Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Valentin Kragelj

Poučevanje algoritmov v srednji šoli

magistrsko delo

magistrski program druge stopnje
pedagoško računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana, 2018

Zahvala

Iskreno se zahvaljujem mentorju dr. Andreju Brodniku za vso strokovno pomoč, nasvete in usmerjanje pri izdelavi magistrskega dela. Zahvaljujem se prijateljem za nesebično pomoč pri pisanju ter širši družini za podporo v času študija.

Valentin Kragelj, 2018

Vsem puncam tega sveta.

"Behind every great man is a woman rolling her eyes."

— Jim Carrey

Kazalo

Povzetek

Naslov: Poučevanje algoritmov v srednji šoli

Računalništvo in informatika velja za eno najhitreje rastočih področij na svetu. To ima za posledico dejstvo, da postaja enako pomemben tudi način poučevanja tega področja na šolah. V tujini se s tem ukvarjajo že več let, v Sloveniji pa smo šele pred kratkim začeli pogumno korakati po njihovih stopinjah - kot pritiče navadi - nekaj korakov za njimi. Motivacija za magistrsko nalogo je tako izboljšati poučevanje informatike v slovenskih gimnazijah. Za cilj smo si zadali izdelavo in ovrednotenje učnih priprav na temo poučevanja algoritmov za predmet informatika, ki se izvaja v prvem letniku slovenskih splošnih, klasičnih in strokovnih gimnazij. V okviru magistrske naloge smo tako za vsako učno temo raziskali sodobne in učinkovite metode poučevanja le-te ter izdelali šolsko učno pripravo. Nato smo za vsako učno pripravo pripravili in izvedli delavnico. Namen le-teh je merjenje učinkovitosti učnih pristopov iz vidika pisnega preverjanja znanja. S pomočjo ovrednotenja pridobljenih rezultatov smo učne priprave popravili in dopolnili. Za konec smo učne priprave naložili na spletno učilnico projekta NAPOJ, kjer so na voljo vsem učiteljem informatike slovenskih srednjih šol.

Ključne besede

računalništvo, informatika, algoritmi, poučevanje algoritmov

Abstract

Title: Teaching algorithms in high school

Computer and information science is one of the fastest growing areas in the world. This results in the fact that the way of teaching this field in schools is equally important. They have been doing this abroad for many years now, but in Slovenia we have only recently started boldly marching on their footsteps - as is tradition - a few steps behind them. Motivation for the master's thesis is thus to improve the teaching of informatics in Slovenian gymnasiums. The aim was to create and evaluate learning materials on the subject of teaching algorithms for the subject of informatics, which is taught in the first year of Slovenian general, classical and professional gymnasiums. In the framework of the master's thesis, for each subject, we studied modern and effective methods of teaching it and produced a school learning preparation. We then prepared and carried out a workshop for each educational preparation. The purpose is to measure the effectiveness of learning approaches from the point of view of written examination of knowledge. With the help of evaluation of the obtained results, the training preparations were additionally improved. In the end, we uploaded learning preparations to the online classroom of the NAPOJ project, where they are available to all teachers of information science in Slovenian secondary schools.

Keywords

computer science, informatics, algorithms, teaching algorithms

Poglavje 1

Uvod

Zaradi hitrega razvoja področja računalništvo in informatika se v svetu izobraževanja postavlja v ospredje problem poučevanja omenjenega področja. Za začetek se postavlja že vprašanje pravilne uporabe terminologije – ali sta pojma računalništvo in informatika pravzaprav sopomenki? Če nista, v čem se področji razlikujeta? Ali sploh razumemo uporabnost teh področij? Kot je omenil Hromkovič [1], veliko ljudi razume računalništvo in informatiko kot sposobnost uporabe računalnika: za brskanje po svetovnem spletu, za uporabo različne programske opreme, kot je program Microsoft Word, in podobno. Pri tem ne pomaga dejstvo, da se v šolah poučuje predvsem uporabo informacijske tehnologije [1]. Še več, v Sloveniji v splošnem velja mnenje, da si kot študent računalništva in informatike uporaben le za popravljanje računalnikov, ko gre kaj po zlu. Skratka, izobraževanje se na tem področju ne razvija v pravo smer.

1.1 Motivacija

Področje računalništva in informatike (RIN) se bo v prihodnosti seveda razvijalo še naprej. Omenimo le eno od mnogih raziskav [2] (več raziskav omenimo v poglavju 2.5), ki pravi, da se bo v celotnem letu 2018 v ZDA povečala potreba po računalniških poklicih za kar 34 %. Čeprav ne bomo vsi postali računalničarji, je znanje računalništva še vedno velika prednost, saj se informatika kot znanost vedno bolj prepleta z ostalimi znanostmi, kot so biologija, kemija in fizika, iz česar tudi nastajajo nove znanosti, kot je bioinformatika. V biologiji se informatiko uporablja za analizo genomov in proteinov, v medicini za shranjevanje zdravstvene zgodovine pacientov, v vremenoslovju za simulacije tornadov, v muzikologiji za razvoj novih glasbenih zvrsti itd. Opazimo, da je torej znanje računalništva in informatike pomembno, zaradi česar mora biti tudi njeno poučevanje

pomembno sodobno vprašanje v izobraževanju. Motivacija magistrske naloge je tako izboljšati stanje poučevanja informatike v Slovenji, pri čemer smo se omejili na predmet informatika, ki se poučuje na slovenskih splošnih, klasičnih in strokovnih gimnazijah.

1.2 Pregled rešitve

Odločili smo se, da bomo stanje poučevanja informatike v Sloveniji izboljšali tako, da bomo v okviru projekta NAPOJ [3] izdelali učne priprave za sklop poučevanja algoritmov, opisanih v e-učbeniku Računalništvo in informatika 1 [4], ter jih ovrednotili. Poleg tega bomo v projektu TOMO [5] izdelali programske naloge za pripadajoče učne priprave.

V empiričnem delu bomo na skupini učencev izvedli delavnice, ki se navezujejo na izdelane učne priprave. Na vsaki delavnici bomo preverili znanje učencev na podlagi pisnega preverjanja znanja pred delavnico in po njej. Na koncu bomo s kvantitativnim pristopom iz rezultatov pisnih preverjanj merili učinkovitost v učnih pripravah uporabljenih učnih pristopov.

Učne priprave in programske naloge bomo nato na spletni učilnici projekta NAPOJ ponudili vsem učiteljem informatike v Sloveniji.

1.3 Struktura dela

V poglavju 2 se bomo najprej posvetili področju računalništva in informatike – opisu terminologije ter opredelitvi pomembnosti tega področja. Temu bo sledil pregled stanja poučevanja RIN v Sloveniji ter v tujini. Za konec poglavja bomo predstavili prenovo poučevanja RIN v Slovenji. Poglavje 3 bo namenjeno metodologiji in bo opisovalo reševanje problema pisanja učnih priprav. V zadnjem, 4., poglavju opišemo vsebino napisanih učnih priprav, potek delavnic ter njihovo ovrednotenje.

Poglavje 2

Pregled obstoječega stanja

Ker poučevanje algoritmov oziroma algoritmike spada na področje računalništva in informatike, se bomo najprej osredotočili na širšo sliko.

2.1 Kaj je računalništvo in informatika

RIN je področje, ki v zadnjih letih hitro napreduje, zaradi česar se na področju uporablja različne izraze, ki imajo enak ali pa zelo podoben pomen. Izraz informatika ima denimo različne opredelitve glede na to, kje je uporabljena. V Evropi lahko uporabimo kar besedo računalništvo (ang. *computer science*), medtem ko se jo v Združenih Državah Amerike povezuje z uporabnim računanjem (ang. *applied computing* ali zgolj *computing*) [6]. G. A. Marco trdi [7], da informatika oziroma informacijska znanost kot disciplina sploh ne obstaja, k zmedi na področju pa prispeva tudi splošno sprejeto mnenje ljudi, ki povezujejo RIN s sposobnostjo uporabe računalnika v smislu uporabe interneta, pošiljanja elektronskih sporočil ter uporabe širše uporabljenih programov, kot so Microsoft Word, Microsoft Excel ipd. [1]. Kot bomo opisali v nadaljevanju, je RIN vse prej kot le to.

Zgodovinsko gledano sta se tako informatika kot tudi računalništvo kot pojma pojavila leta 1956 [8, 9]. Čeprav se slednji nanaša na besedo računalnik, se je takratna znanost bolj ukvarjala z računanjem kot pa s samim računalnikom. Zaradi tega so kasneje nastale skovanke računska znanost, nato pa še podatkovna znanost in informatika v sodobnem pomenu besede [4].

Poleg tega, da izraz računalništvo povzroča rahlo zmedo z besedami, ima tudi veliko definicij. Če jih naštejemo le nekaj:

> Računalništvo je znanost o algoritemski obdelavi, predstavitvi, shranjevanju in prenosu informacij. [1]

> Računalništvo je študij izvedljivosti, strukture, izražanja in mehanizacije metodičnih procesov (ali algoritmov), ki temeljijo na pridobivanju, predstavljanju, obdelavi, shranjevanju, sporočanju in dostopu do informacij, ne glede na to, ali so takšni podatki kodirani v bitih in bajtih, v računalniškem pomnilniku ali zapisani v genih in strukturah beljakovin v človeški celici. [10]

> Računalništvo je preučevanje načel, aplikacij in tehnologij računalništva in računalnikov. Vključuje preučevanje podatkovnih in podatkovnih struktur ter algoritmov za obdelavo teh struktur; principov računalniške arhitekture – strojne in programske opreme; reševanja problemov in oblikovanja metodologij; računalniško povezanih tem, kot so numerična analiza, operativne raziskave in umetna inteligenca; ter oblikovanja, strukture in prevajanja jezikov. [11]

> Disciplina računalništva je sistematično preučevanje algoritmičnih procesov, ki opisujejo in pretvarjajo informacije: njihovo teorijo, analizo, oblikovanje, učinkovitost, izvajanje in uporabo. Temeljno vprašanje, na katerem temelji vse računalništvo, je: Kaj je mogoče (učinkovito) avtomatizirati? [12]

> Računalništvo in informatika je proučevanje računalnikov in algoritmičnih procesov, vključno z njihovimi temelji, njihovo strojno opremo in sestavo programske opreme, njihovo uporabo in njihovim vplivom na družbo. [13]

Opazimo, da je vsem definicijam skupno to, da definirajo računalništvo kot zapis in obdelavo podatkov oziroma informacij z uporabo računalnika.

Informatika ima med drugimi naslednje definicije:

> Informatika je preučevanje strukture, vedenja in interakcij naravnih in računalniških sistemov. [14]

> Izraz informatika na splošno opisuje študij, oblikovanje in razvoj informacijske tehnologije za dobro ljudi, organizacij in družbe. [15]

> Informatika je preučevanje in uporaba informacijske tehnologije za umetnost, znanost in poklice ter za njeno uporabo v organizacijah in v družbi na splošno. [16]

Opazimo dokaj širok in abstrakten opis pojma. S čim vse se informatika ukvarja, bomo tako bolje razumeli, če jo razdelimo na podpodročja. Eno od možnih delitev predstavijo v svetovnem računalniškem združenju ACM (*Advancing Computing as a Science and Profession*), kjer v svojem kurikulumu razdelijo Informatiko na naslednja podpodročja [4]:

1. algoritmi in zahtevnost (*algorithms and complexity*) – osnova informatike;
2. arhitektura in organizacija računalniških sistemov (*architecture and organization*) – osnovni deli in funkcionalnosti računalniškega sistema;
3. računska znanost (*computational science*) – ideje in tehnike reševanja problemov;
4. diskretne strukture (*discrete structures*) – matematični koncepti, kot so teorija množic in grafov;
5. grafika in vizualizacija (*graphics and visualization*);
6. vmesnik človek-stroj (*human-computer interaction*) – kako razviti učinkovit računalniški vmesnik;
7. informacijska zagotovila in varnost (*information assurance and security*) – zaščita in varovanje informacij in informacijskih sistemov;
8. upravljanje informacij (*information management*) – predstavitev, oblikovanje in organizacija informacij;
9. inteligentni sistemi (*ingelligent systems*) – samoučenje računalniških sistemov;
10. omrežja in komunikacije (*networking and communications*) – računalniška omrežja in komunikacija med rečunalniki;
11. operacijski sistemi (*operating systems*) – kaj je in kako deluje operacijski sistem;
12. razvoj v specifičnih okoljih (*platform-based development*) – programiranje za različne naprave;
13. vzporedno in porazdeljeno računalništvo (*parallel and distributed computing*);
14. programski jeziki (*programming languages*);
15. osnove razvoja programske opreme (*software development fundamentals*) – osnovni koncepti programiranja, podatkovnih struktur in algoritmov;
16. programsko inženirstvo (*software engineering*);
17. osnove sistemov (*systems fundamentals*) – sloji računalniškega sistema, od sestavnih delov strojne opreme do storitev operacijskih sistemov;
18. družbena vprašanja in poklicna praksa (*social issues and professional practice*) – zgodovina informatike, vpliv nje na družbo, problem zasebnosti in intelektualne lastnine.

V Sloveniji smo pred letom 1990 delili RIN na tri področja – strojna oprema, programska oprema in informatika, ki so bila izrecno ločena med seboj. Čeprav se v sodobnem času v svetu s pojmoma *Computer science* ter *Informatics* označuje vsa omenjena področja skupaj, v Sloveniji še nismo prišli do skupnega imena. Načeloma velja, da je pojem RIN enakovreden angleškemu pojmu *Computer science*, za katerega pa še nimamo ustreznega prevedenega izraza [4].

Zaradi tega so v tej magistrski nalogi pojmi kot so informatika, računalništvo, računska znanost, uporabno računanje, informacijska znanost in še kaj, uporabljeni izmenjaje, največkrat pa bomo zaradi konsistentnosti uporabili kratico RIN.

2.2 Zakaj poučevati računalništvo in informatiko?

Zaradi široke definicije RIN in posledično njene uporabnosti, je ta v današnjem svetu vseprisotna ter vključuje vse vidike našega življenja. Raziskava Urada za statistiko dela v Združenih Državah Amerike kaže, da se bo leta 2018 povečala potreba po računalniških poklicih za 34 % [2]. Da je učenje računalniškega mišljenja pomembno, med drugim trdi tudi Jeanette Wing, do julija 2017 podpredsednica podjetja Microsoft Research, sedaj pa direktorica Inštituta za podatkovne vede na univerzi Columbia, ki pravi, da tovrstno učenje pomaga ljudem pri abstraktnem razmišljanju ter razdeljevanju problemov na podprobleme. Pri učenju RIN, pravi, mora biti poudarek na učenju reševanja problemov [2].

Poleg tega se informatika kot znanost vedno bolj prepleta z ostalimi znanostmi, kot so biologija, kemija, fizika itd., iz česar nato nastajajo nove znanosti, kot so bioinformatika, ki vključuje integracijo računalnikov, programskih orodij in baz podatkov pri reševanju bioloških vprašanj, predvsem v genomiki (analiza genomov) in proteomiki (analiza proteinov) [17].

