kategooria omadusi ei leidunud
Saa aru	2 – leidus mõningaid antud kategooria omadusi
Rakenda	**3 – kõige rohkem antud kategooria omadusi**
Analüüsi	1 – antud kategooria omadusi ei leidunud
Hinda	1 – antud kategooria omadusi ei leidunud
Loo	1 – antud kategooria omadusi ei leidunud

Loengukontrolltöö variant 16 (Lisa 2) puhul on tegu natukene vahelduvamate ülesannetega. Esimene ülesanne kirjeldab kujutletava programmi tööd ja lahendajal on vaja antud kirjelduse põhjal koostada tsükliline plokkskeem. Proovides antud ülesannet liigitada, osutus see kategooriate omaduste esinemiste poolest väga heaks ülesandeks. Pealtnäha on ülesanne lihtne, kuid plokkskeemini jõudmiseks rakendatakse õppija mitut oskust korraga. Näiteks saab õppija alustada ülesande osadeks jaotamisest, tehes esmalt kindlaks, millised tingimuslaused peaksid esinema, et jõuda õige lahenduseni (sobivus “Analüüsi” kategooriaga). Seejärel tuleb õppijal meenutada õpitud teadmisi tingimuslausete kirjutamisest ja plokkskeemi geomeetriast (sobivus “Pea meeles” kategooriaga). Tähtis roll on antud ülesande puhul ka kriteeriumitele vastavuse kontrollil, sest ülesande nõuded on pandud kirja ühtse tekstina, mitte punkthaaval (sobivus “Hinda” kategooriaga). Kiiresti lugedes võivad mõned kriteeriumid seetõttu jääda tähelepanuta ja ülesande lahendus jääks ebatäielikuks. Olenemata asjaolust, et ülesandes esineb mitme erineva kategooria omadusi, leiab autor, et kõige enam võiks ülesanne sobida kategooriasse “Saa aru”, sest praktiliselt seisneb ülesanne programmi töö kirjeldamises (tabel 9).

Ülesanded 2 ja 3 küsivad sarnaselt MOOCi pabersosa ülesannetele programmi väljundit. Küll aga on need MOOCi ülesannetest pisut erinevad, sest ei küsita ainult programmi väljundit, vaid oodatakse pikemat arutelu. Sellest tulenevalt võib siinkohal “Rakenda” kategooria alakategooriatest (*sooritamine, täideviimine*) kaalukamaks lugeda “Saa aru” kategooriasse kuuluvust. Erinevalt MOOCi kirjalikest ülesannetest lisandub kategoriseerimisel lisaks märksõnadele *summeerimine* ja *järeldamine* ka klappivus

märksõnaga *seletamine*. Sellest tulenevalt kategoriseeriks autor ülesanded 2 ja 3 kategooriasse "Saa aru" (tabel 9).

Tabel 9. "Programmeerimise alused" statsionaarse kursuse loengu kontrolltöö variant 16 ülesannete kategooriad.

Kategooriad	Ülesanne 1	Ülesanne 2 ja 3
Pea meeles	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Saa aru	**3 – kõige rohkem antud kategooria omadusi**	**3 – kõige rohkem antud kategooria omadusi**
Rakenda	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi
Analüüsi	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Hinda	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Loo	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud

2.2.2. "Programmeerimise alused II"

Autor kategoriseerib ülesandeid MOOCi kursuse paberosa 2019. aasta variandist (Lisa 3) ja statsionaarse kursuse 2018. aasta tööst (Lisa 4). Mõlemad tööd koosnevad viiest ülesandest.

MOOCi paberosa esimeses ülesandes on lahendajal vaja koostada funktsioon `paaritus_reas`, mis töötaks ülesandes esitatud koodiga võrdväärselt. Sellest tulenevalt on antud funktsiooni koostades tegu olemasolevale lahendusele alternatiivi loomisega ja ülesanne võimaldab õppijatel loovalt läheneda. Ülesandes esineb Bloomi taksonoomia "Pea meeles", "Analüüsi" ja "Hinda" kategooria omadusi. Õppijal on tarvis presenteeritud koodist ära tunda vajalikud komponendi (sobivus "Analüüsi" kategooriaga), mis ka funktsioonis `paaritus_reas` kindlasti esinema peavad, ja seejärel meenutada nende süntaksit (sobivus "Pea meeles" kategooriaga). Lisaks on vaja kontrollida kas kirjutatud funktsioon on võrdväärne etteantud koodiga ehk nõuetele vastav (sobivus "Hinda" kategooriaga). Koodi analüüsimist ja meenutamist võib siinkohal lugeda aga väiksemaks osaks suuremast tervikust, milleks on uue koodi kirjutamine ja millest tulenevalt liigitub ülesanne kategooriasse "Loo" (tabel 10).

Ülejäänud nelja ülesande küsimuseks on “Mis väljastatakse programmi töötamisel ekraanile?”. Need ülesanded käsitlevad järjest andmetüüpe nagu järjend, sõnastik, hulk ja rekursioon. Arutlust antud ülesannetes ei oodata, seega ei pea õppija midagi kirjeldama ega seletama. Õppijal on vaja meenutada, kuidas antud andmetüüpidega ümber käiakse ja rakendada vastavaid operatsioone/protseduure, mistõttu kategoriseeriks autor ülesanded “Rakenda” kategooriasse (tabel 10). Kattuvus praktikumimaterjalidega oli vähene (piiratud arv näiteid ja näidetega erinev kontekst).

Tabel 10. MOOCi “Programmeerimise alused II” paberosa ülesannete kategooriad.

Kategooriad	Ülesanne 1	Ülesanne 2, 3, 4 ja 5
Pea meeles	2 – leidus mõningaid antud kategooria omadusi	2 – leidus mõningaid antud kategooria omadusi
Saa aru	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Rakenda	1 – antud kategooria omadusi ei leidunud	**3 – kõige rohkem antud kategooria omadusi**
Analüüsi	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Hinda	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Loo	**3 – kõige rohkem antud kategooria omadusi**	1 – antud kategooria omadusi ei leidunud

Statsionaarse kursuse ülesanded (Lisa 4) erinevad MOOCi ülesannetest selle poolest, et vastuseid on vaja põhjendada (sobivus "Saa aru" kategooriaga), lihtsalt programmi väljundi kirjutamisest ei piisa. Esimene ülesanne on sarnaselt MOOCi ülesandele etteantud koodiga võrdväärselt töötava funktsiooni kirjutamine. Ka selles ülesandes esineb Bloomi taksonoomia "Pea meeles", "Analüüsi" ja "Hinda" kategooria omadusi, kuid kuna ülesandeks on taas uue funktsiooni loomine, kus on õpilasel võimalik ülesanne lahendada lähtudes enda nägemusest ja leidub erinevaid võimalikke lahendusi, liigitab autor sarnaselt MOOCi arvestustöö esimesele ülesandele see "Loo" kategooriasse. Lisaks leidub ülesandes töö juhistest lähtuvalt "Saa aru" kategooria omadus, sest õppijatel on palutud oma vastuseid põhjendada, küll aga on põhifookus ülesandel siiski funktsiooni kirjutamisel (tabel 11).

Teine, kolmas ja neljas ülesanne küsivad kõik etteantud koodide väljundit. Teine ülesanne käsitleb kahemõõtlemise järjendi teemat, kolmas sõnastikke ja neljas hulkade teemat. Antud ülesannete puhul õppija midagi uut looma ei pea, vaid olemasolevat programmi õigesti tõlgendama (sobivus "Saa aru" kategooriaga). Antud ülesannete puhul on aga taas rõhutatud ka vastuste põhjendamist, mis suurendab autori veendumust, et ülesanded võiksid kuuluda kategooriasse "Saa aru" (tabel 11).

Viies ülesanne on rekursiooniteemaline. Antud ülesanne on praktikumimaterjalides pea üks ühele olemas. Sama on näiteks funktsiooni nimi, tingimuslause `if x < 0` ja osaliselt *return* laused. Muudetud on neis vaid funktsioonide argumente ja *print* lauset. Asjaolu tõttu, et materjal ja ülesanne on sarnased, oleks võimalik ülesannet liigitada mälupõhiseks (sobivus "Pea meeles" kategooriaga). Samuti oleks rekursiooni puhul õige rääkida ka algoritmi rakendamisest (sobivus "Rakenda" kategooriaga). Lähtudes töö juhistest (vastuste põhjendamine) ja ülesande küsimusest "Mida trükib ekraanile järgmine programmilõik?" liigitab autor ülesande kategooriasse "Saa aru" (tabel 11).

Tabel 11. “Programmeerimise alused II” statsionaarse kursuse kirjalike ülesannete kategooriad.