Že ob pogledu na naš vsakdanjik lahko opazimo, na kakšne načine nam informatika oziroma računalništvo pomagata pri vsakdanjih opravilih. Da se zjutraj zbudimo, uporabimo budilko – bodisi na mobilni napravi, bodisi na digitalni uri na nočni omarici. Za umivanje zob lahko uporabimo električno ščetko. V kuhinji si pripravimo obrok, po potrebi z uporabo opekača, mikrovalovne pečice, ali katere druge naprave. Ob tem se seznanimo z jutranjimi novicami prek radia, televizije ali mobitela. Za transport do službe uporabimo avtomobil, ki vsebuje raznorazne elektronske dele. V službi v takšni ali drugačni obliki uporabljamo raznorazne pametne naprave, s katerimi smo pri delu hitrejši in uspešnejši. Takšnih in podobnih situacij je seveda še veliko in vse se navezujejo na področje RIN.

Vprašanje, ki se na tem mestu običajno pojavi, je, zakaj potrebujemo znanje RIN,

če želimo uporabljati uro, mikrovalovno pečico, avto ipd.? Hromkovič [1] trdi, da učenje računalništva pomeni učenje jezika komuniciranja s tehničnimi sistemi oziroma učenje, kakšno dejavnost želimo imeti od njega. Ker stroji (še) nimajo inteligence, morajo naša navodila zanje biti nedvoumna, brez možnosti pojavitve napak. Razvoj te veščine prispeva k obvladovanju naravnega jezika, tako da jih motivira, da pravilno razmišljajo o tem, kako najbolje izraziti to, kar želijo sporočiti. Poleg tega Hromkovič našteje še 5 razlogov [1], zakaj je znanje osnov računalništva uporabno. Ti razlogi so filozofska globina, uporabnost, življenja doba znanja, interdisciplinarna usmeritev in način mišljenja.

Filozofska globina

Teoretično računalništvo raziskuje znanje in razvija koncepte in pojme, ki vplivajo na samo jedro znanosti. Poleg tega daje odgovore na vprašanja, kot so:

- Ali obstajajo problemi, ki niso algoritmično rešljivi? Če so, kje leži meja med avtomatsko rešljivostjo in avtomatsko nerešljivostjo?
- Ali so nedeterministični in naključni procesi zmožni doseči to, česar deterministični niso? Ali sta nedeterminističnost in naključnost boljši od determinizma?
- Kako definirati težavnost problema?
- Kje so meje »praktične« algoritmične rešljivosti?
- Kaj je matematični dokaz? Ali je težje poiskati matematični dokaz algoritmično ali preveriti pravilnost danega dokaza?
- Kaj je naključni objekt?

Zgornjih vprašanj ni možno pravilno oblikovati brez formalnih konceptov algoritmov. S tem je teoretično računalništvo razširilo jezik znanosti, s čimer je prispevalo k njenemu razvoju.

Uporabnost

S teoretičnim računalništvom lahko rešimo veliko problemov, ki jih sicer ne bi bili zmožni:

- Ali je možno prepričati nekoga, da poznamo geslo, ne da bi razkrili en sam njegov bit?
- Ali je možno izvedeti, kateri od dveh ljudi je starejši, ne da bi oba razkrila starost drug drugemu?
- Ali lahko skoraj z zagotovostjo preverite pravilnost več tisoč strani dolgega matematičnega dokaza samo s pogledom na nekaj naključno izbranih bitov?

S tem se računalništvo izkaže kot uporabno in tudi zanimivo za študij.

Življenjska doba znanja

Kot bomo omenili v naslednjem poglavju, zgolj učenje uporabe trenutno razširjene programske opreme ni priporočljivo, saj s tem iz učencev delamo uporabnike in ne inovatorje. Razvoj na tem področju se odvija z izredno hitrostjo – polovica obstoječih informacij o konkretni programski in strojni opremi je zastarela že po petih letih, zaradi česar takšno izobraževanje ne zagotavlja ustreznih zmožnosti za zaposlitev. Učenje konceptov in metodologij teoretičnega računalništva reši ta problem, saj je to znanje uporabno več desetletij in bo služilo učencem na dolgi rok.

Interdisciplinarna usmeritev

Omenili smo že, da z informatiko nastajajo nove znanosti, kot je bioinformatika. Lahko pa sodeluje tudi na drugih področjih, kot so:

- poslovna znanost – uporaba računalniških sistemov za avtomatizacijo procesov, s čimer se delo pohitri ter postane cenejše [18];
- medicina – podatkovne baze za shranjevanje zdravstvene zgodovine pacientov ter raznorazne naprave za diagnozo in zdravljenje pacientov [18];
- vremenoslovje – simulacija tornadov [19];
- astronomija – simulacija eksplozij supernove [20];
- muzikologija – razvoj novih glasbenih zvrsti;
- avtomatsko prepoznavanje govora [1];
- raziskovanje vesolja [1];
- itd.

Način mišljenja

Matematiki v izobraževanju pripisujejo posebno vlogo pri razvoju, bogatenju ter oblikovanju načina mišljenja, tj. posebno vlogo pri prispevanju k splošnemu razvoju osebnosti učenca. Teoretično računalništvo prav tako spodbuja ustvarjanje in analizo matematičnih modelov realnih sistemov ter iskanje konceptov in metod za reševanje konkretnih problemov. Ravno razumevanje, katere značilnosti realnega sistema so zajete v nekem modelu in katere so zgolj približane ali celo zanemarjene, je glavna predpostavka za uspeh v znanosti in inženirstvu. Zato matematično računalništvo spodbuja poučevanje matematičnih konceptov in modelov v povezavi z resničnimi problemi. S tem se lahko učenec nauči, kako združiti teoretično znanje s praktičnimi izkušnjami in s tem razvija način mišljenja, ki je dovolj sposoben za reševanje nadaljnjih kompleksnejših realnih problemov.

Na pomembnost znanja RIN kaže tudi gospodarstvo. Na VSG GZS (Vrh slovenskega gospodarstva) so sprejeli DigitAgenda 2016 [21], kjer je zapisanih 30 priporočil, kako bi lahko z digitalizacijo povečali povprečno produktivnost do leta 2015 za 3 %, pri čemer bi se ustvarilo 10 000 novih delovnih mest. Poroča tudi, da bo do leta 2020 90 % delovnih mest potrebovalo znanje IKT. Podobno analizo so opravili v Angliji leta 2016, kjer so ugotovili, da je vzrok za izgubo 46 milijard funtov na leto prav zaradi pomanjkanja omenjenega znanja.

Če se ozremo na Slovenijo, je decembra 2017 potekal posvet o poučevanju računalništva in informatike na Slovenski akademiji znanosti in umetnosti, kjer so izpostavili, da sposobnost računalniškega mišljenja prinaša veliko prednosti:

1. v sodobnem svetu je ta sposobnost ključna za učinkovito spoprijemanje s sodobnimi poklicnimi izzivi;
2. urjenje računalniškega mišljenja je urjenje v strategijah reševanja problemov;
3. urjenje v računalniškem mišljenju lahko pomeni tudi dolgoročno opolnomočenje učencev za razvijanje vztrajnosti pri soočanju z neuspehom in posledično večanje psihološke odpornosti.

Skratka, RIN je zelo široko področje, brez katerega si sodobnega sveta ne moremo predstavljati. Učinkovito poučevanje na tem področju je zato ključno za dolgoročni razvoj učencev. V naslednjem poglavju si torej poglejmo, kako učinkovito je poučevanje, tako v Sloveniji, kot tudi v tujini.

2.3 Računalništvo in informatika v izobraževanju v Sloveniji

Poglavje je razdeljeno na opis RIN v slovenskem šolskem izobraževanju, opis in primerjavo današnjega slovenskega kurikuluma s tujimi ter omembe raznih raziskav, ki kažejo na učinkovitost in znanje RIN učencev doma in po svetu.

2.3.1 Kratka zgodovina RIN v slovenskem šolstvu

Leta 1971 se je v Sloveniji začelo poučevanje računalništva in informatike s projektom Uvajanje računalniške pismenosti v srednje šole. Takrat so na izbranih šolah poučevali računalniški strokovnjaki, hkrati pa je potekal tudi tečaj za bodoče učitelje računalništva. V samo štirih letih se je projekt razširil z 20 na 65 slovenskih srednjih šol, pri čemer je število učencev naraslo z 200 na 2500, kar kaže na izjemen uspeh samega projekta. Učni

načrt je predvideval 60 ur pouka, pri čemer je bila prva polovica namenjena teoretičnim snovi (računalnik, algoritem, bitni zapis podatkov, zgradba računalnika, programski jeziki itd.), druga polovica pa programiranju v programskem jeziku Fortran [2, 5].

V šolskem letu 1980/81 so bile z vpeljavo usmerjenega izobraževanja ustanovljene srednje računalniške šole s preko 1000 urami pouka strokovnih računalniških predmetov. Leta 1983 je bila določena standardna strojna in programska oprema na srednjih šolah – računalniki Spectrum, Commodore 64 ter kasneje Partner. V naslednjih letih je naraslo število raznih računalniških tekmovanj, poletnih šol računalništva ter projektov, kot so RAČEK (Računalniška eksplozija), PETRA, Učitelji inštruktorji, Ro (Računalniško opismenjevanje) ter Informatizacija predmetov [22, 23].

V šolskem letu 1990/91 je z ukinitvijo usmerjenega izobraževanja RIN postal obvezen enoletni predmet na vseh štiriletnih srednjih šolah. Predmet je zajemal osnove RIN, baze podatkov, programiranje in urejanje besedil. Ker predmet ni bil uvrščen med maturitetne predmete, je v naslednjih štirih letih stagniral, dokler leta 1995 obvezni del učnega načrta ni doživel prenove, pri kateri je bil večji poudarek na urejanju besedil. Zadnjo spremembo je učni načrt doživel s prenovo v šolskem letu 1997/98 [22, 23].

Ko so v slovenskih osnovnih šolah uvedli devetletko, je RIN postal izbirni predmet na voljo učencem v zadnjem triletju, torej 7., 8. in 9. razredu. Informatika v srednji šoli predstavlja nadgradnjo RIN iz osnovne šole. Leta 2002 je bila uvrščena med izbirne maturitetne predmete, prvič pa se je pojavila na maturi leta 2007. V šolskem letu 2014/15 se je v 4. razredu osnovne šole začel izvajati neobvezni izbirni predmet računalništvo, ki je sicer na voljo učencem 4., 5., in 6. razreda [22, 23].

2.3.2 RIN danes v slovenskih osnovnih šolah

Kot smo omenili, imajo od šolskega leta 2014/15 učenci 4., 5., in 6. razreda osnovne šole na voljo izbirni predmet računalništvo. Predmet lahko učenci obiskujejo eno, dve ali tri leta. Vsako leto pripada predmetu 35 šolskih ur (1 ura tedensko) – skupaj torej največ 105 ur v treh letih. Pri predmetu učenci spoznajo temeljne računalniške koncepte in procese. Učni načrt predmeta ga opredeljuje kot:

»*... Predmet ne temelji na spoznavanju dela s posameznimi programi, ampak učence seznanja s temeljnimi računalniškimi koncepti in procesi. Učenci se pri računalništvu seznanjajo s tehnikami in metodami reševanja problemov in razvijajo algoritmičen način mišljenja, spoznavajo omejitve računalnikov in njihov vpliv na družbo ... S spoznavanjem računalniških konceptov in razvijanjem postopkovnega načina mišljenja učenci pridobivajo znanja, spretnosti in veščine, ki so veliko bolj trajni kot hitro razvijajoče se tehnologije.*« [24]

Predmet računalništvo se sicer deli na 5 vsebin oziroma sklopov, ki so obvezni: algo-

ritmi, programi, podatki, reševanje problemov ter komunikacije in storitve, s katerimi se pokrije vsa ključna področja RIN (to so računalniški sistemi, omrežja in internet, podatki, algoritmi in programiranje, vpliv računalništva na družbo ter računalniško mišljenje [23]). Vsak sklop se lahko poučuje v poljubnem razredu, pri čemer so učenci različnih razredov lahko združeni. Največje dovoljeno število učencev v skupini je 28.

V zadnjem triletju lahko učenci izbirajo med predmeti urejanje besedil, računalniška omrežja ter multimedija, pri čemer v 7. razredu začnejo z urejanjem besedil, v 8. razredu nadaljujejo z računalniškimi omrežji, multimedija pa je namenjena učencem zadnjega razreda devetletke. Raziskave kažejo, da je računalništvo med najbolj priljubljenimi izbirnimi predmeti, saj se za njega odloči povprečno 75 % učencev [22]. Vsi trije predmeti se delijo na naslednje sklope:

- osnove informatike in računalništva;
- obdelava podatkov in komuniciranje z uporabo informacijske tehnologije;
- programiranje, ki spada pod dodatno vsebino.

V prvem sklopu se učenci učijo o tem, kaj je informacija, kaj podatek, in kako ju predstaviti, ter usvajajo osnove strojne in programske opreme. V drugem sklopu imajo opravka s računalniškimi programi za urejanje besedil in programi za grafiko. Pri računalniških omrežjih v ta sklop spada še izdelava spletne strani. Sklop programiranje je povsod dodan kot dodatna vsebina, pri katerem se učenci naučijo napisati preprost algoritem ter izdelati preprost računalniški program [25].

Predmeti urejanje besedil, računalniška omrežja ter multimedija podobno kot izbirni predmet računalništvo iz 2. triletja prav tako pokrijejo vsa ključna področja računalništva z izjemo vpliva računalništva na družbo ter algoritmov in programiranja. Slednje področje namreč spada pod dodatno vsebino, kar pomeni, da se vsak učitelj samostojno odloči, ali ga bo poučeval ali ne [23].

2.3.3 RIN danes v slovenskih srednjih šolah

V izobraževalnem programu gimnazije iz leta 2008 je predmet informatika zapisan kot obvezen predmet, ki obsega 70 ur in se izvaja na splošnih in klasičnih gimnazijah. Na strokovnih gimnazijah informatika traja 105 šolskih ur – 70 ur v prvem letniku ter 35 ur v drugem letniku. Učni načrt je sestavljen za 280 ur, torej za maturitetni predmet, pri čemer ni eksplicitno napisanih učnih ciljev za obveznih 70 ur (za klasične in splošne gimnazije) oziroma 105 ur (za strokovne gimnazije) [23]. Zaradi slednjega se učitelji informatike samostojno odločijo, katere vsebine bodo poučevali v okviru teh ur. Posledično dosežejo vse cilje učnega načrta le dijaki, ki si izberejo informatiko za maturitetni predmet. Takšnih

dijakov, ki so si na maturi izbrali kot izbirni predmet informatiko, je bilo v šolskem letu 2015/16 le 5,34 % od vseh gimnazijcev [23].