Kategooriad	Ülesanne 1	Ülesanne 2, 3 ja 4	Ülesanne 5
Pea meeles	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi
Saa aru	2 – leidus mõningaid antud kategooria omadusi	**3 – kõige rohkem antud kategooria omadusi**	**3 – kõige rohkem antud kategooria omadusi**
Rakenda	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi	2 – leidus mõningaid antud kategooria omadusi
Analüüsi	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Hinda	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Loo	**3 – kõige rohkem antud kategooria omadusi**	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud

2.3. Arvutiülesanded

Arvutiülesanded ehk arvestustöö arvutiosa lahendatakse arvutis ja erinevalt kirjalikest ülesannetest on lubatud kasutada materjale. Ülesanded lahendatakse programmeerimiskeskkonnas Thonny ja lahenduse esitamiseks peab kirjutatud programmid (vajadusel ka .txt sisendfail ja Thonny logifailid) Moodle’isse esitama. Töö edukaks läbimiseks tuleb arvutiosast saada 50% punktidest.

2.3.1. “Programmeerimise alused”

“Programmeerimise alused” statsionaarsel kursusel ja MOOCil on arvutiosa ülesandeks tavaliselt üks suur ja mitmekülgne programmeerimisülesanne. Selle asjaolu tõttu peab ülesanne olema hästi konstrueeritud, et kontrollitud saaks kõik vajalikud õpiväljundid. Kategoriseerimiseks on autor võtnud 2018. aasta MOOCi ülesande “Juubelid” (Lisa 5) ja 2019. aasta statsionaarse kursuse ülesande “Kangakauplus” (Lisa 6).

Nii “Juubelid” kui ka “Kangakauplus” on ülesehituselt sarnased ja koostatud viisil, kus ülesande teksti mõistmiseks ei ole vaja mõne valdkonna spetsiifilisi teadmisi. Muudetud on vaid ülesannete temaatikat.

Ülesandes “Juubelid” (Lisa 5) on vaja koostada ühe argumendiga funktsioon `juubelite_arv`. Argumendiks peab olema järjend ja funktsioon peab tagastama arvu, mitmel töötajal 2019. aastal juubel on. Kuigi materjalides on palju tsüklite ja funktsioonide näiteid, ei ole ükski neist päris selline, mis vastaks kontekstilt antud ülesandele. Õppija koostab antud funktsioonis midagi uut enda nägemuse järgi ja selle võib lugeda kategooria “Loo” omaduseks. Põhiprogramm, mille õppija koostama peab, seisneb alustuseks kasutajalt sisendi küsimises. Seejärel tuleb läbi viia faili töötlus – andmete lugemine järjendisse, mis saab olema argumendiks funktsioonile `juubelite_arv`. Täpselt selline koodisegment, kus loetakse failist andmeid reahaaval järjendisse, on praktikumi materjalides üks ühele olemas ja seega ei ole tegu millegi uue loomisega, vaid konkreetselt õpitu meenutamisega (sobivus “Pea meeles” kategooriaga). Seejärel tuleb väljastada (*print*) töötajate andmeid, kasutades loodud funktsiooni ja järjendit. Programmi töö lõppeb uuesti sisendi küsimisega ja *print* lausega.

Autor leiab, et ülesannet selgelt ühte konkreetsesse kategooriasse liigitada päris võimalik ei ole, kuid kõige õigem tundub see liigitada see kategooriasse “Loo” (tabel 12). Ülesandes oli rohkem tundmatut konteksti, kui tuttavat.

Tabel 12. MOOCi “Programmeerimise alused” arvutiosa ülesande kategooriad.

Kategooriad	Ülesanne “Juubelid”
Pea meeles	2 – leidus mõningaid antud kategooria omadusi
Saa aru	1 – antud kategooria omadusi ei leidunud
Rakenda	2 – leidus mõningaid antud kategooria omadusi
Analüüsi	2 – leidus mõningaid antud kategooria omadusi
Hinda	1 – antud kategooria omadusi ei leidunud
Loo	**3 – kõige rohkem antud kategooria omadusi**

Ülesandele “Kangakauplus” (Lisa 6) pealevaadates hakkab silma selle mahukus (teemade osas). Selle asemel, et kohe koodima hakata, võiks eeldada, et õppija jagab ülesande endale väiksemateks osadeks ja identifitseerib arvestuseks tähtsad komponendid (sobivus

“Analüüsi” kategooriaga). Ka kategoriseerimisel oli mõistlik ülesanne jällegi väiksemateks osadeks (teemade kaupa) jagada ja liigitada iga osa omaette kategooriasse.

Ülesande sooritamiseks on alustuseks vaja õppijal koostada kahe argumendiga funktsioon. Argumentideks on täisarv ja ujukomaarv, seega käsitletakse erinevaid andmetüüpe. Funktsiooni kirjutamise puhul on ka siin tegu uue terviku loomisega, sest kursuse materjalidest autor ligilähedast lahendust ei tuvastanud (sobivus “Loo” kategooriaga). Antud funktsioonis peab õpilane õige väärtuse tagastamiseks kasutama tingimuslauseid, mis on õpitud konstruktsioon ja kus peab lauseid kirjutama vaid õigete muutujaega või sobivate aritmeetiliste tehetega (sobivus “Rakenda” kategooriaga). Järgmisena on õpilasel tarvis vajalikud andmed failist enda programmi salvestada. Ujukomaarvude failist lugemise näitega on õppijad kursuse materjalidest üks ühele tuttavad, mistõttu ei ole siin tegu millegi uue loomisega, vaid õpitu meeldetuletamisega ja rakendamisega (sobivus “Rakenda” kategooriaga). Kattuvus materjalidega annab märku, et ülesanne “Kangakauplus” võiks liigituda Bloomi taksonoomia “Rakenda” kategooria alla (tabel 13). Siinkohal on aga potentsiaalne probleem õppijate tausta tundmatus – kõik õpilased ei pruugi kursuse materjale järjepidevalt lugeda või läbi töötada, seega nende õppijate teadmistest lähtudes oleks vale liigitada ülesannet “Rakenda” kategooriasse ja tegu oleks hoopis näiteks “Loo” kategooriaga (mõningate õpitud elementide kokku konstrueerimine). Kuna õppijatel ei palutud ülesandes midagi seletada ega sõnaliselt analüüsida, ei hõlmanud ülesanded näiteks Bloomi taksonoomia kategooriaid “Saa aru” ja “Hinda”.

Tabel 13. “Programmeerimise alused” statsionaarse kursuse arvutiosa ülesande kategooriad.

Kategooriad	Ülesanne “Kangakauplus”
Pea meeles	1 – antud kategooria omadusi ei leidunud
Saa aru	1 – antud kategooria omadusi ei leidunud
Rakenda	**3 – kõige rohkem antud kategooria omadusi**
Analüüsi	2 – leidus mõningaid antud kategooria omadusi
Hinda	1 – antud kategooria omadusi ei leidunud
Loo	2 – leidus mõningaid antud kategooria omadusi

2.3.2. "Programmeerimise alused II"

"Programmeerimise alused II" kursustel koosneb arvutiosa kolmest või neljast väiksemast kontrollülesandest. Ka sellel kursusel tuleb töö edukaks läbimiseks saada 50% punktidest. Autor on kategoriseerimiseks valinud 2019. aasta MOOCi kursuse arvutiosa ülesanded (Lisa 7) ja 2017. aasta statsionaarse kursuse kasutatud kontrolltöö ülesanded (Lisa 8).

MOOCi arvutiosa ülesandeid (Lisa 7) on kokku kolm. Esimene ja teine ülesanne on ülesehituselt osaliselt sarnased, kuid käsitlevad erinevaid teemasid. Esimene ülesanne seisneb kahemõõtmelise järjendi koostamises ja käsitlemises. Õppija peab lugema failist andmed kahemõõtmelisse järjendisse, kus sisemised järjendid tähistavad ühes failireas olevaid tähti. Järjendi kasutamiseks on vaja koodida funktsioon (sobivus "Loo" kategooriaga), mis võtab argumendiks kahemõõtmelise järjendi ja ühe sümboli (tähe) ning tagastab sümboli asukoha järjendis. Antud juhul on sarnane lahendus välja toodud ka õppematerjalides, kus on näitekoodile antud ka sama kontekst – "Vahel on vaja teada, kus täpselt teatud tingimustele vastav element on". Lahendajal on vaja muuta vaid tingimuslause enda ülesandega kohanduvaks. Sellest tulenevalt võiks ülesannet lugeda mälupõhiseks ja liigitada see kategooriasse "Pea meeles" (tabel 14).

Teine ülesanne on pisut laiahaardelisem kui esimene ning siinkohal tuleks kasuks ülesande väiksemateks osadeks jaotamine (sobivus "Analüüsi" kategooriaga). Ülesandes peab õppija esmalt funktsiooni abil failist andmed lugema sõnastikku (sobivus "Loo" kategooriaga). Seejärel tuleb luua programm, mis kasutab antud funktsiooni tagastamaks rallisõitja, kellel on rohkem punkte kui programmi sisendina etta antud sõitjal. Faili lugemine tuleks õppijal meenutada eelneva kursuse materjalidest (sobivus "Pea meeles" kategooriaga) ja rakendada seda uuele andmetüübile (sobivus "Rakenda" kategooriaga). Programmi koostamisel kasutab õppija eeldatavasti sõnastikust sobiva elemendi leidmiseks tsükleid, mis on õpitud kontseptsioon rakendatud antud juhul võõrale kontekstile ja andmetele, sest kursuse materjalidega kattuvus puudub. Kuna programmi loomine on aga ülesande põhiliseks osaks ja kattuvus materjalidega vaid osaline, kategoriseerib autor ülesande "Loo" kategooriasse (tabel 14).

Kolmandaks ülesandeks on koostada rekursiivne funktsioon (sobivus "Rakenda" kategooriaga) `paaritu_korrutis`, mis võtab argumendiks positiivse täisarvu ja tagastab paaritute arvude korrutise, mis on argumendist väiksemad või võrdsed. Antud

ülesande puhul praktikumi materjaliga kattuvust autor ei leidnud ja kategoriseerib selle kategooriasse “Loo” (tabel 14).

Tabel 14. MOOCi “Programmeerimise alused II” arvutiosa ülesannete kategooriad.

Kategooriad	Ülesanne 1	Ülesanne 2	Ülesanne 3
Pea meeles	**3 – kõige rohkem antud kategooria omadusi**	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Saa aru	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Rakenda	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi	2 – leidus mõningaid antud kategooria omadusi
Analüüsi	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Hinda	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Loo	2 – leidus mõningaid antud kategooria omadusi	**3 – kõige rohkem antud kategooria omadusi**	**3 – kõige rohkem antud kategooria omadusi**

Statsionaarse kursuse kontrolltöö (Lisa 8) esimese ülesande jaoks on vaja kirjutada programm koos abifunktsiooniga, mis arvutab argumentide alusel sündimata lapse pikkuse. Ülesandes on ette andud lapse pikkuse arvutamiseks kindlad valemid, seega on abifunktsiooni näol on tegu vaid kahe etteantud valemi implementeerimisega (sobivus “Rakenda” kategooriaga) kasutades tingimuslauseid, mis on eeldusaine “Programmeerimise

alused" kursuse raames tuttav kontseptsioon. Programmi osa kirjutamine ei ole samuti midagi uut – tegu on vaid funktsiooni välja kutsumise ja *print* lausetega. Kategoriseerimisel ülesannet analüüsides osutuski, et ühtegi uue teema käsitlust selles ülesandes tegelikult ei leidu ja see ülesanne on võimalik lahendada kursusel "Programmeerimise alused" õpitud teadmistega. Sellest tulenevalt on aga keerukas valida, kas ülesanne võiks liigituda Bloomi taksonoomia "Pea meeles" või "Rakenda" kategooriasse, kuna ülesandel on nende mõlema kategooriate omadusi. Kui võtta arvesse, et antud ülesande andmed on pisut teisel kujul, kui "Programmeerimise alused" kursuse materjalides, liigitab autor esimese ülesande "Rakenda" kategooriasse (tabel 15).

Teises ülesandes on vaja luua kaks funktsiooni ja programm. Ülesanne käsitleb korraga kahemõõtmelisi järjendeid ja kahekordseid tsükleid. Uurides kursuse materjale, on kahemõõtmelise järjendi töötlemist kahekordsete tsüklitega pigem õpetatud süntaktiliste näidete põhjal, seega töö sooritaja ülesande täpset lahendust näinud ei ole ja peab uuele olukorrale oskama rakendama õpitud protseduure. Seetõttu kategoriseerib autor ülesande Bloomi taksonoomia "Rakenda" kategooriasse (tabel 15).

Kolmandas ülesandes on vaja kirjutada kolm funktsiooni ja programm (sisuliselt funktsioonide väljakutsumine). Esmalt peab meelde tuletama "Programmeerimise alused" aines õpitud kasutajalt sisendi küsimist ja faili lugemist (sobivus "Pea meeles" kategooriaga). Küll aga on see väike (kuid tähtis) osa antud ülesandest ja suurem rõhk on ikkagi funktsioonide kirjutamisel. Antud funktsioonide sisu on kahemõõtmelise järjendi käsitlemine ja lähtuvalt kriteeriumitele õige väärtuse tagastamine. Kursuse materjalides õpilased sellise funktsiooniga kokku puutunud ei ole, kuid on tuttavad materjali ja komponentidega antud funktsioonide kirjutamiseks. Sellest tulenevalt võiks tõdeda, et ülesandes on tegu uute tervikute loomisega ja ülesande võib lugeda Bloomi taksonoomia "Loo" kategooriasse (tabel 15).

Neljas ülesanne keskendub vaid rekursiivsele funktsioonile ja rekursioon on teema, mida eelnevalt ehk kursusel "Programmeerimise alused" ei käsitleta. Kuna "Programmeerimise alused II" kursuse materjalides täpselt identset ülesannet (k.a sama probleemiga) lahendatud ei ole, puutub õppija nüüd kokku tundmatu ülesandega, millele ta saab rakendada talle nüüdseks tuttavat algoritmi ehk rekursiooni. Sellest tulenevalt võiks kategoriseerida ülesande Bloomi taksonoomia "Rakenda" kategooriasse (tabel 15). Küll aga saab öelda, et ülesandes leidub ka "Loo" kategooria jooni, sest õppija peab koostama uue funktsiooni.

Antud kursuse arvutiülesandeid on kategoriseerida mõnevõrra keerulisem, kui kursuse "Programmeerimise alused" arvutiülesandeid. Kategoriseerimist mõjutab asjaolu, et suur osa ülesannete sooritamisest seisneb ka kursuse "Programmeerimise alused" materjali meenutamises (sest kõike üle ei korrata) ja taksonoomia rakendamisel puudub selge juhis, kuidas võtta täpsemalt arvesse varasemalt omandatud teadmisi.

Tabel 15. "Programmeerimise alused II" statsionaarse kursuse arvutiosa ülesannete kategooriad.

Kategooriad	Ülesanne 1	Ülesanne 2	Ülesanne 3	Ülesanne 4
Pea meeles	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud	2 – leidus mõningaid antud kategooria omadusi	1 – antud kategooria omadusi ei leidunud
Saa aru	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Rakenda	**3 – kõige rohkem antud kategooria omadusi**	**3 – kõige rohkem antud kategooria omadusi**	1 – antud kategooria omadusi ei leidunud	**3 – kõige rohkem antud kategooria omadusi**
Analüüsi	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Hinda	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud
Loo	1 – antud kategooria omadusi ei leidunud	1 – antud kategooria omadusi ei leidunud	**3 – kõige rohkem antud kategooria omadusi**	2 – leidus mõningaid antud kategooria omadusi

2.4. Arvestustööde vastavus õpiväljunditega

Statsionaarse kursuse ja MOOCi "Programmeerimise alused" puhul on õpiväljundites märkimisväärne rõhk just kursuse nii-öelda teooria osal. Õpiväljunditena (peatükk 2.1, tabel 6) on üles loetletud näiteks oskus leida ja sõnastada programmeerimisülesandeid, programmeerimise ajaloo ülevaate omamine ja õpingute jätkamise võimalustest teadlik olemine. Küll aga jäävad need õpiväljundid ülesannete sisu poolest arvestustöödes tähelepanuta. Autor leiab, et "Programmeerimise alused" MOOCi kursustel on arvestustöö ülesanded vastavuses kolme õpiväljundiga seitsmest. Nendeks on:

- **oskus kasutada** erinevaid andmetüüpe, operatsioone jms (nt Lisa 1 ül 1 ja 2);
- oskus ülesandeid programmina **realiseerida** (nt Lisa 5);
- oskus programmeerimisülesandeid **lahendada** (nt Lisa 5).

Kui arvestustöö kõrval võtta arvesse aga lisaks kursuste loovtööd/projekti, saab kontrollitud ka õpiväljund oskus leida ja sõnastada programmeerimisülesandeid.

"Programmeerimise alused" statsionaarse kursuse korral leidis autor, et läbi ülesannete said kontrollitud kaks neljast püstitatud õpiväljundist. Nendeks on:

- oskus tekstina püstitatud ülesande **realiseerida** arvutiprogrammina (Lisa 6);
- oskab programmeerimise baaskonstruktsioone **esitada** nii plokkskeemidena kui ka programmilõikudena (Lisa 2).

Erinevalt teistest kursustest sisaldasid "Programmeerimise alused" statsionaarse kursuse õpiväljundid aga mõnevõrra mõõtmatuid protsesse nagu "motiveeritud kasutama arvutit" ja "motiveeritud koostama programme", mis said eelkõige kategoriseeritud verbide *kasutama* ja *koostama* järgi, kuid ka need õpiväljundid saavad sarnaselt MOOCi kursusel kontrollitud läbi kursuse loovtöö/projekti.