Cilji in vsebine predmeta so razporejeni na dve ravni – splošna znanja in posebna znanja. Pri splošnih znanjih dijaki razvijajo temelje digitalne kompetence, potrebne za uporabo digitalne tehnologije pri razvijanju lastnega znanja in za njegovo predstavitev. Vsebuje tri tematske sklope:

1. Osnove informatike, ki obsega naslednje vsebine:
   - temeljne pojme – kaj je podatek, informacija, računalnik, informacijski sistem, računalniška pismenost itd.;
   - družbene vidike informatike – razumejo vlogo informacije v sodobni družbi ter razložijo in s primeri ovrednotijo pomen varovanja, zaščite podatkov in zasebnosti;
   - komuniciranje – opredelijo komuniciranje in njen pomen, poznajo sestavine komuniciranja in jih opredelijo, opredelijo učinkovitost in uspešnost komuniciranja ipd.
2. Digitalna tehnologija, ki obsega naslednje vsebine:
   - namen, vloga in pomen digitalne tehnologije – poznajo digitalne tehnologije, uporaba, zgodovina računalništva itd.;
   - zgradba in delovanje računalnika – poznajo Von Neumannov model računalnika in razložijo njegovo delovanje;
   - strojna oprema računalnika – sestavijo osebni računalnik in opredelijo temeljne tehnične lastnosti posameznih enot računalnika;
   - programska oprema računalnika – poznajo vrste programske opreme in njihove vloge, poznajo vrste zaščite programov, poznajo oblike računalniških vsiljivcev;
   - računalniška omrežja – razložijo pomen računalniškega omrežja, poznajo načine organiziranja omrežij, opredelijo internet, poznajo njegove pomembnejše storitve in opredelijo njihovo funkcijo.
3. Predstavitev informacij, kjer dijaki spoznajo osnove in pomen zapisa podatkov, pomen standardizacije, načine predstavitev informacij, opredelijo zgoščevanje podatkov itd.

Pri posebnih znanjih dijaki spiralno nadgradijo znanje, veščine in spretnosti, ki so jih pridobili iz splošnih znanj. Vsebuje dva tematska sklopa:

1. Predstavitev informacij, ki obsega naslednje vsebine:

    - pisna predstavitev informacije,
    - slikovna predstavitev informacije,
    - zvočna predstavitev informacije,
    - predstavitev informacije z gibljivo sliko,
    - računalniške prosojnice,
    - predstavitev informacije na svetovnem spletu.

2. Odelava podatkov, ki obsega naslednje vsebine:

    - računalniška obdelava podatkov – poznajo pomen obdelave podatkov in programiranja;
    - algoritem – opredelijo algoritem, poznajo temeljne gradnike algoritma, zapišejo algoritem na več načinov, analizirajo algoritem;
    - programski jezik – opredelijo programski jezik, poznajo temeljne gradnike in poznajo njihove funkcije;
    - programiranje – napišejo računalniški program, opišejo njegov namen, analizirajo rezultate programa;
    - podatkovna baza – poznajo sestavine tabele, pomen ključa, osnovne tipe podatkov ipd.;
    - preglednica – poznajo osnove dela s preglednicami in temeljne oblike grafikonov;
    - tehnologija znanja – poznajo vrste tehnologij znanja, razložijo faze odločitvenega procesa in pomen simulacije pri reševanju problemov.

S temi sklopi predmet pokrije temeljna znanja računalništva in informatike: računalniški sistemi, omrežja in internet, podatki, algoritmi in programiranje, vpliv računalništva na družbo ter računalniško mišljenje [23]. Kljub temu je treba poudariti že omenjeno slabost kurikuluma, da ni izrecne razdelitve med obveznimi in izbirnimi vsebinami oziroma kaj naj bi se učilo v prvem letniku gimnazije, kjer je informatika obvezen predmet, in kaj v ostalih letnikih, kjer je predmet izbiren. Zaradi tega se poučevanje predmeta od šole do šole lahko zelo spreminja. Opazimo tudi, da se predmet preveč nagiba k sklopoma informacijska tehnologija in digitalna pismenost, premalo pa pokriva področje algoritmov in programiranja [26].

Tabela 2.1: Uporabljena standardizacija.

Starost učencev v letih	**Razred**
6–7	1. razred
7–8	2. razred
8–9	3. razred
9–10	4. razred
10–11	5. razred
11–12	6. razred
12–13	7. razred
13–14	8. razred
14–15	9. razred
15–16	10. razred (1. letnik)
16–17	11. razred (2. letnik)
17–18	12. razred (3. letnik)
18–19	13. razred (4. letnik)

2.4 Računalništvo in informatika v izobraževanju v tujini

Tuje države vpeljujejo vsebine RIN v obvezne predmete skozi celotno izobraževanje. Ker bi bil opisovanje kurikulumov RIN vseh držav prevelik zalogaj, smo se odločili za prikaz primerjave poučevanja RIN v izbranih pomembnejših državah v obliki tabele 2.2. Ta za izbrane države prikazuje, v katerih razredih se učenci seznanijo z vsebinami RIN ter katere vsebine vključuje kurikulum [26, 27]. Ker šolski sistemi v državah, ki so v tabeli, niso enaki (npr. učenci višjih šol v Italiji so enako stari kot učenci srednjih šol v Sloveniji, zaradi česar bi zapis »1.–4. letnik« pomenil učence starosti 12–15 let v Italiji, v Sloveniji pa 15–19 let), smo se odločili za standardizacijo rezultatov, kot je prikazano v tabeli 2.1.

Tabela 2.2: Primerjava kurikulumov in vpeljav predmeta RIN po svetu.

Država	**Obvezen predmet RIN**	**Integrirano v drugem obveznem predmetu**	**Obvezno izbirni predmet RIN ali integriran v drug obvezno izbirni predmet**	**Prosto izbirni predmet RIN ali integriran v drugem prosto izbirnem predmetu**	**kurikulum**
Avstrija	9.			10.–12.	uvod v programsko opremo, strojna oprema, operacijski sistemi, zasebnost podatkov, moč računalnikov
Nemčija (Bavarska)	6.–12.	7.–10.	10.–12.		uvod v strojno in programsko opremo, računalniška arhitektura, osnove IKT, računalniško mišljenje, osnove informatike, računalništvo in družba, računalniška omrežja, algoritmi, podatkovne strukture

Nadaljevanje na naslednji strani.

Tabela 2.2 – nadaljevanje

Država	**Obvezen predmet RIN**	**Integrirano v drugem obveznem predmetu**	**Obvezno izbirni predmet RIN ali integriran v drug obvezno izbirni predmet**	**Prosto izbirni predmet RIN ali integriran v drugem prosto izbirnem predmetu**	**kurikulum**
Latvija		9.–12.	11.–12.		IKT, uvod v algoritme, programiranje, informacijska teorija, logika, algoritmi
Poljska	1.–12.	1.–3.	11.–12.		v OŠ računalniške aktivnosti, v srednji šoli računalniško mišljenje, algoritmi, programiranje
Slovaška	1.–13.				informacijske tehnologije, digitalne komunikacije, računalniško mišljenje, informatika in družba, informacijska družba
Slovenija	10.	1.–9.	11.–13.	7.–9., 11.–13.	(opisano v poglavju 2.3.3)

Nadaljevanje na naslednji strani.

Tabela 2.2 – nadaljevanje

Država	**Obvezen predmet RIN**	**Integrirano v drugem obveznem predmetu**	**Obvezno izbirni predmet RIN ali integriran v drug obvezno izbirni predmet**	**Prosto izbirni predmet RIN ali integriran v drugem prosto izbirnem predmetu**	**kurikulum**
Švica		1.–9.	9.–12.		uvod v algoritme, programska in strojna oprema, operacijski sistemi, zasebnost podatkov
ZDA	1.–5., 9.		6.–8., 10.–12.	9.–12.	najprej učenje konceptov RIN, potem RIN v modernem svetu, principi RIN in specifična področja RIN
Anglija	1.–10.		11.–12.		osnovna področja RIN, informacijske tehnologije in digitalna pismenost
Nizozemska	1.–9.		10.–12.		informatika v perspektivi, terminologija in veščine, operacijski sistemi, IKT

Opazimo, da kurikulumi izbranih držav pokrivajo podoben nabor učnih tem, kot so programska in strojna oprema, IKT in algoritmi. Razlike so predvsem v različnem poimenovanju področij, vključevanju različnih vsebin v različna področja ter pri katerih letih so predstavljene različne vsebine. Opazka, mimo katere nikakor ne moremo, je, da je od omenjenih držav Slovenija edina brez obveznega predmeta iz področja RIN v osnovni šoli. Po pregledu in primerjavi učnih ciljev različnih kurikulumov še opazimo, da je učni načrt zelo skop kar se tiče področja algoritmov in programiranja, saj recimo ne pokriva razlage podatkovnih tipov, vzporednega procesiranja, rešljivih in nerešljivih problemov, hevrističnih algoritmov, paralelnih tokov in podatkovne analize. Te vsebine sicer pokrivajo naprednejše države, kot so ZDA, Poljska ter Anglija [26].

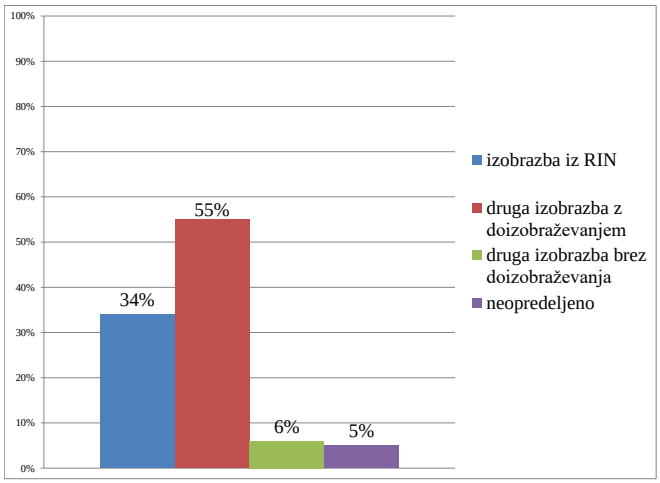
Posledice stanja RIN v slovenskem šolstvu se kažejo tudi v raziskavah, ki jih predstavimo v nadaljevanju.

2.5 Zaključki raziskav

Na osnovnošolskih učencih je bila opravljena mednarodna raziskava računalniške in informacijske pismenosti ICILS 2013 [28], ki kaže, da le 16 % slovenskih osmošolcev dosega nivo, s katerim izkazujejo zmožnost samostojne rabe IKT za zbiranje in obdelavo podatkov ter reševanje problemov. Le 0,3 % slovenskih osmošolcev je v raziskavi doseglo 4. raven, na kateri znajo ovrednotiti uporabnost ter zanesljivost informacij ter izbrati najustreznejšo informacijo za uporabo v komunikacijske namene, kar je glede pa povprečje zelo slab rezultat. Dodatni zaključek raziskave pravi, da je dejanska uporaba IKT v razredu povezana z usposobljenostjo učitelja in njegovim odnosom do IKT v šoli. Ugotovljeno je bilo tudi, da učenci uporabljajo IKT le za najosnovnejša opravila, kot so priprava predstavitev, iskanje podatkov na spletu in urejanje besedil. Statistika o aktivnostih, ki jih učitelji vpeljejo v pouk za razvoj višjih digitalnih kompetenc učencev, ni kaj prida opogumljajoča. Posamezne teme, kot so vrednotenje najdenih podatkov, delo na odprtih problemih ter obdelava in analiza podatkov, so bile v pouk vključene le v deležu od 11 % do 14 % odstotkov. Brez njih pa seveda učenci ne morejo doseči boljših rezultatov.

Zakaj učitelji ne vpeljejo več višjenivojskih aktivnosti v pouk? Enega od razlogov nam pojasni kvaliteta učiteljev RIN v slovenskih osnovnih šolah. Podatki iz Ministrstva za šolstvo, znanost in šport kažejo, da je v šolskem letu 2016/17 RIN poučevalo 399 učiteljev, od katerih je 135 (33,8 %) končalo študij ustrezne smeri. 245 učiteljev (61,4 %) ima osnovno izobrazbo iz drugega predmetnega področja, od katerih je 90 % opravilo ustrezno doizobraževanje za poučevanje RIN. Iz tega sledi, da 6 % učiteljev RIN ni imelo ustrezne izobrazbe za poučevanje. Diagram 2.1 prikaže te podatke v grafični obliki.

Naslednjo raziskavo, ki jo želimo povzeti, je iz šolskega leta 2009/2010 in govori o

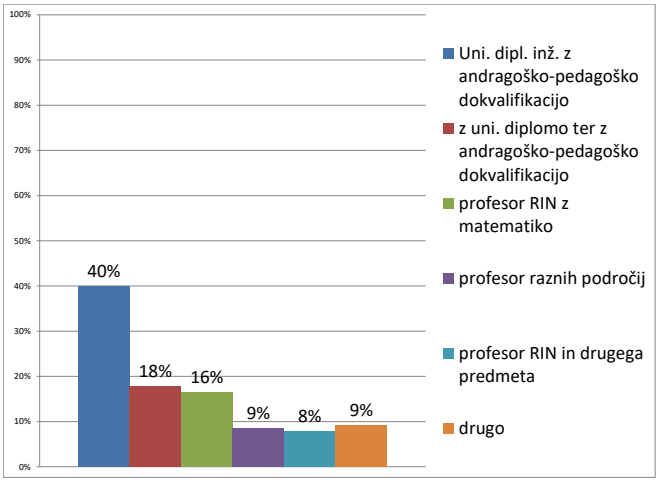


Slika 2.1: Temeljna izobrazba učiteljev RIN v slovenskih osnovnih šolah (N=399).

slovenskih srednješolskih učiteljih, ki poučujejo RIN [29]. Ta kaže, da je leta 2009 81,6 % šol imelo zaposlenega učitelja RIN, 15,6 % šol ga je imelo zaposlenega delno oziroma v obtoku pridobivanja, 2,9 % šol pa ga ni imelo zaposlenega. Pri tem se izkaže, da je 40 % učiteljev RIN inženirjev z univerzitetno diplomo in andragoško-pedagoško dokvalifikacijo, 17,9 % učiteljev ima univerzitetno diplomo s taisto dokvalifikacijo, sledijo profesorji RIN z matematiko (16,4 %), profesorji raznih področij (8,6 %) ter profesorja RIN in drugega predmeta (7,9 %). Diagram 2.2 prikaže te podatke v grafični obliki.

Pomemben pokazatelj znanja slovenskih učencev so tudi dosežki na mednarodnih tekmovanjih. Poglejmo si uvrstitve na mednarodnem tekmovanju IOI (*International Olympiad in Informatics*) [30]. Primerjali smo dosežke slovenskih dijakov z dijaki držav, ki smo jih opisali v tabeli 2.2. Tabela 2.3 prikazuje prejeto število bronastih, srebrnih in zlatih medalj po državah do vključno leta 2018. Opazimo, da smo po številu medalj med izbranimi državami na zadnjem mestu, pri čemer še nismo prejeli zlate medalje. Prednjačijo države za katere velja, da imajo posodobljene in ene boljših kurikulumov za RIN, kot so Anglija, Poljska, Slovaška in ZDA.