Kursusel "Programmeerimise alused II" on õpiväljundeid (peatükk 2.1, tabel 7) sõnastatud rohkem ja ülesannete kategoriseerimise läbi leidis autor, et kontrollitud said kõik kursusel püstitatud õpiväljundid. Autor ei leidnud otsest vastavust ülesannete näol vaid õpiväljunditega nagu oskus programme siluda ja oskus vormistada lahendusprogramme. Need oskused on arvestatud arvutiülesannete sisse – töö juhendite alusel on lahenduse esitamise eeldusena kirjas, et õppija vormistab enda lahendused süntaktiliselt korrektselt ja

loodud programmid ei anna veateateid. Mõned näited õpiväljundite vastavusest ülesannete sisuga on:

- oskus programme **selgitada** (nt Lisa 4 ül 1);
- oskus lahendusprogramme **testida** (nt Lisa 4 ül 1);
- oskus standardoperatsioone **kasutada** (nt Lisa 7 ül 2);
- oskus **luua** algoritme (nt Lisa 8 ül 4);
- oskus **koostada** lahendusprogramme (nt Lisa 7 ül 2).

Üldjoontes on kõigil neljal kursusel kasutatavad ülesanded hästi valitud ja konstrueeritud ning kirjalikud ja arvutiülesanded kokku kontrollivad suurel osal kursustel märgitud õpiväljunditest. Autoripoolse soovitusena võiks aga ülesannete mitmekesisuse parandamiseks esineda rohkem "Hinda" ja "Analüüsi" kategooriatesse kuuluvaid ülesandeid. Näiteks võiks ülesannetesse rohkem sisse tuua koodi kommenteerimist, olgugi et iseenesest mõistetavat koodi (informatiivsete muutujate ja funktsioonide nimedega) võib lugeda programmeerimise heaks tavaks. See tooks aga rohkem tähelepanu õppijate analüüsioskusele ja õppijate tegevus oleks potentsiaalselt rohkem mõtestatud.

Bloomi taksonoomiast oli kasu arvestustöö ülesannete õpiväljunditele vastavuse kontrollimisel. Õpiväljundid on kursustel kirja pandud kasutades sõnu nagu *oskama*, *seletama*, *sõnastama*, *realiseerima* jpm, mis võimaldab lihtsasti määrata nende Bloomi taksonoomia kategooria. Kui on teada ka ülesande kategooria, siis saab nende vastavust õpiväljunditega paremini kontrollida. Juhul kui proovida ülesandeid õpiväljunditega sobitada ilma õpiväljundeid kategoriseerimata, võivad osapoolte tulemused tulla kohati väga erinevad. Õpiväljundite kategoriseerimine Bloomi taksonoomia alusel aitab subjektiivsust vähendada, küll aga ei kõrvalda seda.

2.5. Kategoriseerimise raskused

Kategoriseerimine osutus keeruliseks protsessiks ja selle käigus tekkis mitmeid küsimusi, mis loetud materjali põhjal suures osas vastamata jäid. Üheks suurimaks probleemiks olid kategooriate vahelised hägused piirid. Kõige enam esines kõhklusi Bloomi taksonoomia kategooriate "Pea meeles" ja "Rakenda" ning "Rakenda" ja "Loo" vahel. Autor mõistis, et "Pea meeles" kategooriasse kuulumine seisneb teadmiste meenutamises ja õpitut rakendamises probleemile, mille lahendust on varem täpselt nähtud. Autoril tekkis aga

kõhklusi, kui mälupõhise kategooriaga on tegu ja kui täpne võiks tähendada "täpne". Jäi segaseks, et kas "Pea meeles" kategooria all mõeldakse ülesande/koodi üks ühele samasust või pigem lihtsalt sarnasust, näiteks ülesehituselt. Autor tundis pärast kategoriseerimist, et mõningates kohtades olles ülesande liigitanud kategooriasse "Pea meeles", võis tegu olla vale otsusega. Töö vältel sai kategoriseerimise käigus selgemaks, et "Pea meeles" kategooria võiks esineda rohkem vaid kirjalike ülesannete puhul, kui et programmeerimiseülesannete seas.

Kategooriate "Rakenda" ja "Loo" vahel tekkis aga küsimusi, et kuidas kategooriaid ikkagi üksteisest eristada. Kui kategooria "Loo" puhul räägitakse justkui uute lahenduste loomisest, siis kuidas hinnata õppija poolt kirja pandud vastuse uudsust? Mis piirist ei ole tegu enam "Rakenda" kategooriasse sobituva ülesandega (küsimuse/lahendusega), mille puhul räägime samuti kas algoritmide või muude kontseptsioonide rakendamisest millegi koostamiseks? Antud kategooriad tõestasidki end olema sisu poolest üpris sarnased. Mõlemad seisnevad õpitud materjali rakendamises, küll aga natukene erinevate nurkade alt.

Kategooriate sisulise kattuvuse probleem oli ka ühe kriitikana välja toodud teadustöös "*Developing a Computer Science-specific Learning Taxonomy*" [2]. Antud kriitikaga autor nõustub ja leiab, et kattuvus kategooriate vahel tõepoolest raskendab kategoriseerimist. Küll aga üldjoontes tööst rohkem antud peatükis mainitud kriitikat ei ilmnenud, paljustki just seetõttu, et antud bakalaureusetöö üheks osaks ei olnud õpilaste lahenduste analüüsimine.

Järgmise raskusena ilmnes raskus hinnata, et millist lahendust õpilane ülesande vastusena esitada võib. Kuna antud bakalaureusetöö õppijate arvestustööde lahendusi ei analüüsinud, et leida näiteks iga ülesande vastustes kasutatud kõige sagedasemad lahendused, mõjutas see kategoriseerimise protsessi. Autor proovis lähtuda kõige enam kursuste materjalis õpetatust.

Raskuseks oli kategoriseerimisel ka õige tasakaalu leidmine, et kui palju arvestada küsimuse ja/või etteantud koodi sisu, et määrata kõige õigem kategooria. Teadustöös "*Bloom's Taxonomy for CS Assessment*" [18], mis oli kategoriseerimisel abiks, olid näiteülesanded enamjaolt kategoriseeritud vaid lähtuvalt küsimusest. Kuna kategoriseerimise juures räägiti aga palju õppematerjali olulisusest ülesannete lahendamisel, sai kiiresti selgeks, et ainult ülesande küsimusest lähtudes liigitada ei saa. Väga tähtsaks osaks kategoriseerimisel on õppematerjali tundmine.

Lisaks olid teadustöös "*Bloom's Taxonomy for CS Assessment*" [18] kategoriseeritud näiteülesanded väga lühikesed ja ühekülgsed. Kursuste "Programmeerimise alused" ja "Programmeerimise alused II" ülesanded olid seevastu aga sisukamad ja keerulisemad, koosnedes mitmetest erikomponentidest ning kontrollides korraga mitmeid kursuse teemasid ja õpiväljundeid. Sellest tulenevalt oleks autor siinkohal soovinud leida rohkem informaatikakeskseid teadustöid, kus oleks olnud läbi viidud ka keerukamate näiteülesannete kategoriseerimist.

Kategoriseerides sai ka selgeks, et õige kategooria leidmise teeb lihtsamaks ülesannete väiksemateks osadeks jagamine ja seda just suuremate, mitut alaülesannet sisaldavate ülesannete puhul. Kategoriseerimist alustades ilmnes, et täielikult ükski ülesanne ühteainsassse kategooriasse ei liigitu, vaid koosneb erinevate kategooriate oskustest ja omadustest ning lõppkokkuvõttes tasub teha järeldused selle põhjal, et millise kategooria omadused on ülekaalus.

Samuti peab enne kategoriseerimist selgelt paika panema täpsed eeldused (veelgi täpsemalt, kui seni). Antud bakalaureusetöös sai sõnastatud, et ülesanded on kategoriseeritud eeldades, et õppijate programmeerimisteadmised on omandatud vaid kursuste "Programmeerimise alused" ja "Programmeerimise alused II" raames. Küll aga tekkis töö käigus mõtteid, et kas peaks ka lähtuma õppijatest (täpsemini nende teadmistest), kes ei pruugi materjali iganädalaselt läbi töötada ja kelle teadmised osade teemade suhtes on seeläbi puudulikud. See on asjaolu, mis mõjutab kategoriseerimist suuresti, sest sellised õpilased kalduvad ülesannete lahendamisel kasutama teisi oskusi kui õpilased, kes on materjali täies mahus omandanud. Antud bakalaureusetöö jättis selle aspekti siiski arvestamata.

Kokkuvõttes on Bloomi taksonoomiaga kategoriseerimise puhul tegu veel väga subjektiivse ja keerulise tegevusega ning selleks, et Bloomi taksonoomiat oleks võimalik veelgi teaduslikumalt kasutada, peaks seda erialastest teadmistest lähtuvalt täiendama. Kategoriseerimisel peab arvesse võtma väga mitmeid nüansse – õppematerjale, küsimuse sisu, etteantud koodi sisu, õpilase võimalikke teadmisi, ülesannete võimalikku lahendust. Kuigi kõiki detaile ja ülesandetüüpe ei oleks võimalik taksonoomiasse kirjutada, looks Bloomi taksonoomia täiendamine mõne kursuse tarbeks kategoriseerimisele selgemad piirid ja vähendaks kõhklusi. Bloomi taksonoomia rakendamine juba välja arendatud kursusele ehk olemasolevale materjalile on mõõdukalt tülikas protsess ja selle rakendamine tundub perspektiivikam alles arendamisjärgus olevale kursusele.

3. Kokkuvõte

Antud bakalaureusetöö andis ülevaate Bloomi taksonoomiat käsitlevast kirjandusest ja bakalaureusetöö käigus püüdis autor kategoriseerida programmeerimise algkursuste arvestustöö ülesandeid. Valitud kursusteks olid statsionaarsed kursused ja MOOCid "Programmeerimise alused" ning "Programmeerimise alused II". Bakalaureusetöö eesmärk oli leida kategoriseerimisel tekkivad raskused ja teha kindlaks, kas ülesanded kontrollivad nimetatud kursuste õpiväljundeid. Autor otsustas kategoriseerimise läbi viia kasutades Bloomi taksonoomiat, mille puhul on tegu ühe levinuima haridusliku taksonoomiaga ja mida kasutatakse kursuste loomisel tihti õpiväljundite defineerimisel. Sellest tulenevalt saab seda kasutada ka õpiväljundite kontrollimiseks. Antud bakalaureusetöö aluseks olev modifitseeritud Bloomi taksonoomia jaotub kuueks kategooriaks: "Pea meeles", "Saa aru", "Rakenda", "Analüüsi", "Hinda" ja "Loo".

Bakalaureusetöös kirjeldati laiemalt taksonoomia mõistet ja Bloomi taksonoomia sisu, et viia lugeja kurssi kategoriseerimiseks vajalike teadmistega. Lisaks oli välja toodud Bloomi taksonoomiat puudutav kriitika ja ülevaated mõningatest teadustöödest, mis käsitlevad Bloomi taksonoomiat informaatikavallas.

Töö raames kategoriseeris autor kokku kaheksa arvestustöö variandi ülesandeid. Nendeks ülesanneteks olid nelja kirjaliku arvestustöö ülesanded ja nelja arvuti arvestustöö ülesanded eelnimetatud kursustelt. Iga arvestustöö variandi ülesanded olid kategoriseerimise järgselt illustreeritud kokkuvõtva tabeliga, et lugejal oleks ülesannete kategoriseerimise tulemustest parem ülevaade.

Pärast ülesannete kategoriseerimist pani autor kirja raskused ja probleemid, millega ta kategoriseerimisel kokku puutus ja andis kategoriseerimise põhjal hinnangu, kas õpiväljundid on vastavuses ka ülesannetes kontrollitavate teadmistega. Raskustena olid bakalaureusetöös välja toodud näiteks kategooriate vahelised hägused piirid ja informaatikaalast kategoriseerimist toetava kirjanduse puudujääk.

Õpiväljundite kontrollimisel selgus, et kursustel kasutatavad arvestustööd tervikuna (kirjalikud ja arvutiülesanded kokku) kontrollivad suures osas kursustele sõnastatud õpiväljundeid. Vähem oli kontrollitud nii-öelda teooriat puudutavaid õpiväljundeid, näiteks programmeerimiseajaloo ülevaate omamine. Õpiväljundite ja arvestustöö ülesannete sisu vastavuse ebakõla esines vaadeldavatest kursustest rohkem MOOCi kursusel "Programmeerimise alused".

Autor loodab, et antud bakalaureusetöö julgustab õpetajaid ja õppejõude kursuste ning ülesannete koostamisel lähenema senisest veelgi teaduslikumalt ja struktureeritumalt, kasutades abivahendina näiteks just Bloomi taksonoomiat või ka teisi hariduslikke taksonoomiaid. Taksonoomiast lähtumine võimaldab ühtlustada sama kursuse õpetajate ja õppejõudude arusaama kursuse eesmärkidest ja ülesehitusest, vähendades võimalikke arusaamatusi ja potentsiaalselt parandades õpitulemusi (vähenevad võimalikud erinevused õpetamisel). Kuna praegu veel on Bloomi taksonoomia alusel informaatika materjalide kategoriseerimine aga raskemapoolne tegevus, peaks kategoriseerimise aluseks olevaid materjale esmalt täiendama. Antud bakalaureusetöö edasiarendusena oleks võimalik näiteks Tartu Ülikooli programmeerimise algkursuste jaoks koostada Bloomi taksonoomia põhjal uus informaatikaspetsiifiline taksonoomia, mida saaks kergesti rakendada ka teiste programmeerimiskursuste ülesannete koostamiseks, hindamiseks ja arendamiseks ning kursuse struktuuri täiendamiseks.

Viidatud kirjandus

[1] Pors M. Maailmas populaarne programmeerimisvahend nüüd eesti keeles. 2014. https://koolielu.ee/info/readnews/352268/maailmas-populaarne-programmeerimisvahend-nuud-eesti-keeles (16.04.2019)

[2] Fuller U, Johnson CG, Ahoniemi T, Cukierman D, Hernán-Losada I, Jackova J, et al. Developing a Computer Science-specific Learning Taxonomy. Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education. New York, NY, USA: ACM; 2007. 152–170.

[3] Schweizer C. What is the Difference between Taxonomy and Ontology? It is a Matter of Complexity. https://www.earley.com/blog/what-difference-between-taxonomy-and-ontology-it-matter-complexity (21.04.2019)

[4] Smith KB. Typologies, Taxonomies, and the Benefits of Policy Classification. Policy Stud J. 2002; 30(3): 379–395.

[5] Krull E. Pedagoogilise psühholoogia käsiraamat. Tartu: Tartu Ülikooli Kirjastus; 2018.

[6] O'Neill G, Murphy F. Guide to Taxonomies of Learning. University College Dublin; 2010.

[7] MOOC Programmeerimise alused 2018/19 sügis. https://courses.cs.ut.ee/2018/eprogalused/fall (05.01.2019)

[8] Programmeerimise alused 2018/19 kevad. https://courses.cs.ut.ee/2019/prog-alused/spring (25.03.2019)

[9] MOOC Programmeerimise alused II 2018/19 kevad. https://courses.cs.ut.ee/2019/eprogalused2/spring (05.01.2019)

[10] Programmeerimise alused II 2017/18 kevad. https://courses.cs.ut.ee/2018/prog-alusedII/spring (05.01.2019)

[11] Coleman V. On the reliability of applying educational taxonomies. 2017; (24): 30–37.

[12] Armstrong P. Bloom's Taxonomy. Vanderbilt University. 2010. https://wp0.vanderbilt.edu/cft/guides-sub-pages/blooms-taxonomy/ (13.01.2019)

[13] Taxonomies of Learning. https://bokcenter.harvard.edu/taxonomies-learning (13.01.2019)

[14] Biggs J, Tang C. Õppimist väärtustav õpetamine ülikoolis. 3. tr. Tartu Ülikooli Kirjastus; 2007.

[15] Kirk K. What is the Affective Domain anyway?. Student Motivations and Attitudes: The Role of the Affective Domain in Geoscience Learning. https://serc.carleton.edu/NAGTWorkshops/affective/intro.html (13.01.2019)

[16] Simpson E. Educational objectives in the psychomotor domain. Behavioral Objectives in Curriculum Development: Selected Readings and Bibliography. Educational Technology; 1971. 60–67.

[17] Shuhidan S, Hamilton M, D'Souza D. A Taxonomic Study of Novice Programming Summative Assessment. Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95. Darlinghurst, Australia, Australia: Australian Computer Society, Inc.; 2009. 147–156.

[18] Thompson E, Luxton-Reilly A, Whalley JL, Hu M, Robbins P. Bloom's Taxonomy for CS Assessment. Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78. Darlinghurst, Australia, Australia: Australian Computer Society, Inc.; 2008. 155–161.

[19] Rõmmel K. E-kursuse „Programmeerimise alused II" kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs. Tartu: Tartu Ülikool; 2017.

[20] Tagam K. E-kursuse „Programmeerimise alused II" rekursiooni temaatika küsimuste ja ülesannete loomine. Tartu: Tartu Ülikool; 2017.

[21] Hendrikson H. MOOCi „Programmeerimise alused" ülesannete lahenduste analüüs. Tartu: Tartu Ülikool; 2018.

[22] Atkinson SP. Taxonomy Circles: Visualizing the possibilities of intended learning outcomes. BPP Univ Coll. 2013.

[23] Lahtinen E. A Categorization of Novice Programmers: A Cluster Analysis Study. Proc 19th Annu Workshop Psychol Program Interest Group. 2007; 16: 32–41.