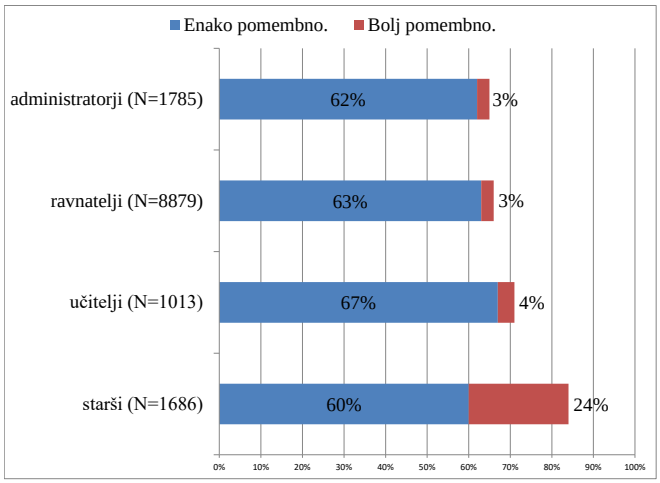
Za slednjo državo (ZDA) se je pred kratkim opravilo raziskavo, ki lepo ponazori, kakšno mnenje imajo učenci, učitelji in ravnatelji o poučevanju RIN na ameriških šolah. Ker je bila raziskava zelo obsežna, v nadaljevanju predstavimo ugotovitve le-te. Podobne raziskave za



Slika 2.2: Temeljna izobrazba učiteljev RIN v slovenskih srednjh šolah (N=140).

Tabela 2.3: Število osvojenih bronastih, srebrnih in zlatih medalj po državah na meddržavnem tekmovanju IOI.

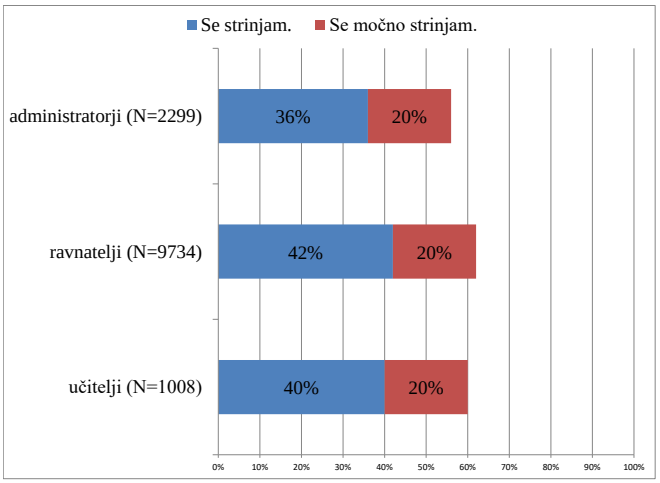
Država	Bronaste medalje	Srebrne medalje	Zlate medalje	Skupno
Slovenija	24	4	0	28
Avstrija	23	5	3	31
Nemčija	39	28	15	82
Latvija	45	26	7	75
Poljska	30	39	40	109
Slovaška	32	37	24	93
Švica	27	8	1	36
ZDA	16	34	49	99
Anglija	35	18	6	59
Nizozemska	39	15	3	57



Slika 2.3: Kako pomembno je omogočiti učencem izobraževanje RIN glede na obvezne predmete.

Slovenijo še ni bilo narejene. V decembru 2015 in januarju 2016 je podjetje Google najelo podjetje Gallup, da izvede obsežno raziskovalno delo v namen boljšega razumevanja, kako učenci, starši in učitelji dojemajo RIN v ameriških šolah [31]. V raziskavi je sodelovalo 16 000 učencev od sedmega razreda osnovne šole do drugega letnika srednje šole, njihovi starši, učitelji RIN, ravnatelji in administratorji. Glavna spoznanja raziskave so:

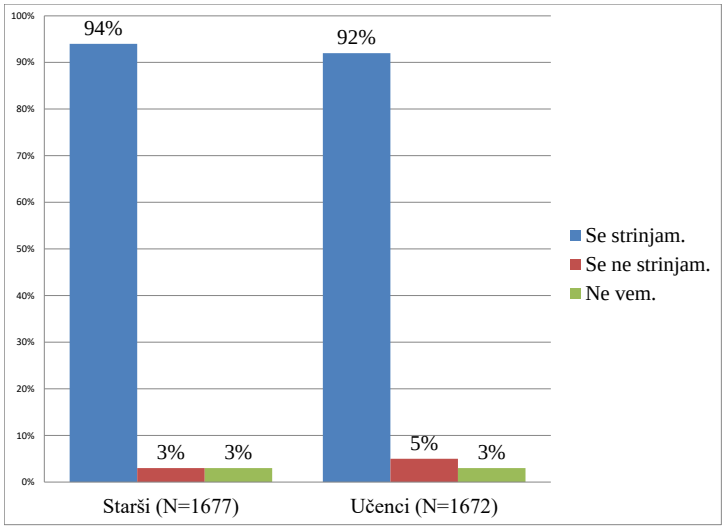
1. 84 % staršev, 71 % učiteljev, 66 % ravnateljev in 65 % administratorjev menijo, da je glede na predmete, kot so matematika, zgodovina in angleščina, enako pomembno oziroma bolj pomembno ponuditi učencem tudi predmet RIN (slika 2.3).
2. 76 % ravnateljev in 88 % ravnateljev srednjih šol pravi, da je njihovim učencem na voljo učenje RIN bodisi preko pouka, bodisi preko raznih klubov. 60 % ravnateljev pravi, da ponujajo učencem vsaj 1 predmet RIN, medtem ko za srednje šole enako pravi 78 % ravnateljev.
3. Medtem ko je v šolskem letu 2014/2015 ponujalo predmet RIN, kjer se učenci lahko učijo programiranja, 25 % ravnateljev, je ta odstotek zrasel na 40 % za naslednje leto. Pri tem je pomembno izraziti podatek, da se 60 % učiteljev, 62 % ravnateljev in 56 % administratorjev strinja, da mora biti RIN obvezen predmet (slika 2.4).
4. 28 % staršev in 30 % učiteljev je izrecno izrazilo podporo izobraževanju RIN vodstvu šol.



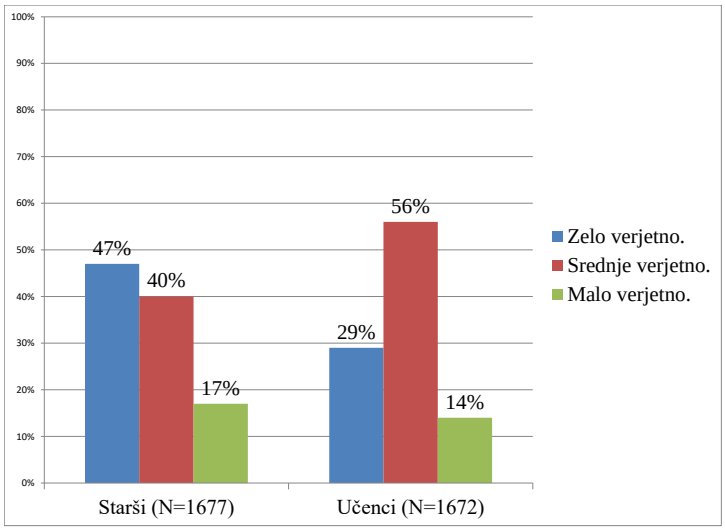
Slika 2.4: Večina učencev bi morala obiskovati predmet RIN, če je le-ta na voljo.

5. Podobno kot v enaki raziskavi leto prej, se tudi v tej raziskavi izkaže, da primanjkuje kvalificiranih učiteljev za učenje RIN – tako pravi 63 % ravnateljev in 74 % administratorjev, ko so bili vprašani, zakaj ne ponujajo učencem izobraževanja RIN.
6. Večina staršev (kar 93 % (N=1677)) se strinja, da je učenje RIN koristna poraba šolskih virov.
7. Praktično vsi starši in učenci se strinjajo, da informatika izboljšuje svet, kar je zelo vzpodbudno (slika 2.5).
8. Zelo zanimivi so rezultati vprašanja ali bo oseba potrebovala znanje RIN pri svoji zaposlitvi. Opazimo, da zelo majhen odstotek staršev in učencev misli, da je to malo verjetno, kar pomeni, da se zavedajo pomembnosti področja RIN (slika 2.6).
9. Zanimivi so tudi rezultati rahlo komične trditve »Ljudje, ki se ukvarjajo z RIN, morajo biti zelo inteligentni«. Tu se več kot polovica staršev in učencev strinja z izjavo, kar kaže na rahlo nerealno predstavo o tem poklicu (slika 2.7).

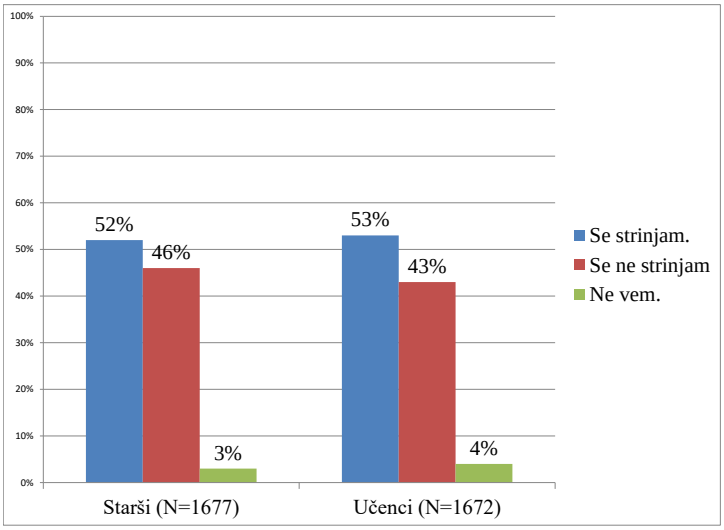
Zaključki raziskave so, da se večina učencev, učiteljev, staršev in ravnateljev v ZDA v veliki meri zaveda pomembnosti poučevanja RIN na šolah ter, da se mora posledično še veliko postoriti na tem področju.



Slika 2.5: Informatiki s svojim znanje izboljušjejo življenja ljudi.



Slika 2.6: Koliko verjetno je, da boste bodisi vi, bodisi vaš otrok potreboval znanje RIN pri svoji zaposlitvi?



Slika 2.7: Informatiki morajo biti zelo pametni.

Opisali smo stanje poučevanja in izobrazbe RIN v Sloveniji ter ga primerjali s tujino. Ugotovili smo, da je v Sloveniji stanje relativno bistveno slabše, zaradi česar so na posvetu o poučevanju računalništva in informatike v Ljubljani decembra 2017 [23] predstavili predloge za izboljšanje stanja, katere skuša ta magistrska naloga vpeljati. V naslednjem delu si poglejmo te predloge ter kako jih ta magistrska naloga skuša rešiti.

2.6 Prenova poučevanja računalništva in informatike v Sloveniji in projekt NAPOJ

V nadaljevanju predlagamo rešitev problema poučevanja RIN v Sloveniji, ki smo ga predstavili v prejšnjih razdelkih. Začeli bomo z opisom posveta poučevanja RIN, ki ga je pripravila skupina RINOS, nadaljevali z opisom slovenskega projekta NAPOJ, ki skuša praktično vpeljati predloge iz posveta, končali pa z opisom prispevka magistrske naloge k temu projektu.

2.6.1 RINOS

RINOS je strokovna delovna skupina za analizo prisotnosti vsebin računalništva in informatike v programih osnovnih in srednjih šol ter za pripravo študije o možnih spremembah. Skupino sestavlja 12 doktorjev in magistrov, katerih večina poučuje na slovenskih univerzah, kot so Fakulteta za računalništvo in informatiko, Pedagoška fakulteta in Fakulteta za elektrotehniko, računalništvo in informatiko v Mariboru.

Dne 1. 12. 2017 je omenjena strokovna skupina sodelovala na posvetu o poučevanju računalništva in informatike, ki ga je pripravil razred za matematične, fizikalne, kemijske in tehniške vede SAZU (Slovenska akademija znanosti in umetnosti). Posvet je bil razdeljen na 3 dele – nagovori, predstavitev problematike, potrebe po znanju in kompetencah; poučevanje RIN v Sloveniji ter razprava. Med drugim je bila predstavljena pomembnost RIN v sodobnem svetu, stanje izobraževanja RIN v tujini ter analiza poučevanja RIN v Sloveniji. Na posvetu je bilo ugotovljeno naslednje:

1. Stanje v svetu: sodobno poučevanje RIN se obrača k poučevanju temeljnih vsebin v obveznem delu kurikulov in praviloma skozi celoten vzgojno izobraževalni sistem od vrtca do konca srednje šole.

2. V Sloveniji v obveznem delu splošnega izobraževanja obstaja samo eno leto poučevanja RIN v gimnaziji, ki ima sicer precej odprt učni načrt. Posledica je pogosto zgolj opismenjevanje in poučevanje rabe tehnologije.

3. V državah, ki so že izvedle reformo RIN, so hkrati sistematično nadgradile sistem izobraževanja in nadaljnjega strokovnega usposabljanja učiteljev RIN finance, kadri, vsebine, deležnike).

Na podlagi zgornjih zaključkov so bili predstavljeni naslednji predlogi sprememb:

1. uvesti temeljne vsebine RIN v kurikul vrtcev ter učne načrte osnovne in srednjih šol ter pri tem razvijati zavedanje vzajemnega vpliva med tehnologijo in družbo;

2. zagotoviti celovito preverjanje opismenjevanja in uporabe tehnologij v okviru vseh predmetov skladno z obstoječimi učnimi načrti;

3. postaviti učinkovit sistem za kakovostno izobraževanje in nadaljnje strokovno usposabljanje vzgojiteljev in učiteljev na področju RIN;

4. vzpostaviti sistema odprtega izobraževanja, ki omogoča vključevanje deležnikov v oblikovanje vizije ter zagotavljanje in spremljanje kakovosti poučevanja RIN.

Ker brez dela ni jela, je potrebno zgornje predloge tudi vpeljati v praksi. Tu vstopi projekt NAPOJ.

Slika 2.8: Logo projekta NAPOJ.

2.6.2 NAPOJ

Projekt NAPOJ (Načrtovanje poučevanja Algoritmov in Programiranja ter OrganizaciJa skupnosti) [3] je bil izbran na razpisu podjetja Google CS4HS (*Computer Science for High School*). Vodi ga Univerza v Ljubljani, Fakulteta za računalništvo in informatiko v sodelovanju z MIZŠ (Ministrstvo za izobraževanje, znanost in šport) in ZRSŠ (Zavod republike Slovenije za šolstvo), pobudniki pa so Gregor Anželj, dr. Andrej Brodnik, mag. Matija Lokar ter Nataša Mori. Cilj projekta NAPOJ je slediti prvemu in tretjemu predlogu sprememb skupine RINOS, torej vzpostaviti skupnost učiteljev in profesorjev RIN (tretja točka) ter jih opremiti s potrebnimi gradivi in orodji (prva točka) s poudarkom na poučevanju programiranja. Za slednje NAPOJ uporablja učbenik Računalništvo in informatika 1 [4] ter projekt TOMO [5], za vzpostavitev in delovanje skupnosti pa je na voljo spletna učilnica na portalu SIO. V nadaljevanju si oglejmo omenjen učbenik in projekt TOMO.