[24] ÕIS2. https://ois2.ut.ee/ (07.05.2019)

Lisad

I. “Programmeerimise alused” MOOCi kirjalikud ülesanded

Paberosa A

Paberosa ülesanded lahendatakse ilma arvutita ja ilma materjalide abita. Paberosa lahenduste äraandmisaja otsustab kursuslane (soovitatav lahendusaeg on 30 minutit). Kokku on kolm ülesannet, igaüks maksimaalselt 10 punkti. Arvestuseks on vaja vähemalt 24 punkti.

Ülesanne 1

```
sisend = int(input("Palun sisestage täisarv: "))
if sisend > 1:
    if sisend > 3:
        print("Variant 1")
else:
    if sisend <= 3:
        print("Variant 2")
    else:
        print("Variant 3")
```

Mis väljastatakse ekraanile, kui kasutaja sisestab

- 0
- 1
- 3
- 4

Ülesanne 2

```
järjend = [5, 4, 3, 8, 11]

print(järjend[järjend[2]])

print("----")

for element in järjend:
    if element % 3 == 2: # jääk 3-ga jagamisel
        print(element)
```

Mis väljastatakse ekraanile?

Ülesanne 3

```
def poolitaja(arv):
    return arv / 2

m = 16.0
print(poolitaja(m))
print("---")
while m > 3:
    m = poolitaja(poolitaja(m))
    print(m)
```

Mis väljastatakse ekraanile?

II. “Programmeerimise alused” statsionaarse kursuse kirjalikud ülesanded

Loengu kontrolltöö variant 16

Aega on 35 minutit. Materjale kasutada ei tohi.

Kontrolltöö arvestuseks on vaja vähemalt 12 punkti (maksimaalne on 24). Kokku on loengu arvestuseks vaja vähemalt 20 punkti.

Ülesannne 1. 8 p

Müügiautomaat küsib kasutajalt küsimisvoorus kõigepealt salasõna, milleks on “Sala123” ja seejärel tootekoodi (täisarv 1, 2, 3 või 4). Kui sisestatud salasõna on vale või sisestatud tootekood on midagi peale 1, 2, 3 või 4, siis küsitakse neid mõlemaid uuesti (uus küsimisvoor). Protsessi korratakse kuni lõpuks sisestatakse õiged või on toimunud 10 küsimisvooru. Kui sisestatakse õige salasõna ja tootekood, siis väljastatakse nõutud tootekood ja töö lõpeb. Kui ka 10. küsimisvoorus pole salasõna ja tootekood õiged, siis töö lõpeb.

Koostage tsükliline plokkskeem.

Ülesannete 2 ja 3 puhul on ülesande kirjeldus ühesugune. Milliseid väärtusi saavad muutujad, parameetrid? Mida väljastatakse järgmise programmi töötamisel ekraanile? Põhjendada vastuseid. Veateate korral pole vajalik selle täpse teksti esitamine, vaid põhjuse kirjeldamine.

Ülesanne 2. 8 p

```
kala = "tint"
kalad = ["rääbis", "lest", kala]
print(min(kalad))
print(kala[len(kalad) // 2])
print("a" not in kala)
for i in range(0, 4, 2):
    print(kala[i])
```

Ülesanne 3. 8 p

```
def f(x, y):
while x > y:
    x -= 2
    if x < 0:
     return y
return x
print(f(1, f(4, 2)))
```

III. “Programmeerimise alused II” MOOCi kirjalikud ülesanded

Paberosa ülesanded

Kontrollitud oludes arvestustöö koosneb paberosast ja arvutiosast. Kokku on aega 180 minutit. Paberosa lahenduste äraandmisaja otsustab kursuslane. Arvutiosa ülesanded lahendatakse pärast paberosa lahenduste äraandmist.

Paberosa ülesanded lahendatakse

- ilma arvutita;
- ilma materjalide abita.

Positiivseks hindeks on vaja saada vähemalt 50% punktidest eraldi nii paberosa kui arvutiosa eest. Paberosa eest on vaja saada vähemalt 25 punkti.

Koguhinde alampiirid on "E" - 51%, "D" - 61%, "C" - 71%, "B" - 81%, "A" - 91%.

Ülesanne P1 (10 punkti)

Järgnev programmilõik väljastab kahemõõtmelise täisarvude järjendi `a` korral, mitu ühest suuremat arvu on programmeerimise mõistes paarituarvuliste indeksitega ridade hulgas. Programm töötab ka sellisel juhul, kui reas ei ole ühtegi elementi.

```
kokku = 0
for i in range(1, len(a), 2):
    for j in range(len(a[i])):
        if a[i][j] > 1:
            kokku += 1
print(kokku)
```

Koostada funktsioon `paaritus_reas`, mille puhul alltoodud programmilõik töötaks ülaltooduga võrdväärselt.

```
kokku = 0
for i in range(1, len(a), 2):
    kokku += paaritus_reas(a[i])
print(kokku)
```

Ülesanne P2 (10 punkti)

Mis väljastatakse programmi töötamisel ekraanile?

```
lst1 = [5, 5, 3]
lst2 = lst1[:]
lst3 = lst2

lst2[1] += 1
lst1[1] = 1
lst3[2] = lst2[2] - 1

print(lst1)
print(lst2)
print(lst3)
```

Ülesanne P3 (10 punkti)

Mis väljastatakse programmi töötamisel ekraanile?

```
sõna = ["siluma", "programmeeritav", "maalähedane",
"Thonny"]
tüüp = ["omadussõna", "nimisõna", "asesõna"]
tüüp[2] = "tegusõna"
sõnastik = {sõna[3]: tüüp[1],
            sõna[1]: tüüp[0]}
sõnastik["väljastama"] = tüüp[2]
print(sõnastik["väljastama"])
for a in sõnastik:
    print(sõnastik[a])
```

Ülesanne P4 (10 punkti)

Mis väljastatakse programmi töötamisel ekraanile?

```
h1 = {0, 2, 3}
h2 = {1, 9, 7}
h2.add(7)
print(h2)
h3 = {3, 4, 7} ^ h1
print(h3)
h4 = {2, 0, 3, 5}
h5 = h4 | h1 & h2
print(h5)
```

Ülesanne P5 (10 punkti)

Mis väljastatakse programmi töötamisel ekraanile?

```
def f(x):
    print(x)
    if x <= 2:
        return x
    else:
        return f(x - 2) + 2

print("Tulemus:", f(6))
```

IV. “Programmeerimise alused II” statsionaarse kursuse kirjalikud ülesanded

Programmeerimise alused II

Eksam 11. juunil 2018

Eksam kestab 90 minutit

- ilma arvutita,
- ilma materjalideta,
- ilma suhtlemiseta.

Maksimaalselt on võimalik saada 26 punkti, positiivse hinde jaoks peab saama vähemalt 13 punkti

Kõiki vastuseid tuleb põhjendada! Veateate korral pole vajalik selle täpse teksti esitamine, vaid põhjuse

kirjeldamine.

Punkte

1.

2.

3.

4.

5.

Kokku

Ülesanne 1 (6 punkti)

Järgnev programmilõik väljastab kahemõõtmelise järjendina esitatud täisarvude tabeli a korral, mitu rida on sellised, milles kõik elemendid on võrdsed (n-ö ühtlased read).

```
ühtlasi_ridu = 0
for r in a:
     on_ühtlane = True
     for v in r:
          if not r[0] == v:
               on_ühtlane = False break
     if on_ühtlane:
          ühtlasi_ridu += 1

print("Ühtlasi ridu on " + str(ühtlasi_ridu))
```

Koostada funktsioon leidub_erinev, mille puhul alltoodud programmilõik töötaks ülaltooduga võrdväärselt.

```
ühtlasi_ridu = 0
     for r in a:
          if not leidub_erinev(r):
               ühtlasi_ridu += 1

print("Ühtlasi ridu on " + str(ühtlasi_ridu))
```

Ülesanne 2 (4 punkti)

Mida väljastatakse programmi töötamisel ekraanile?

```
järjenda = [2, 1, -2, 4]
järjendb = järjenda
järjendb[1] = 5
print(järjenda[1])
järjendc = järjenda[:]
järjenda[3] = 7
print(järjendc[3])
```

Ülesanne 3 (6 punkti)

Mida väljastatakse programmi töötamisel ekraanile?

```
linnad = ["Riia", "Pihkva", "Vilnius", "Kaunas"]
riigid = ("Venemaa", "Läti", "Leedu", "Poola")
geokogu = {linnad[1]: riigid[0], linnad[0]: riigid[1],
          linnad[3]: riigid[2]}
geokogu["Vilnius"] = riigid[2]
for v in geokogu.keys():
     print(geokogu[v])
geokogu2 = geokogu
del geokogu2["Läti"]
print(geokogu["Leedu"])
```

Ülesanne 4 (4 punkti)

Mida väljastatakse programmi töötamisel ekraanile?

```
h1 = {3, 1, 8}
h1.add(8)
print(h1)
h2 = {3, 5, 4, 7}
h3 = {1, 2, 4}
h4 = h1 | h2 & h3
print(h4)
print({2, 5} ^ h3)
```

Ülesanne 5 (6 punkti)

Mida trükib ekraanile järgmine programmilõik?

```
def rekFun(x, y):
     print(str(x) + " " + str(y))
     if x < 0:
          return y
     else:
          return rekFun(x-2, y+1) + rekFun(x-3, y-3)
print(rekFun(2, 6))
```

V. “Programmeerimise alused” MOOCi arvutiülesanded

Arvestustöö 15.12.2018

Materjale võib kasutada, suhelda ei tohi.
Esitatav programm peab töötama ja tegema seda, mis ette nähtud. Programm peab vastama ülesandes kirjeldatud nõuetele (sisaldama vajalikke funktsioone jne). Kui funktsioon peab midagi tagastama, siis funktsiooni sees peab olema võtmesõna `return`. Programm peab käivituma ja ei tohi näidata veateateid.
Lahendus ja Thonny logi(d) tuleb esitada Moodle-is (Arvestustöö 15.12) ja saata aadressile XXXXX.

Ülesanne
Ühes firmas on komme kinkida töötajale juubeli (20, 30, 40, ...) puhul kristallvaas. Vaaside tellimiseks on vaja loendada, mitmel töötajal on 2019. aastal juubel. Kõigi töötajate sünniaastad on kirjas tekstifailis (tehke see fail ise). Näiteks

```
1980
1969
1985
1956
1979
```

Kui firmasse tuleb tööle uus töötaja, siis tahetakse teada, mitme töötajaga on tal sama sünniaasta.

Koostage funktsioon `juubelite_arv`, mis saab argumendiks aastate järjendi ja tagastab mitmel töötajal on 2019. aastal juubel. Võimalik on kontrollida, kas aasta lõpeb 9-ga või vanus (`2019 - aasta`) jagub 10-ga.

Koostada programm, mis

- küsib kasutajalt, mis failis sünniaastad on;
- loeb failist sünniaastad ja paneb need järjendisse;
- arvutab ja väljastab ekraanile töötajate koguarvu;
- arvutab (ülalmainitud funktsiooni abil) ja väljastab ekraanile nende töötajate arvu, kel on aastal 2019 juubel;
- küsib kasutajalt uue töötaja sünniaasta ja väljastab ekraanile, mitu töötajat on sündinud temaga samal aastal.

PROGRAMMI TÖÖ NÄIDE ON PÖÖRDEL

Funktsiooni `juubelite_arv` kasutamise näide:

```
>>> juubelite_arv([1980, 1969, 1985, 1980, 1956, 1979,
1991])
2
```

Näiteks faili *aastad.txt* sisu

```
1980
1969
1985
1980
1956
1979
1991
```

korral peab programm andma tulemuse (kasutaja sisend on siin näidatud kaldkirjas):

```
Sisestage failinimi: aastad.txt
Firmas on 7 töötajat
Aastal 2019 on juubel 2 töötajal
Sisestage uue töötaja sünniaasta: 1980
2 töötajat on sündinud samal aastal
```

VI. "Programmeerimise alused" statsionaarse kursuse arvutiülesanded

Programmeerimise alused
Arvutikontrolltöö 02.04.2019 kell 14.15-16.00

Oluline info:

- Materjale (courses leht, enda tehtud kodu- ja praksiülesanded jne) **võib kasutada.**
- Suhtlemine on **keelatud!**
- Moodle'is tuleb esitada **oma programm** (.py)**, sisendfail** (.txt), ja **logid!**

Logifailidest:

Enne lahendamist **sulgege Thonny ja käivitage uuesti** – nii ei jää logidesse üleliigset infot. Plagiaadikahtluste vältimiseks **ärge kopeerige Thonnysse koodijuppe**, kuna kõik *copy-paste* tegevused on logidest näha ja tekitavad küsimusi, kas kood on ikka enda kirjutatud.

Kui olete ülesandega valmis, siis sulgege ja taaskäivitage Thonny veelkord, et uued logid salvestuks. Valige menüüst *Tools → Open Thonny data folder…* → ava kaust *user_logs.* Valige kõik kontrolltöö ajal tekkinud logid (kui te Thonnyt ülesande lahendamise keskel ei sulgenud, siis ongi ainult üks fail.
Aega on täistunnini. Edukat lahendamist! :)

Kontrollülesanne. Kangakauplus

Restorani soovitakse uusi kardinaid tellida ning selleks külastatakse kohalikku kangapoodi, kus on hetkel käimas kampaania, mille raames on üle 6 meetri kanga ostmisel sellest kardinate valmistamine tasuta. Koosta programm, mis lihtsustab kardinate tellimist.

Tekstifailis on toodud erinevate kangaste meetrihinnad ujukomaarvudena (*float*) nii, et igal real on ühe konkreetse kanga meetri hind. Näiteks võib faili sisu olla selline:

12.24
16.60
11.45
13.68

Defineeri funktsioon, mis võtab argumentideks kanga meetri hinna ning soovitud koguse (meetrites) ja tagastab ostu kogusumma ümardatuna kahe komakohani. Kanga hinnale (meetri hind * kogus meetrites) lisandub ka kardinate valmistamise tasu 8 eurot. Eespool mainitud kampaania tõttu **ei lisandu** valmistamise tasu juhul, kui kangast ostetakse rohkem kui kuus meetrit. Järgnevalt näide funktsiooni tööst:

`maksumus(9.2, 5.8)` tagastab 61.36 *(sest 9.2*5.8+8)*
`maksumus(9.2, 6.2)` tagastab 57.04 *(sest 9.2*6.2)*

Lisaks koosta programm, mis...

- Küsib kasutajalt failinime, tema eelarve (ujukomaarvuna) ning soovitud kanga koguse (ujukomaarvuna).
- Loeb tekstifailist järjendisse kangaste meetrihinnad.
- Väljastab defineeritud funktsiooni abil iga kanga ostu kogusumma. Kui ostu maksumus ületab kasutaja eelarvet, tuleb kasutajat sellest teavitada.
  NB! Kood peaks töötama sõltumata sellest, mitu rida tekstifailis on!
- Küsib kasutajalt, mitmendat kangast ta osta soovib. (Võib eeldada, et kasutaja ei vali toodet, mis on tema jaoks liiga kallis.)
- Väljastab ekraanile, kui palju raha kulus kasutajal ostu peale ning kui palju alles jäi (samuti ümardatuna 2 komakohani).

Järgnevalt näide programmi tööst (kasutaja sisend on kaldkirjas):

```
Sisesta failinimi: hinnad.txt
Sisesta eelarve: 115.3
Sisesta kanga kogus: 8.7
Tootevaliku hinnakiri on järgmine:
1. toote maksumus on 106.49 eurot.
2. toote maksumus on 144.42 eurot. --See on liiga kallis!
3. toote maksumus on 99.61 eurot.
4. toote maksumus on 119.02 eurot. --See on liiga kallis!
Mitmendat toodet osta soovid? 3
Ost sooritatud! Kokku kulus 99.61 eurot. Alles jäi 15.69 eurot.
```

NB! Loe ülesande juhis veelkord läbi ja pööra erilist tähelepanu allajoonitud ja **paksus kirjas** osadele. Kontrolli, kas sinu kood vastab kõigile kirjeldatud nõuetele!

VII. “Programmeerimise alused II” MOOCi arvutiülesanded

Arvestustöö kursusel *Programmeerimise alused II*

30. märts 2019

Arvutiosa ülesanded

Arvutiosa ülesanded lahendatakse

- pärast paberosa lahenduste äraandmist;
- arvutiga kasutades programmeerimiskeskkonda Thonny;
- vajadusel materjale kasutades.

Lahendusprogramm ja Thonny logifailid esitatakse Moodle’is (automaatkontrolli ei ole).

Positiivseks hindeks on vaja saada vähemalt 50% punktidest eraldi nii paberosa kui arvutiosa eest. Arvutiosa eest on vaja saada vähemalt 25 punkti.

Koguhinde alampiirid on "E" - 51%, "D" - 61%, "C" - 71%, "B" - 81%, "A" - 91%.

Ülesanne A1 - Salakood (20 punkti)

Faili *tähestik.txt* (kodeering UTF-8) on salvestatud tähed selliselt, et iga kahe tähe vahel on tühik. Igas reas on täpselt sama arv tähti. Tähtede arv reas ja ridade arv võib erineda erinevates failides. Programm peab lugema faili sisu kahemõõtmelisse järjendisse, kus sisemised järjendid sisaldavad ühes reas olevaid tähti.