Učbenik Računalništvo in informatika 1

Učbenik Računalništvo in informatika 1 [4] je elektronski učbenik, namenjen gimnazijam za pomoč pri poučevanju predmeta informatika (na sliki 2.9 je prikazan logo učbenika).Izdale so ga sestrske fakultete UL FRI, UM FERI ter UP FAMNIT leta 2015, avtorji pa so profesorji informatike iz gimnazij in profesorji iz univerz oziroma inštitutov. Učbenik je zasnovan na podlagi dveh vodil:

1. Učbenik naj dijaku poda takšno znanje, da bo le-ta sposoben tvoriti novo znanje in ustvarjanje novih informacijskih rešitev in ne zgolj uporabe tehnologij.

2. Učbenik skuša dijaku predstaviti tisto znanje, katerega so deležni njegovi vrstniki

po svetu.

Učbenik vključuje tri vidike – digitalno pismenost, informacijsko-komunikacijsko tehnologijo in temeljno znanje, pri čemer je večji poudarek na slednjem, saj le-to ni časovno omejeno, naslanja pa se tudi na tako imenovano računalniško mišljenje (*computational thinking*) (več v razdelku 3.1). Računalniško mišljenje velja za eno najpomembnejših kompetenc, ki jih lahko doseže izobraževanje v informatiki [32], česar se zaveda tudi strokovna skupina RINOS, ki pravi, da je urjenje računalniškega mišljenja pravzaprav urjenje v strategijah reševanja problemov, kar ima v sodobnem času izrednega pomena [23].

Tu se postavi vprašanje katera znanja štejemo pod temeljna znanja. Avtorji so vsebino učbenika zastavili tako, da podaja predvsem takšna temeljna znanja, ki jih je moč kasneje nadgraditi. Tako se e-učbenik deli na pet poglavij:

1. Uvod: kaj je informatika in kje vse jo najdemo v sodobnem svetu.
2. Programiranje in algoritmi: najosnovnejši elementi za pisanje programov v namen učenja zadovoljivega formalnega predznanja za spoznavanje osnov algoritmike. Deli se na 12 podpoglavij, pri čemer se prvih šest tiče učenja programiranja (osnovni koncepti programiranja, izdelava programov, zanke, tabele, funkcije in nizi), drugih šest pa algoritmov (osnovni pojmi algoritmike, časovna zahtevnost, zaporedje, urejanje, množica in slovar).
3. Sistemi: Strojna in programska oprema in odnos med njima. Med drugim vključuje bitno predstavitev, arhitekturo računalnika, operacijski sistem, varnost, procesi in vzporednost.
4. Omrežja in porazdeljeni sistemi: Osnove računalniških omrežij, vse od povezovanja naprav prek naslavljanja, interneta, svetovnega spleta do varnosti in zaščite.
5. Informatika in družba: Vpliv informatike na vsakdanje življenje posameznika, razvoj informacijskih tehnologij in podobno.

Čeprav je oblikovanje izvedbe učnega načrta pri informatiki povsem v domeni učitelja, avtorji učbenika priporočajo, da se uporabi vsaj prvo poglavje ter prvih šest podpoglavij drugega poglavja, saj le-ta dijaku pomagajo vzpostaviti zadovoljiv formalen okvir razmišljanj.

Projekt TOMO

TOMO je spletna storitev, namenjena hitrejšemu učenju programiranja (na sliki 2.10 je prikazan logo storitve). Ponuja bazo programerskih nalog, narejenih s strani učitelja. Te učenec samostojno rešuje ter avtomatsko dobi povratno informacijo sistema. Ta je

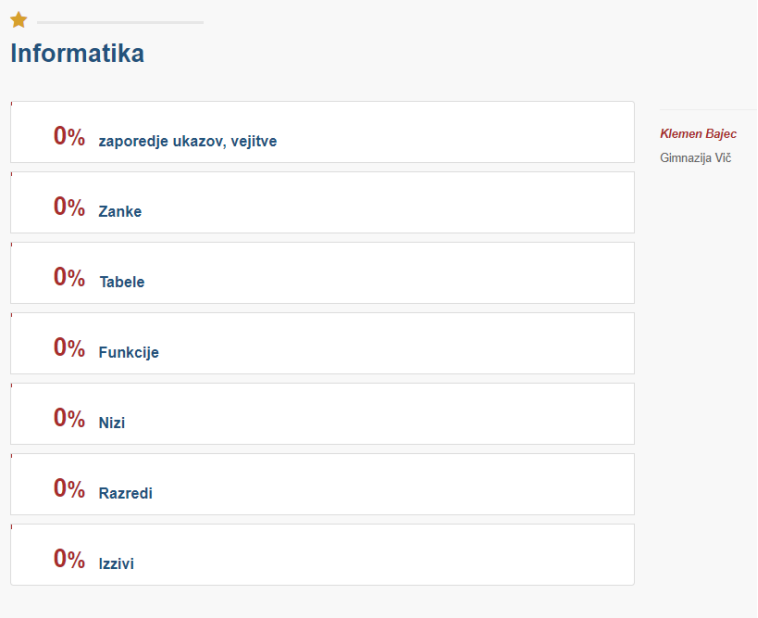
Slika 2.9: Naslovna stran učbenika RIN 1.

Slika 2.10: Logo spletne storitve TOMO.

odvisna od učitelja, ki je nalogo sestavil – lahko je enostavna (npr. `pravilna`/ `nepravilna rešitev`), ali pa za učenca uporabnejša (npr. `Za testni primer X vaša rešitev ni pravilna.`). S storitvijo TOMO lahko vsak učenec napreduje s svojo hitrostjo dela, saj dobi takojšnjo povratno informacijo, ali je njegova rešitev pravilna ali ne.

Za lažjo razumevanje si poglejmo kako kot dijak rešimo eno od nalog. Na spletni strani projekta imamo na voljo več predmetov.

Na sliki 2.11 je na vrhu strani zapisano ime predmeta (Informatika), pod njim pa sklopi iz e-učbenika RIN 1, pri čemer za vsakega piše kolikšen odstotek nalog smo že rešili. Ob desnem robu slike vidimo kdo je učitelj oziroma lastnik tega predmeta. Lastnik lahko poljubno dodaja in briše sklope ter dodaja, ureja in briše naloge iz posameznih sklopov. Ko kliknemo na sklop se znajdemo pred nalogami, kot je razvidno na sliki 2.12. Na desni



Slika 2.11: Sklopi predmeta.

strani za zapisane naloge, pod vsako nalogo pa naša trenutna uspešnost reševanja teh nalog. Rdeč krogec pomeni, da naloge še nismo reševali, rumen krogec pomeni, da smo jo rešili napačno, zelen krogec pa kaže na pravilnost naše rešitve. Da se lotimo reševanja naloge, moramo najprej shraniti datoteko na naš računalnik s klikom na gumb, prikazan na sliki 2.13.

S tem se nam datoteka z nalogo shrani na osebni računalnik. Odpremo jo lahko v poljubnem okolju (npr. Python IDLE, IEP, VisualStudio, Jupiter, PyCharm ipd.). Na sliki 2.14 smo jo odprli v okolju Python IDLE.

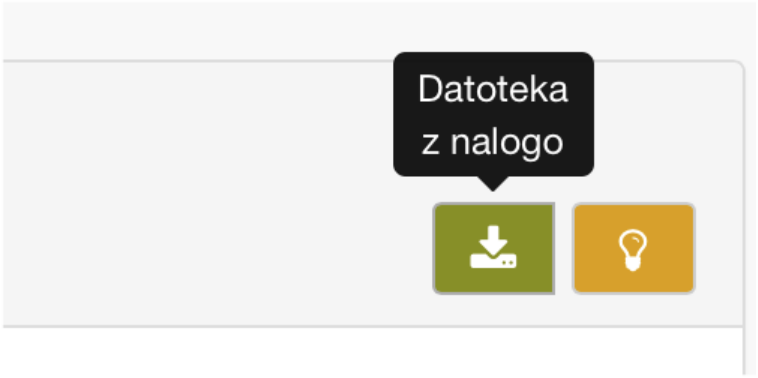
V datoteki so zapisana navodila za reševanje nalog. Ko datoteko oziroma modul poženemo, se zaženejo testi. Pri tem se naša rešitev samodejno pošlje na strežnik projekta TOMO, kjer se preveri pravilnost rešitve, za tem pa se uporabniku pošlje povratno informacijo v novem oknu, kot je prikazano na sliki 2.15.

Če je naša rešitev napačna, nam strežnik sporoči za kateri primer to velja (slika 2.16).

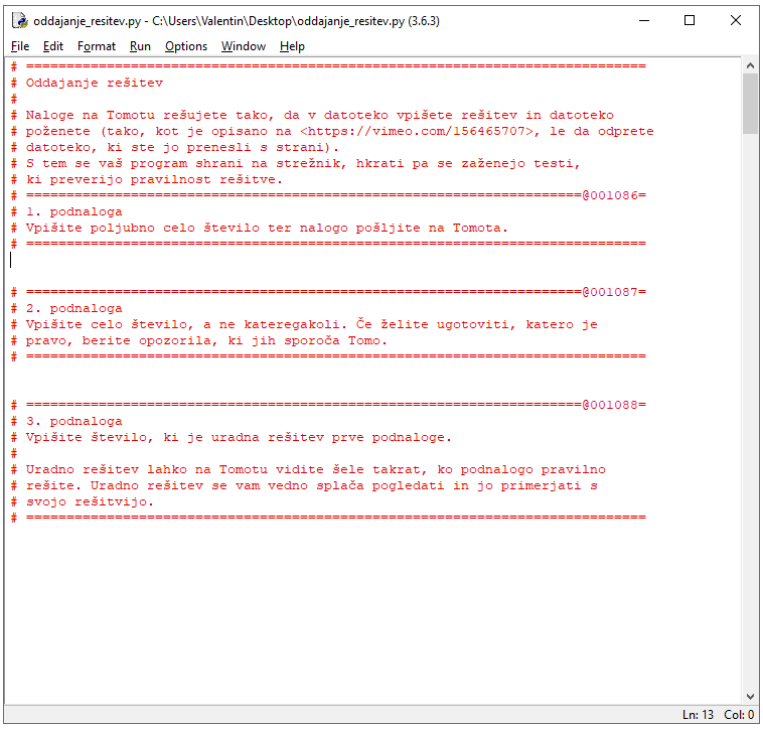
Ko nalogo pravilno rešimo se lahko lotimo naslednje po istem postopku.

Svoje rešitve določene naloge lahko vidimo, če kliknemo na gumb z žarnico ob imenu naloge. Odpre se nova stran, kjer je na levi prikazana naša rešitev, na desni pa uradna rešitev tistega, ki je nalogo sestavil. Učitelj lahko določi, kdaj učenec vidi uradno rešitev – pred reševanjem, po oddani pravilni rešitvi ali pa nikoli. Tako lahko učenec po oddani svoji rešitvi vidi še učiteljevo rešitev, ki je lahko boljša (enostavnejša in/ali hitrejša) od

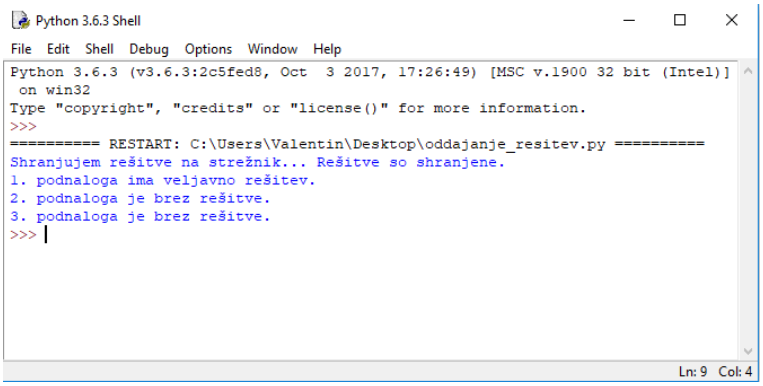
Slika 2.12: Primer nalog znotraj sklopa.



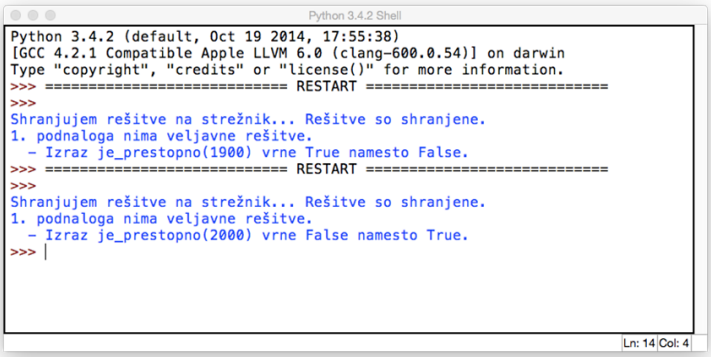
Slika 2.13: Gumb za dolpoteg naloge.



Slika 2.14: Naloga, odprta v okolju Python IDLE.



Slika 2.15: Povratna informacija za pravilno rešeno nalogo nalogo.



Slika 2.16: Povratna informacija za napačno rešeno nalogo.

učenčeve.

Predmete in naloge na spletni strani projekta dodajajo učitelji sami za svoje razrede, v katerih učijo. Če želijo, lahko kopirajo naloge drugih učiteljev, pri čemer kopirana naloga postane samostojna in ni več v povezavi z originalno. Učitelji imajov projektu pregled nad vsemi učenci, ki so vključeni v njihov predmet, pri čemer lahko med drugim za vsak sklop in nalogo vidijo, koliko učencev jo je rešilo.

Spletna učilnica

Za vzpostavitev in delovanje skupnosti učiteljev računalništva in informatike so pri projektu NAPOJ ustvarili spletno učilnico na portalu SIO (Slovensko izobraževalno omrežje) ter na sistemu Moodle. Tu sta na voljo predmeta Projekt NAPOJ (CS4HS) ter Projekt NAPOJ (CS4HS) – MU. Aktivno delovanje skupnosti poleg vodij projekta spodbujajo predvsem tako imenovani mojstri učitelji (MU, kot v imenu enega od predmetov). Slovenijo so razdelili na zahodno, vhodno, severno in osrednjo, pri čemer je vsakemu območju dodeljeno nekaj mojstrov učiteljev, ki za svoje območje skrbijo za razvoj skupnosti v obliki delavnic za lokalne učitelje ter izdelave učnih priprav in nalog v storitvi TOMO.

2.6.3 Prispevki magistrske naloge

Kot rečeno bodo na koncu projekta NAPOJ na spletni učilnici za učitelje RIN na voljo učne priprave, ki se navezujejo na sklope e-učbenika Računalništvo in informatika 1, poglavja programiranje in algoritmi. Cilj magistrske naloge je priprava učnih priprav ter nalog v storitvi TOMO za sklope o algoritmih (to so osnovni pojmi algoritmike, poraba in časovna

zahtevnost, zaporedje, urejanje, množica in slovar). Po oddaji učnih priprav na e-učilnico bodo le-te v uporabo na voljo vsem učiteljem RIN v Sloveniji.

Poglavje 3

Učne priprave

Pri pisanju učnih priprav moramo imeti v mislih, da bodo učenci razvijali računalniško mišljenje, zato se bomo na začetku poglavja posvetili tej temi. Nadalje je potrebno izbrati ustrezno okolje za programiranje, s čimer se bo naloga nadaljevala. Čeprav je zaporedje sklopov že določeno v e-učbeniku Računalništvo in informatika 1, se bomo vseeno morali prepričati, če v tujini niso izbrali kakšnega drugačnega zaporedja poučevanja le-teh. Na koncu pridemo do samega bistva poglavja, in sicer do opisa poteka pisanja učnih priprav.

3.1 Kako učiti računalniško mišljenje

Kot smo omenili v prejšnjem poglavju, je računalniško mišljenje pristop k reševanju problemov na način, ki ga lahko implementiramo z računalnikom in vključuje rabo abstrakcije, iteracije, rekurzije, procesiranja podatkov in drugih znanj [13]. Pri pisanju učnih priprav smo naleteli na vprašanje, kako učence učiti takšnega načina mišljenja. Hsu [33] opiše 16 kategorij učnih strategij za učenje računalniškega mišljenja, med katerimi so najbolj priljubljeni projektno učenje, učenje skozi igro in sodelovalno učenje. V magistrski nalogi smo si za didaktičen pristop k učenju računalniškega mišljenja izbrali sodelovalno učenje ter reševanje problemov, tj. učenje s prakso. Te naloge je treba izbrati pazljivo za začetnike v informatiki, kar dijaki prvega letnika gimnazije so. Ugodno za učence je, da so problemi kar se da neodvisni od razvojnega okolja, saj se tako lahko osredotočijo na problem in se ne mučijo z okoljem samim. Dober primer okolja je pisanje v tako imenovani psevdokodi. Poleg tega problemi ne smejo biti prelahki ali pretežki. Pri kompleksnejših problemih imamo na voljo več manevrskega prostora za kreativnost ter včasih tudi nenavadne rešitve problema s strani učencev [34].

Za izbiro, obliko in način podajanja problemov obstaja veliko znanstvenih člankov [35,

36, 37, 38, 39], pri čemer smo v magistrski nalogi izpostavili naslednje nasvete, ki so najbolje zaobjeli omenjene vire:

1. Naloga naj bo oblikovana tako, da bo zanimiva za učence.
2. Uporabljeno naj bo vodeno raziskovanje. To je, predstavimo problem in zagotovimo, da ima problem določene meje.
3. Problem naj ima več rešitev, pri čemer lahko tudi pokažemo delovanje rešitve na več vhodnih podatkih.
4. Zagotovljena naj bo raznolikost dejavnosti. Reševanje problemov je le ena od možnih načinov učenja računalniškega mišljenja. Programske dejavnosti se lahko gibljejo od zaključenih nalog s fiksnim končnim ciljem, z delno strukturiranimi dejavnostmi, ki vključujejo smernice konstruktov, ki se uporabljajo, do popolnoma odprtih projektov.
5. Vpeljani naj bodo primeri rešitev z vključenimi napakami. Pri tem se lahko uporabi učenčeve napačno napisane algoritme.
6. Učenci naj preberejo kodo in jo opišejo. S tem vadijo branje in razumevanje algoritmov.
7. Učenci naj načrtujejo rešitev, preden jo začnejo pisati. S tem se učijo, da je programiranje tudi reševanje problemov, ne zgolj pisanje kode.
8. Rešitev problema je lahko vnaprej podana v naravnem jeziku ali v psevdokodi. S tem učencem svetujemo, kako naj se reševanju naloge približajo.
9. Predstavi rešitev problema na drugačne načine.
10. Uporabljeno naj bo samostojno delo ter delo v skupinah.
11. Uporabljeno naj strokovno izrazoslovje. Namesto besede postopek je bolje uporabiti besedo algoritem.
12. Učenci naj imajo možnost predstavitve svojih rešitev. Primer je lahko izdelava spletne strani, na katero lahko kot učitelji postavimo najboljše rešitve učencev.
13. Uporabljeno naj bo preverjanje znanja. Še posebej če delo poteka v skupinah, ne bodo vsi učenci naredili enakega dela, kar pomeni, da se ne bodo vsi naučili vsega. S preverjanjem znanja, bodisi v obliki kvizov, delovnih listov ali česa drugega, lahko tudi počasnejši učenci dojamejo smisel snovi.

Pri pisanju nalog za učne priprave smo se skušali držati čim več zgoraj naštetih nasvetov, s čimer smo želeli zagotoviti njih čim višjo kakovost.

3.2 Izbira programskega okolja

Naslednje vprašanje se navezuje na izbiro ustreznega okolje za učenje računalniškega mišljenja. Določili smo naslednja dva kriterija [40]:

1. **primernost programskega jezika in okolja;**
   Izbran programski jezik mora biti enostavno berljiv in zapisljiv ter mora podpirati razširjene paradigme programiranja, kot je objektno programiranje. Programsko okolje mora biti enostavno, konsistentno, funkcionalno, stabilno, uporabniku prijazno ter imeti možnost lokalizacije.

2. **cena;**
   Ta kriterij nadalje razdelimo na ceno nakupa programskega okolja, ceno izdelave virov za predmet (učbenik, naloge v izbranem programskem jeziku ipd.) ter seznanitev učiteljev z izbranim programskim jezikom in okoljem.

Glavni cilji učenja programiranja na srednjih šolah je, da učence naučimo osnov programiranja in računalniškega mišljenja ter da pripravimo učence za nadaljnji študij. Po pregledu naših možnosti in primerjanju uporabljenih programskih jezikov v tujih učnih načrtih smo se odločili za programski jezik Python. Čeprav so drugi programski jeziki, kot so Java, C in C++ bolj razširjeni na fakultetah in v industriji, moramo imeti v mislih tudi, da se poučevanje programiranja srednješolcev, ki se prvič srečujejo s programiranjem, in bolj izkušenih programerjev, razlikuje [41].

Če primerjamo jezik Python po prvem opisanem kriteriju, ima v primerjavi z ostalimi programskimi jeziki veliko prednosti, kot so [41, 42, 43]:

1. čisto in minimalno sintakso;
2. dinamično tipiziranje spremenljivk;
3. izrecno semantiko;
4. takojšnjo povratno informacijo;
5. logično strukturno zasnovo, kot je zamik vrstic;
6. vsebuje brezplačno programsko okolje IDLE;
7. Python razvija neprofitno podjetje Python Software Foundation na podlagi skupnosti, zaradi česar uživa veliko podporo in popularnost;
8. zaradi svoje popularnosti ima na voljo veliko vodičev, učbenikov, nalog in splošne podpore v svetu.

Čeprav obstajajo učni načrti, v katerih se prvo polovico šolskega leta poučuje en programski jezik, drugo polovico pa drug, se za to opcijo nismo odločili, saj je naš cilj učenje algoritmov in računalniškega mišljenja, in ne programskih jezikov. Naše mnenje podpirajo rezultati anketiranja univerzitetnih študentov, kjer se jih je kar 44 % strinjalo ali zelo strinjalo, da bi morali pri predmetu, kjer so pol semestra uporabljali Python, pol pa Javo, nameniti več časa prvemu programskemu jeziku in manj slednjemu [44]. Pogosto namreč pride pri menjavi do nepotrebne zmede v glavah učencev, saj začnejo mešati sintakso omenjenih programskih jezikov.

Iz stališča kriterija cene je jezik Python brezplačen, glede izdelave gradiv in urjenja učiteljev pa vskoči projekt NAPOJ, kjer, kot že opisano, učitelji mojstri počnejo ravno to. Tudi iz tega stališča je torej programski jezik Python najboljša izbira.

3.3 Vrstni red sklopov

Preden začnemo s samim delom na učnih pripravah, moramo še določiti, v kakšen zaporedju bomo poučevali snov oziroma sklope. Naj spomnimo, da so ti sklopi osnovni pojmi algoritmike, poraba časa in časovna zahtevnost, zaporedje, urejanje, množica ter slovar. V tem zaporedju so tudi zapisani v e-učbeniku Računalništvo in informatika 1. V tabeli 3.1 smo naredili primerjavo med tremi gradivi – dvema knjigama za učenje algoritmov in podatkovnih struktur [45, 46] ter predmetom *Problem-Based Intro to CS* [44], ki ga učijo na ameriški univerzi RIT (*Rochester Institute of Technology*).

Ugotovili smo, da gre pri vseh virih večinoma za enako zaporedje sklopov, razen sklopa urejanje, ki je v e-učbeniku RIN 1 pred sklopoma množica in slovar, v drugih dveh učbenikih pa pred njima. Drugi vir se z osnovami algoritmike ne ukvarja, predvsem zato, ker predvideva, da bralec že pozna ta sklop. Zadnji vir sklopa o slovarju ne navaja.

Odločili smo se za vrstni red e-učbenika Računalništvo in informatika 1 iz treh razlogov:

1. E-učbenik RIN 1 je bil napisan s strani učiteljev in profesorjev namenoma za pomoč pri poučevanju informatike v slovenskih srednjih šolah.
2. Eden od ciljev e-učbenika RIN 1 je, da je pridobljeno znanje podobno tistemu, ki ga dobijo dijaki drugje po svetu.
3. Avtor magistrske naloge je v šolskem letu 2015/16 poučeval predmet informatika na gimnaziji Vič v Ljubljani po enakem vrstnem redu, kot je v e-učbeniku RIN 1. Na koncu šolskega leta smo dijake anketirali, kjer smo dijake vprašali po njihovem mnenju o predmetu informatika. Anketa je vsebovala vprašanje »Kakšno je vaše mnenje glede predmeta informatika?«, pod katero se se lahko dijaki po želji razpisali. Namen ankete je bil, da ugotovimo splošno mnenje dijakov o izvedbi predmeta

Tabela 3.1: Primerjava vrstnih redov sklopov različnih virov.

	Računalništvo in informatika 1	***Data structures and algorithm analysis***	***Data structures and algorithm analysis in Java***	***Problem-based intro to CS***
1	osnovni pojmi algoritmike		osnovni pojmi algoritmike	osnovni pojmi algoritmike
2	poraba časa in časovna zahtevnost	poraba časa in časovna zahtevnost	poraba časa in časovna zahtevnost	zaporedje
3	zaporedje	zaporedje	zaporedje	poraba časa in časovna zahtevnost
4	urejanje	množica	množica	urejanje
5	množica	slovar	slovar	množica
6	slovar	urejanje	urejanje	

informatika. Anketo je izpolnilo 12 dijakov. Štirim dijakom je bil predmet všečen, štirim uporaben in trem zanimiv. Enemu dijaku se predmet ni zdel zahteven, medtem ko je en dijak potožil, da smo učno snov jemali prehitro. Dva dijaka sta si želela več pozornosti pri učenju programiranja, medtem ko je bilo dvema dijakoma všečna napisana razlaga učne snovi na spletni učilnici. Glede domačih nalog so trije dijaki njihovo sprotno oddajanje ocenili kot uporabno, le eden je omenil, da so bile domače naloge občasno prezahtevne. En dijak je tudi pripomnil, da bi bilo lahko ocenjevanje domačih nalog boljše izvedeno. Pet dijakov je imelo negativno mnenje glede projektne naloge, 3 so izrazili željo, da bi projektni nalogi namenili več pozornosti, en dijak je definiral kriterije za ocenjevanje projektne naloge kot slabe, enemu dijaku pa je bil praktični del projektne naloge prezahteven. Ugotovili smo, da je bila večini dijakov izvedba predmeta všečna, uporabna in ne prezahtevna, z izjemo izvedbe projektne naloge, ki pa se te magistrske naloge ne tiče. Odgovori ankete so dodani v prilogi A.

3.4 Struktura učne priprave

Za učinkovitejše pisanje učnih priprav si poglejmo, zakaj jih potrebujemo ter za kakšno zgradbo učne priprave smo se odločili.

Učiteljeva učna priprava je nekakšen scenarij, po katerem poteka šolsko delo. Je vsebinski zapis učiteljeve priprave na šolsko uro. Razlogov za njihovo pisanje je veliko, med drugim zaradi optimalnega uresničevanja predpisanega učnega načrta, strokovne odgovornosti učitelja (priprava na pouk je obvezna po šolski zakonodaji), učiteljevega občutka strokovne kompetentnosti, gotovosti, varnosti in sproščenosti, za lažje spremljanje učiteljevega lastnega dela in napredka itd. [47].

Učna priprava vsebuje glavo priprave, jedro priprave ter analizo ure, ki jo učitelj opravi po končani učni uri. V glavi določimo naslednje elemente:

1. šola, letnik, datum in predmet izvajanja učne priprave;
2. učna tema – širša učna enota;
3. učna enota – učna snov za eno učno uro;
4. učne oblike – skupinsko, frontalno in/ali individualno delo;
5. učne metode – razlaga, pogovor, demonstracija in/ali reševanje problemov;
6. potrebno predznanje učencev;
7. operativni učni cilji;
8. učna sredstva;

9. didaktične etape učnega procesa;
10. medpredmetne povezave;
11. uporabljena literatura;
12. novi pojmi, ki jih učenci osvojijo;
13. morebitne priloge.

Jedro priprave je navpično razdeljeno na aktivnosti učitelja in aktivnosti učencev, vodoravno pa na učne stopnje. Razdelili smo ga na tri enote: uvodni del, glavni del in zaključni del. Uvodni del smo poimenovali uvajanje, glavni del obravnavanje učne snovi, zaključni del pa ponavljanje oziroma preverjanje. Za takšno strukturo smo se odločili zaradi konsistentnosti. Za sklope o programiranju iz e-učbenika Računalništvo in informatika 1 so namreč vse že napisane učne priprave v spletni učilnici projekta NAPOJ napisane s taisto trodelno zgradbo.

Vsak del jedra priprave smo glede na obravnavano snov po potrebi razdelili na razdelke, vsakemu od teh pa namenili določeno število minut in zapisali naloge učitelja, naloge učencev ter uporabljene učne oblike in metode. Na dnu učnih priprav so dodane priloge, omenjene v glavi priprave. Za lažje razumevanje osnovne strukture učne priprave, jo prilagamo v prilogi B.

Pri opisu pregleda rešitve v razdelku 1.2 smo omenili pripravo programskih nalog v projektu TOMO za vsako od učnih priprav. Tega zaradi časovne stiske nismo imeli.

Zadnji del izdelave učne priprave je ovrednotenje opravljene učne ure. Naše učne priprave smo ovrednotili tako, da smo za vsako od njih izvedli delavnico. Vsebine napisanih učnih priprav in potek delavnic opišemo v poglavju 4.

Poglavje 4

Rezultati in razprava

Prvi del poglavja je namenjen opisu napisanih učnih priprav, kjer za vsako od njih naštejemo učne cilje ter opišemo postopek pisanja učne priprave, drugi del pa je namenjen njihovemu ovrednotenju z izvezdbo delavnic.

4.1 Učne priprave po sklopih

Ker so učne priprave obsežne, smo v prilogo C dodali zgolj eno od njih, ostale pa so na voljo na spletnem naslovu `https://redmine.lusy.fri.uni-lj.si/documents/276`. V vseh učnih pripravah smo poskusili prikazati snov na zanimiv in praktičen način z upoštevanjem nasvetov za razvijanje računalniškega mišljenja, opisanimi v podpoglavju 3.1.