Näide tekstifaili *tähestik.txt* (Moodle’is leitav) võimalikust sisust:

```
a b c d e f g h
i j k l m n o p
q r s š z ž t u
v w õ ä ö ü x y
```

Eelnevast failist loetud ridade põhjal moodustuks näiteks kahemõõtmeline järjend järgneval kujul (siin näidises paremaks illustreerimiseks mitmele reale paigutatud!):

```
[['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
['i', 'j', 'k', 'l', 'm', 'n', 'o', 'p'],
['q', 'r', 's', 'š', 'z', 'ž', 't', 'u'],
['v', 'w', 'õ', 'ä', 'ö', 'ü', 'x', 'y']]
```

Kirjuta funktsioon `salakood`, mis võtab argumendiks kahemõõtmelise järjendi ja ühest sümbolist koosneva sõne. Funktsioon tagastab kahest arvust koosneva enniku, mis näitab tähe asukohta antud järjendis. Enniku esimene element on rea indeks, teine element on veeru indeks. Võib eeldada, et küsitud sümbol on alati olemas.

Funktsiooni kasutamise näide eelneva näite järjendiga:

```
>>> salakood(järjend, "c")
(0, 2)
```

Ülesanne A2 - Rallisõit (20 punkti)

Margus on suur rallifänn ja tal on oma lemmiksportlased, kelle WRC-sõite ta jälgib. Sõitjate tulemused kirjutab ta faili *wrc.txt* (kodeering UTF-8). Failis on igal real sõitja perenimi ja seejärel kooloniga eraldatult sõitja kogutud punktid.

Näide tekstifaili *wrc.txt* (Moodle'is leitav) võimalikust sisust:

```
Evans:70
Ogier:204
Paddon:55
Tänak:181
```

Koosta funktsioon `andmed_sõnastikku`, mis võtab argumendiks faili nime ja tagastab sõnastiku, kus iga kirje võtmeks on sõitja perenimi ning väärtuseks sõitja punktide arv.

Funktsiooni kasutamise näide eelneva tekstifailiga:

```
>>> andmed_sõnastikku("wrc.txt")
{'Ogier': 204, 'Evans': 70, 'Paddon': 55, 'Tänak': 181}
```

Koosta programm, mis esmalt loob funktsiooni `andmed_sõnastikku` abil faili *wrc.txt* põhjal sõnastiku. Seejärel küsib programm kasutajalt rallisõitja perenime ning kuvab ekraanile nende sõitjate perenimed, kellel on sellest sõitjast rohkem punkte (väljastatud sõitjate perenimede järjekord ei ole oluline). Näide programmi käivitamisest on järgnev (kasutaja sisend kaldkirjas):

```
Sisesta sõitja nimi: Evans
Rohkem punkte on järgmistel sõitjatel:
Ogier
Tänak
```

Ülesanne A3 - (10 punkti)

Koostada rekursiivne funktsioon `paaritu_korrutis`, mis võtab argumendiks ühe positiivse täisarvu ja tagastab kõikide positiivsete paaritute arvude korrutise, mis on väiksemad või võrdsed argumendiga. Näiteks kui argumendi väärtuseks on 5, siis korrutis on
1 * 3 * 5 = 15. Kui argumendi väärtuseks on 8, siis korrutis on 1 * 3 * 5 * 7 = 105.

Funktsiooni kasutamise näited:

```
>>> paaritu_korrutis(5)
15
>>> paaritu_korrutis(9)
945
>>> paaritu_korrutis(10)
945
```

VIII.“Programmeerimise alused II” statsionaarse kursuse arvutiülesanded

Programmeerimise alused II

Kontrolltöö 24.05.2017 kell 8.15-10.00

Kontrolltöö eest saab maksimaalselt 26 punkti. Minimaalselt on vaja saada 13 punkti.

NB! Olulised kohad programmis peavad olema kommenteeritud. Samuti lisada nimi kommentaarina.

Ülesanne 1. Maa tuleb täita lastega (6 punkti)

Tänapäeval on veel sündimata lapse kohta võimalik vanemate pikkuste põhjal teada saada lapse eeldatava pikkuse. Järgmiseid valemeid kasutavad lastearstid oma töös lapse eeldatava pikkuse leidmiseks:

$$Poeglastel\text{: } \frac{\text{isa pikkus (cm)} + (\text{ema pikkus (cm)} + 13\text{ cm})}{2}$$

$$Tütarlastel\text{: } \frac{\text{isa pikkus (cm)} + (\text{ema pikkus (cm)} - 13\text{ cm})}{2}$$

Esmalt kirjutada funktsioon lapse_pikkus, mis

1. võtab esimeseks argumendiks ema pikkuse sentimeetrites;
2. võtab teiseks argumendiks isa pikkuse sentimeetrites;
3. võtab kolmandaks argumendiks lapse soo (M või N);
4. tagastab vastavalt soole lapse eeldatava pikkuse sentimeetrites.

Ema ja isa pikkused ning lapse sugu on esitatud järjendis, kus elementideks on kolmeelemendilised ennikud. Enniku esimene element on ema pikkus, teine element isa pikkus ja kolmas element on lapse sugu.

Näide

[(163, 175, 'N'), (170, 180, 'M'), (153, 184, 'M'), (177, 165, 'N'), (166, 183, 'M')]

Teiseks kirjutada programm, mis

1. leiab laste eeldatavad pikkused sentimeetrites, kasutades funktsiooni lapse_pikkus,
2. väljastab laste pikkused,
3. väljastab laste keskmise pikkuse.

Ülesanne 2. Ei jagu viiega (6 punkti)

Koostada funktsioon summa_reas, mis

1. võtab argumendiks kahemõõtmelise tabeli t ja arvu i;
2. loendab kõik 5-ga mittejaguvad arvud tabeli t reas nr i.

Kirjutada programm, mis loendab seda funktsiooni kasutades kõik 5-ga mittejaguvate arvud kogu tabelis t.

Programmi testimiseks tuleb koostada funktsioon genereeri_tabel, mis

- võtab esimeseks argumendiks ridade arvu m tabelis ja võtab teiseks argumendiks veergude arvu n tabelis,
- tagastab kahemõõtmelise järjendi mõõtmetega $m \times n$, kus elemendid on juhuslikult genereeritud täisarvud lõigust 1 kuni 100.

Ülesanne 3. Mida külvad, seda lõikad (8 punkti)

Põllu külviplaan on esitatud failis tabelina, kus veerg tähistab põllu vagu. Põllu suurus ei ole fikseeritud. Külvatavad seemned on märgitud failis vastava IDga, mis on eraldatud teineteisest tühikute abil. Agronoom käskis seemned külvata nii, et vao peal on ühe sordi taimed ja sama sordi taimed ei tohi kasvada kõrvuti olevate vagude peal.

Koostada funktsioon failist_jarjendisse, mis

- võtab argumendiks failinime,
- tagastab kahemõõtmelise järjendi põllu külviplaani kohta.

Koostada funktsioon kontroll_vaod, mis

- võtab argumendiks kahemõõtmelise külviplaani tabeli,
- tagastab vastavalt True või False, kas külviplaani vao peal sama sordiseemned,

Koostada funktsioon kontroll_seemned,mis

- võtab argumendiks kahemõõtmelise külviplaani tabeli,
- tagastab vastavalt Truevõi False, kas põllul on erineva sordi taime seemed kõrvuti vagudel.

Koostada programm, mis

- küsib kasutaja käest failinime,
- loeb failist andmed järjendisse kasutades funktsiooni failist_jarjendisse,
- kontrollib, kas failist loetud külviplaan vastab nõuetele
    - kui külviplaan vastav nõetele, siis väljastab ekraanile SOBIB,
    - kui külviplaan ei vasta nõutele, siis väljastab ekraanile EI SOBI.

Näide programmi tööst:

Faili *pold1.txt* sisu
K1 S3 M1 K3
K1 S3 M1 K3
K1 S3 M1 K3
K1 S3 M1 K3
K1 S3 M1 K3
S2 S3 M1 K3
K1 S3 M1 K3
K1 S3 M1 K3

Faili *pold2.txt* sisu
K1 S3 M1 M1
K1 S3 M1 M1
K1 S3 M1 M1
K1 S3 M1 M1
K1 S3 M1 M1
K1 S3 M1 M1

Faili *pold3.txt* sisu
K1 S3 M1 M0
K1 S3 M1 M0
K1 S3 M1 M0
K1 S3 M1 M0

```
>>> %Run KT_3_Sade.py

  Sisestage failinimi: pold1.txt
  EI SOBI

>>> %Run KT_3_Sade.py

  Sisestage failinimi: pold2.txt
  EI SOBI

>>> %Run KT_3_Sade.py

  Sisestage failinimi: pold3.txt
  SOBIB
```

Ülesanne 4. Paaritu korrutis (6 punkti)

Koostada rekursiivne funktsioon paaritu_korrutis, mis võtab argumendiks ühe positiivse täisarvu ja tagastab kõikide positiivsete paaritute arvude korrutise, mis on väiksemad või võrdsed argumendiga.

IX. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Eva-Maria Pedosk**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Programmeerimise algkursuste ülesannete kategoriseerimine Bloomi taksonoomia alusel**,

   mille juhendaja on **Eno Tõnisson ja Marina Lepp**,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

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

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

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

Eva-Maria Pedosk

10.05.2019