4.1.1 Osnovni pojmi algoritmike

Učni cilji sklopa:

1. učenec razume pojem abstrakcije in model abstrakcije,
2. učenec pozna pojem programerski problem, algoritem, pravilnost algoritma in ustavljivost algoritma,
3. učenec zna zapisati algoritem v naravnem jeziku, kot psevdokodo ter kot računalniški program.

Splošni cilj pri pisanju učne priprave je bil, da učenci poleg pojmov, kot so algoritem in programerski problem, razumejo pomen abstrakcije ter da se reševanja problema lotijo na znanstven način in ne pišejo rešitev problema na slepo.

V uvodnem delu učne priprave smo dodali krajši motivacijski učiteljev govor, s katerim smo želeli učencem pokazati pomembnost učne enote. Pri pisanju glavnega dela učne priprave smo se najprej spopadli z vprašanjem, kako bi učence naučili pojma abstrakcije. Abstrakcija je bistven koncept v računalništvu, ki velja za pomembno orodje pri razvoju programske opreme. Je proces generalizacije in ohranitve pomembnih informacij nekega koncepta po sistemu posplošitve specifičnega primera ali primerov problema; identifikacije, izluščenja in izolacije bistvenih sestavin ter ignoriranje nebistvenih podrobnosti [48].

Prvi učni cilj smo izpolnili tako, da smo vzeli model reševanja problemov z uporabo abstraktnih podatkovnih struktur (v nadaljevanju APS) [48], saj uporaba APS velja za uporabno orodje za reševanje računalniških problemov [49]. Model se deli na dva dela, vsak od njih ima več poddelov:

1. razumevanje problema;
   (a) konceptualizacija – razumevanje in opis problema;
   (b) posplošitev – zanemarjanje nebistvenih podrobnosti;
   (c) izbira abstraktnega modela – izbira APS;
2. izdelava ustrezne rešitve;
   (a) formalizacija – povežemo podan problem in izbrano APS;
   (b) realizacija in testiranje – pisanje rešitve in testiranje pravilnosti.

Čeprav v omenjenem viru avtor opiše realizacijo in testiranje kot dva ločena sklopa, se nam je njuna združitev zdela bolj smiselna, saj se v praksi nikoli ne izkaže, da sprva v celoti napišemo rešitev računalniškega problema in ga šele nato testiramo, ampak se ti stopnji dogajata izmenjaje – napišemo del rešitve, testiramo njeno pravilnost, dodamo še en del rešitve, jo ponovno testiramo itd.

Pri pisanju učne priprave smo spoznali, da zgolj suhopoarno naštevanje delov modela ni zanimivo za učence, zato smo se odločili snov umestiti v programersko nalogo, pri kateri bi sodelovali tudi učenci. S tem smo dosegli drugi učni cilj, ki smo si ga zadali. Podali smo nalogo, kjer moramo kot programerji sprogramirati delovanje gumba `Nazaj` v spletnem brskalniku (s klikom gumba se odpre zadnja obiskana spletna stran). Tako smo v sklopu konceptualizacije definirali glavne naloge gumba (hranjenje spletnih povezav, dodajanje novih spletnih povezav ter pridobitev zadnje spletne povezave). V posplošitvi smo spoznali, da lahko v naši rešitvi hranimo poljubno vrsto podatkov, ne zgolj nizov (spletnih povezav). Pri izbiri abstraktnega modela smo izbrali APS sklad, saj omogoča hranjenje podatkov od najnovejšega do najstarejšega, pri formalizaciji pa smo povezali naloge iz konceptualizacije z operacijami sklada. Realizacijo in testiranje smo izvedli tako, da smo za vsako podnalogo iz formalizacije napisali rešitev v naravnem jeziku, s psevdokodo ter

v programskem jeziku Python. S tem smo dosegli tretji zadani učni cilj sklopa. Za lažje razumevanje si poglejmo zapis operacije, ki se zgodi ob kliku gumba `Nazaj`. Operacijo smo zapisali v naravnem jeziku(4.1), s psevdokodo (4.2) in v programskem jeziku Python (4.3).

Algorithm 4.1: Zapis pridobitve zadnje spletne povezave iz sklada v naravnem jeziku.

```
Ce sklad ni prazen, vzemi ven zadnjo dodano spletno povezavo ter
jo vrni.
```

Algorithm 4.2: Zapis pridobitve zadnje spletne povezave iz sklada s psevdokodo.

```
Ce sklad ni prazen, potem:
  povezava = vzemi povezavo iz sklada
  vrni povezava
```

Algorithm 4.3: Zapis pridobitve zadnje spletne povezave iz sklada v programskem jeziku Python.

```
def pridobiZadnjoPovezavo(povezave):
  if not jePrazen(povezave):
    povezava = vzemi(povezave)
    return povezava
```

Za zaključni del učne priprave smo oblikovali delovni list, ki služi za ponovitev in utrditev predelane snovi.

4.1.2 Poraba časa in časovna zahtevnost

Učni cilji sklopa:

1. učenec razume pomen časovne zahtevnosti algoritma,
2. učenec zna meriti čas izvajanja programa,
3. učenec zna prešteti število osnovnih operacij v algoritmu,
4. učenec razume pomen uporabe in zna našteti rede časovnih zahtevnosti,
5. učenec zna izboljšati časovno zahtevnost enostavnejših algoritmov.

Za splošni cilj učne priprave smo si zadali, da učenec razume pomen časovne zahtevnosti algoritma ter da ga zna na različne načini meriti s poudarkom na sodobnem načinu merjenja, to je z redi časovne zahtevnosti.

V uvodnem delu učne priprave smo dodali krajši motivacijski učiteljev govor, s katerim smo želeli učencem pokazati pomembnost učne enote. Govor vsebuje nekaj vprašanj, kjer se pokaže pomembnost hitrosti izvajanja rešitve. Primer takšnega vprašanja je ≫Koliko časa potrebuje avtomobtil za samodejno zaviranje, če oceni, da se lahko zgodi nesreča?≪. Učence smo v temo vključili tako, da so morali tudi sami podati nekaj primerov takšnih vprašanj. S tem smo izpolnili prvi učni cilj.

V prvem sklopu glavnega dela učne priprave smo se dosegli drugi učni cilj tako, da smo prikazali merjenja časa v programskem jeziku Python. Tu so morali učenci napisati programersko rešitev za nalogo `poisciMin(sez)` za iskanje najmanjšega elementa v seznamu. Profesor nato požene rešitev na zelo veliki tabeli (v učni pripravi je tabela velika 1 000 000 elementov), pri čemer meri hitrost izvajanja z uporabo Python modula `time`. Sledi prehod na naslednji sklop z vodenim pogovorom učitelja z učenci na temo, ali je takšna oblika merjenja hitrosti algoritma ustrezna. Skupaj pridejo do zaključka, da ni, in preidejo na naslednji sklop, tj. štetje osnovnih operacij v algoritmu. Sledi razlaga učitelja štetja osnovnih operacij na rešitvi prej podane naloge `posciMin(sez)`. S tem smo izpolnili tretji učni cilj. Sledi navezava na zadnji sklop glavnega dela, to je redi časovne zahtevnosti. Z učiteljevo pomočjo učenci pridejo do spoznanja, da ni treba vedeti točnega števila operacij algoritma, saj so računalniki dovolj hitri, da rahla sprememba v številu operacij ne botruje bistveni razliki v izvajanju algoritma. Učitelj nato predstavi rede časovne zahtevnosti in notacijo veliki O. Ker je ta snov dokaj zahtevna, smo v učno pripravo dodali razlago redov časovnih zahtevnosti s spletno stranjo [50], kjer učitelj izriše in opiše primer logaritmične, linearne, kvadratne in eksponentne funkcije. Z razlago redov časovnih zahtevnosti smo izpolnili četrti učni cilj.

Za zaključni del učne priprave smo izdelali obsežen delovni list, ki lahko v primeru časovne stiske služi tudi kot domača naloga. Za izdelavo večine nalog smo uporabili že izdelane primere [51]. Zadnjo nalogo smo izdelali z virom [52], ki svetuje izdelavo nalog s tremi specifičnimi podnalogami. V prvi podnalogi učenec opiše v nalogi podan algoritem – kateri programerski problem rešuje in kako ga rešuje. V drugi podnalogi učenec določi red časovne zahtevnosti algoritma, v zadnji podnalogi pa le-tega izboljša tako, da bo njegova rešitev za časovni red hitrejša (primer: če ima podan algoritem red časovne zahtevnosti O(n), mora učenčeva rešitev vsebovati največ O(1)). Takšna zgradba naloge se izkaže za zelo učinkovito [52]. S tem smo tudi izpolnili zadnji, četrti učni cilj te učne teme.

4.1.3 Zaporedje

Učni cilji sklopa:

1. učenec razume pojem in pomena zaporedja,
2. učenec razume podatkovno strukturo tabela,
3. učenec razume podatkovno strukturo povezan seznam,
4. učenec razume podatkovno strukturo vrsto,
5. učenec razume podatkovno strukturo sklad.

Uvodni del učne priprave smo začeli z obnovo snovi, ki je učencem že znana, tj. podatkovna struktura tabela, pri čemer smo dodali razlago časovnih zahtevnosti osnovnih operacij, to so `dostop do poljubnega elementa`, `dodajanje elementa`, `vstavljanje elementa`, `brisanje elementa`, `odvzemanje elementa` ter `število elementov`. S tem smo izpolnili drugi učni cilj. S to že obravnavano snovjo učitelj vodi pogovor z učenci o pomenu in praktični uporabi zaporedij, s čimer smo dosegli prvi učni cilj.

Sledi razlaga povezanega seznama in s tem zadostitev tretjemu učnemu cilju. To strukturo smo spoznali z uporabo listkov [53], na katerih sta dva podatka. Prvi je `Element` (v našem primeru ime hrane), drugi pa `Naslednji element`, ki je na začetku prazen. Učitelj določi učenca, ki najprej izbere poljuben listek (hrano), ki jo poda poljubnemu sošolcu. To naredi večkrat, ob vsakem podajanju listka pa morajo učenci ustrezno ugotoviti, na katerih že oddanih listkih se izpolni podatek `Naslednji element`. Primer: če učenec najprej izbere listek s hrano banana, nato pa listek s hrano jabolko, mora zatem na prvem listku pod `Naslednji element` zapisati jabolko. Ko učenci razumejo postopek dodajanja elementov, na podoben način sledi še njihovo vstavljanje, odvzemanje in brisanje. Za konec učenci s pogovorom skupaj z učiteljem ugotovijo rede časovnih zahtevnosti teh operacij.

Četrti učni cilj je dosežen v naslednjem sklopu glavnega dela – razlaga strukture vrsta, ki jo učitelj z učenci implementira z uporabo povezanega seznama. Za konec sledi razlaga strukture sklad, pri kateri smo za implementacijo uporabili tabelo. S tem zadostimo petemu učnemu cilju. Ob razlagi sklada in vrste smo se oprli na e-učbenik [4].

V zaključnem delu učitelj skupaj z učenci ponovi snov obravnavanih podatkovnih struktur.

4.1.4 Urejanje

Učni cilji sklopa:

1. učenec spozna problem urejanja podatkov,
2. učenec spozna različne koncepte urejanja,

Tabela 4.1: Hranjenje števila primerjav za posamezen koncept urejanja.

	prosto urejanje	**urejanje z izbiranjem**	**urejanje z vstavljanjem**	**urejanje z zlivanjem**	**hitro urejanje**
1. skupina					
2. skupina					
3. skupina					
4. skupina					

3. učenec razume dobre in slabe strani obravnavanih konceptov.

V uvodnem delu učne priprave učenci skupaj z učiteljem ugotovijo pomembnost urejanja z naštevanjem primerov, kjer je pomembno, da so stvari urejene, ter po kakšnih kriterijih so urejene, denimo odtenek barve, moč zvoka, mehkost predmeta ipd. S tem učenec spozna problem urejanja podatkov, kar je tudi prvi učni cilj sklopa.

V glavnem delu smo izbrali razlago snovi z izvedbo aktivnosti, pri kateri učenci s pripomočki spoznajo različne koncepte urejanja elementov [54, 55]. Odločili smo se za uporabo tehtnic in lončkov z različno težo, pri čemer učenci s tehtnico uredijo lončke po teži od najlažjega do najtežjega. Sprva smo uporabljali igralne karte, a smo to idejo opustili, saj je pomembno, da učenci elementov ne morejo urediti z metodo ostrega pogleda, ker se s tem izgubi razumevanje, kako računalnik primerja elemente.

Učna ura se prične tako, da učitelj učence razdeli v skupine. Namen skupin je, da bodo učenci z medskupinskim tekmovanjem bolj zagnani za delo in posledično za učenje. Vsaka skupina dobi eno tehtnico in nekaj lončkov, ki so različno težki. Učitelj skupinam poda navodila, naj lončke uredijo glede na težo, pri čemer morajo spremljati število primerjanj, tj. kolikokrat so primerjali teži dveh lončkov. Po končani nalogi skupine sporočijo svoja števila primerjanj učitelju, ta pa jih zapiše na tablo v tabelo, kot je prikazana v tabeli 4.1.

Učenci ob tem spoznajo, da je lahko urejanje različno učinkovito glede na to, kateremu konceptu urejanja sledimo.

Sledijo štirje sklopi – urejanje z izbiranjem, urejanje z vstavljanjem, urejanje z zlivanjem ter hitro urejanje – ki potekajo na enak način. Na začetku vsakega sklopa učitelj razloži določen koncept urejanja, pri čemer za primer uporabi kar seznam elementov, ki jih uredi po velikosti. Zatem nastopi delo v skupinah, kjer vsaka skupina uredi lončke po predstavljenem konceptu ter si zapisuje število primerjanj, ki jih učitelj na koncu zapiše v

tabelo 4.1 na tabli. Na koncu sklopov sledi pogovor o uporabljenem konceptu. Postavljajo se vprašanja kot denimo: zakaj pri nekaterih konceptih vse skupine dobijo enako število primerjanj, pri drugih pa ne; ali so kakšni posebni primeri, ko se koncept izkaže za slabega ipd.

Zadnji sklop glavnega dela je namenjen pregledu časovnih zahtevnosti konceptov urejanj. Učitelj prikaže psevdokodo posameznega koncepta, čemur sledi pogovor o tem, kakšen red časovne zahtevnosti pripada vsakemu.

V zaključnem delu učenci z učiteljem ponovijo pomen urejanja in razlike med koncepti urejanja.

4.1.5 Množica

Učni cilji sklopa:

1. učenec razume abstraktno podatkovno strukturo množica,
2. učenec razume abstraktno podatkovno strukturo dvojiško drevo in dvojiško iskalno drevo,
3. učenec pozna osnove uravnoteženja dvojiškega iskalnega drevesa.

Splošni cilj sklopa je naučiti učence pomena strukture množica in jih seznaniti z dvojiškim iskalnim drevesom, saj je zanimivo za širjenje razumevanja RIN.

V uvodnem delu učne priprave učenci skozi metodo pogovora spoznajo podatkovno strukturo množica in njene primere. S tem učenci tudi razumejo njeno vlogo v vsakdanjem življenju.

Glavni del smo razdelili na pet sklopov, ki so navedeni v nadaljevanju. Prva dva služita za zadostitev prvega učnega cilja, drugi in tretji sklop izpolnjujeta drugi učni cilj, zadnji sklop pa tretji učni cilj.

Podatkovni tip `set`

Učitelj razloži podatkovni tip set programskega jezika Python ter operacije `union` (unija) in `intersection` (presek). Za utrditev znanja učenci na koncu sklopa rešijo programersko nalogo, kjer imajo podane podatke, na katere dneve je prost vsak od štirih prijateljev, ugotoviti pa morajo, kateri prosti dan je skupen vsem. Reševanje naloge na pamet se lahko izkaže za zahtevno, v programerski rešitvi pa zgolj uporabimo omenjeno operacijo `intersection`, s čimer že dobimo rešitev naloge.

Implementacija množice s tabelo

Tu učitelj s pomočjo učencev implementira množico z uporabo tabele. V programskem jeziku Python napiše funkcijo `ustvariMnozico()`, ki ustvari novo prazno množico. S

funkcijo `dodaj()` doda element v množico. S funkcijo `jePrisoten()` preveri, ali je prejet element že vsebovan v množici. S funkcijo `odstrani()` pa odstrani prejet element iz množice. Nato se še preveri pravilnost rešitve ter poda programerska naloga, kjer morajo učenci napisati funkcijo, ki prejme dve množici in izpiše skupne elemente. Na koncu sklopa z metodo pogovora učitelj z učenci ugotovi, da je red časovne zahtevnosti omenjenih osnovnih operacij O(n). Učitelj predstavi novo podatkovno strukturo, dvojiško iskalno drevo.

V e-učbeniku je obrazložena tudi implementacija množice s tabelo logičnih vrednosti, vendar se nam je ta zaradi časovne omejenosti zdela nesmiselna, zato smo jo izpustili. Dodaten čas smo porabili, da smo podrobneje razložili uravnoteženje drevesa.

Dvojiško iskalno drevo

Učitelj razloži strukturo in delovanje dvojiškega iskalnega drevesa (v nadaljevanju drevesa), nato sledi njegova implementacija s tabelo. Tu smo se odločili za klasični pristop, ko so otroci vozlišča `A` na indeksih `2*i` in `2*i+1`, pri čemer je `i` indeks vozlišča `A`. Funkcije `ustvari()`, `dodaj()` in `jePrisoten()` smo zavoljo razumevanja strukture najprej zapisali s psevdokodo, nato pa še v jeziku Python. Za funkcijo `odstrani()` smo se zgledovali po [4] in je zaradi kompleksnosti kode nismo napisali. S tem sklopom zadostimo drugemu učnemu cilju.

Časovna zahtevnost dvojiškega iskalnega drevesa

Tu učitelj razloži, zakaj operaciji `dodaj()` in `odstrani()` potrebujeta O(log(n) časovno zahtevnost.

Uravnoteženje dvojiškega drevesa

Zadnji sklop pripada razlagi uravnoteženja drevesa. Za razlago te operacije smo uporabili poenostavljen način [56]. Postopek leve rotacije za vozlišče `A` se izvede na naslednji način:

1. identificiraj najmanjšega otroka v desnem poddrevesu kot vozlišče `MIN`
2. kopiraj vozlišče A in ga vstavi v njegovo levo poddrevo
3. vrednost vozlišča `MIN` zapiši v vozlišče `A`
4. izbriši vozlišče `MIN` iz desnega poddrevesa vozlišča `A` in po potrebi preveži otroke vozlišča `MIN`

Za postopek desne rotacije ustrezno preslikamo navodila:

1. identificiraj najmanjšega otroka v levem poddrevesu kot vozlišče `MIN`

2. kopiraj vozlišče A in ga vstavi v njegovo desno poddrevo
3. vrednost vozlišča `MIN` zapiši v vozlišče `A`
4. izbriši vozlišče `MIN` iz levega poddrevesa vozlišča `A` in po potrebi preveži otroke vozlišča `MIN`

S tem postopkom se učenci spoznajo z osnovami uravnoteženja drevesa, s čimer zadostimo tretjemu učnemu cilju.

V zaključnem delu učne priprave učenci za ponovitev in utrditev snovi rešijo delovni list, katerega rešitve skupaj z učiteljem preverijo na koncu ure.

4.1.6 Slovar

Učni cilji sklopa:

1. učenec razume pomen iskanja podatkov,
2. učenec razume in implementira podatkovno strukturo slovar,
3. učenec spozna razpršeno tabelo in razpršilno funkcijo,
4. učenec spozna problem sovpadanja pri razprševanju in njegovo reševanje.

Pri pisanju učne priprave smo se osredotočili na razumevanje učenca, da je iskanje podatkov pomembna tema v RIN ter na to, da učenec zna napisati učinkovito podatkovno strukturo, ki rešuje ta problem. Iz učne priprave smo izpustili podpoglavje »Slovar kot temeljni gradnik podatkovnih baz« e-učbenika RIN 1 [4], saj smo mnenja, da zaradi omejenosti časa in obširnosti teme učencem ne bi uspeli podati uporabnega znanja.

V uvodnem delu učne priprave učitelj predstavi problem iskanja podatkov in skupaj z učenci našteje praktične primere, v katerih iščemo podatke glede na podani ključ (primer: telefonski imenik, kjer je ključ ime in priimek osebe, iskani podatek pa pripadajoča telefonska številka). S tem smo izpolnili prvi učni cilj.

V glavnem delu učne priprave smo se odločili predstaviti snov s programersko nalogo, kjer je potrebno hraniti podatke tipa `<ime priimek>:<bančni račun>`, pri čemer mora biti iskanje računa po imenu in priimku čim hitrejše. Učitelj skupaj z učenci reši nalogo z uporabo tabele, kjer bančni račun predstavlja indeks, v katerega bomo zapisali lastnika računa. Pri tem učitelj pokaže na problem, ko je številka bančnega računa zelo velika, saj to posledično pomeni, da mora biti tudi tabela zelo velika. Zaradi tega učitelj predstavi podatkovno strukturo razpršena tabela, kjer predstavi tudi razprševanje in enostavno razprševalno funkcijo, ki uporabi operacijo modul [57]. Indeks se tako izračuna s formulo `kljuc mod length(slovar)`. S tem smo dosegli drugi in tretji učni cilj.

Nadaljevali smo s snovjo o sovpadanju elementov oziroma kolizijah. Učitelj učencem poda nalogo, kjer morajo vstaviti podana števila na ustrezno mesto v razpršilni tabeli, pri čemer se namenoma zgodi, da podana razpršilna funkcija vrne isti indeks za dve različni števili. Na tem mestu učitelj začne pogovor o tem, kako bi učenci rešili problem sovpadanja. Nato razloži in pokaže psevdokodo za zaprto razprševanje, tj. vstavljanje na naslednje prosto mesto v tabeli, in odprto razprševanje, tj. hranjenje povezanega seznama v vsaki celici razpršilne tabele. S tem smo dosegli četrti učni cilj. Za zaprto razprševanje morajo učenci podano psevdokodo še napisati v jeziku Python.

Za zaključni del smo pripravili delovni list, ki služi kot ponovitev in utrditev obravnavane snovi.

4.2 Delavnice in ovrednotenje

4.2.1 Opredelitev raziskovalnega problema

V magistrskem delu raziskujemo poučevanje algoritmov pri predmetu informatika v srednješolskem programu gimnazija. Odločili smo se za izvedbo delavnic, kjer bomo preverjali znanje dijakov pred in po delavnici.

V raziskavi bomo ugotavljali, ali so dijaki osvojili učne cilje, ki smo jih zastavili v učnih pripravah v razdelku 4.1. Zanima nas, ali so dijaki dosegli zastavljene učne cilje z vidika preverjanja znanja.

Raziskovalna vprašanja

Določili smo naslednja raziskovalna vprašanja:

- Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo urejanje?
- Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo zaporedje?
- Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo množica?
- Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo slovar?
- Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo osnovni pojmi algoritmike?

4.2.2 Metoda in raziskovalni pristop

V okviru empiričnega dela magistrske naloge smo izvedli raziskavo, ki temelji na kvantitativnem pristopu, kjer smo pridobili rezultate s pisnim preizkusom znanja. Dijaki so reševali naloge, s katerimi smo preverili njihovo znanje o učni temi algoritmi. Za iskanje odgovora na vsa raziskovalna vprašanja smo uporabili rezultate, pridobljene na pisnem preizkusu znanja pred in po delavnici. Rezultate smo med seboj primerjali in na podlagi tega prišli do ugotovitve, ali so učenci dosegli zastavljene učne cilje za posamezno delavnico.

Vzorec

Uporabljeni vzorec je bil slučajnosten. Sodelovali so učenci prvega in drugega letnika srednje šole. Skupino je sestavljalo 10 učencev, pri čemer med spoloma nismo razlikovali.

Raziskovalni instrumenti

Za odgovarjanje na raziskovalna vprašanja smo uporabili pisno preverjanje znanja pred in po vsakem eksperimentu.

Testiranje

Pred začetkom in po koncu izvajanja vsake delavnice so dijaki rešili pisni preizkus znanja. Vsak vsebuje od 3 do 4 naloge oziroma vprašanja, ki se nanašajo na učno snov z delavnice. Odgovor na vsako vprašanje iz preizkusa znanja je bodisi pravilen bodisi napačen. Za pravilen odgovor je učenec dobil eno točko, za nepravilen pa nič točk. Preizkuse znanja prilagamo k opisu delavnic.

Postopki obdelave podatkov

Pri raziskovanju smo uporabili kvantitativni pristop. Za analizo pridobljenih podatkov smo uporabili ustrezna orodja. Podatke smo obdelali z računalniškimi programi Microsoft Excel in IBM SPSS Statistics. Ker imamo binomske spremenljivke in dva povezana vzorca, smo podatke analizirali z uporabo McNemarjevega testa. Rezultate prikažemo v opisu vsake delavnice.

4.2.3 Izvedba raziskovanja

V nadaljevanju poglavja opišemo izvedene delavnice v kronološkem zaporedju. Za vsako delavnico opišemo njen potek, predstavimo rezultate in ovrednotenje pisnega preverjanja znanja ter po potrebi morebitne popravke učnih priprav.

Delavnica 1: Urejanje

Učno pripravo smo za potrebe delavnice rahlo spremenili. Del »časovna zahtevnost« smo predstavili bolj opisno, brez strokovnih besed, kot so red časovne zahtevnosti veliki O in rekurzija. Tako smo se odločili, ker učenci še niso spoznali poglavja o časovnih zahtevnostih algoritmov. Namesto tega smo časovne zahtevnosti konceptov predstavili opisno ter na konkretnih primerih.

Na začetku delavnice smo prvič pisno preverili znanje učencev. Preizkus znanja vsebuje štiri vprašanja, s katerimi smo želeli izvedeti, ali učenci razumejo pomen razvrščanja, ali poznajo več kot en koncept urejanja ter ali bi se z namigom znali spomniti novega koncepta urejanja. Nato smo izvedli samo delavnico. Kot je razvidno iz učne priprave, priložene v prilogi magistrske naloge, smo za izvedbo potrebovali učne pripomočke. Uporabili smo lončke s kovanci, s katerimi smo razložili štiri koncepte urejanja: urejanje z izbiranjem, urejanje z vstavljanjem, urejanje z zlivanjem ter hitro urejanje. Na koncu so učenci še enkrat rešili test.

Ime in priimek:

Preverjanje znanja

1. Ali misliš, da je pomembno, da imamo stvari urejene po vrsti? Zakaj?

2. Recimo da imaš v roki karte, kot so na spodnji sliki. Kako bi jih najhitreje uredil po velikosti? Poskusi natančno opisati svoje razmišljanje.

3. Ali bi znal karte iz slike urediti tudi na kakšen drugačen način? Če da, kateri način misliš, da je hitrejši?

4. Ali bi vam pomagalo, če bi karte iz slike razdelili na 2 dela in jih potem nekako uredili?

Vrednotenje rezultatov in delavnice

V tabeli 4.2 so prikazani rezultati preverjanj znanja pred delavnico in po njej, v tabeli 4.3 pa je navedena statistika rezultatov. Grafikon 4.1 predstavlja rezultate v grafični obliki.

Raziskovalno vprašanje: Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo urejanje?

McNemarjev test je pokazal, da se med vzorcema pojavljajo statistično pomembne razlike (p=0,0005). Iz rezultatov lahko ugotovimo, da so učenci preizkus znanja v povprečju reševali bolje po delavnici (M=9,0), kot pred njo (M=6,0)

Na koncu smo prišli do naslednjih ugotovitev:

1. Učna priprava je občutno preveč obsežna za popolno izvedbo v dveh šolskih urah. Zaradi tega smo se odločili, da sledimo nasvetu v viru [55] ter izpustimo razlago koncepta urejanje z vstavljanjem.
2. Iz analize preverjanj znanja je jasno, da so vprašanja preveč enostavna, zaradi česar nismo pridobili uporabnih podatkov o napredku učencev. Primer tega je drugo vprašanje, kjer je v obeh vzorcih 100 % učencev odgovorilo pravilno. Za nadaljnje delavnice smo se odločili, da bomo izdelali rahlo kompleksnejša vprašanja. Izkazalo se je tudi, da so se učenci najslabše odrezali pri zadnji nalogi. Razlog pripisujemo temu, da smo nalogo zastavili na dokaj nejasen način, brez nadaljnjih nasvetov, ki bi jih učenci potrebovali pri reševanju.

Delavnica 2: Zaporedje

Tudi tu smo v učni pripravi vnaprej spremenili razlago časovnih zahtevnosti, tokrat osnovnih operacij opisanih podatkovnih struktur. Primer: za operacijo dodajanje pri podatkovni strukturi vrsta smo namesto O(1) zapisali kar ≫takojšnja≪. Vse podatkovne strukture smo opisali z uporabo listkov in nalog, kot je zapisano v učni pripravi, priloženi v prilogi.

Pri izdelavi preverjanja znanja smo upoštevali ugotovitev s prejšnje delavnice, da morajo biti vprašanja nekoliko zapletenejša. S prvo nalogo smo želeli preveriti splošno znanje učencev, ali vedo kaj več kot ≫Zaporedje so elementi zapisani po vrsti.≪. Pri drugem vprašanju so učenci morali povezati teoretično znanje z vsakdanjimi izkušnjami, tretje vprašanje je povezalo dosedanje znanje z RIN, za zadnji del pa smo podali praktično nalogo.

Ime in priimek:

Preverjanje znanja

1. Ali veš kaj je zaporedje oziroma seznam? Napiši čim več kar veš o njem.

2. Ali lahko našteješ nekaj zaporedij, ki jih srečaš vsak dan? Ali je vrstni red stvari važen?

3. Ali si vedel/-a, da računalniki tudi uporabljajo zaporedja? Zakaj misliš da jih uporabljajo? Zakaj so dobra?

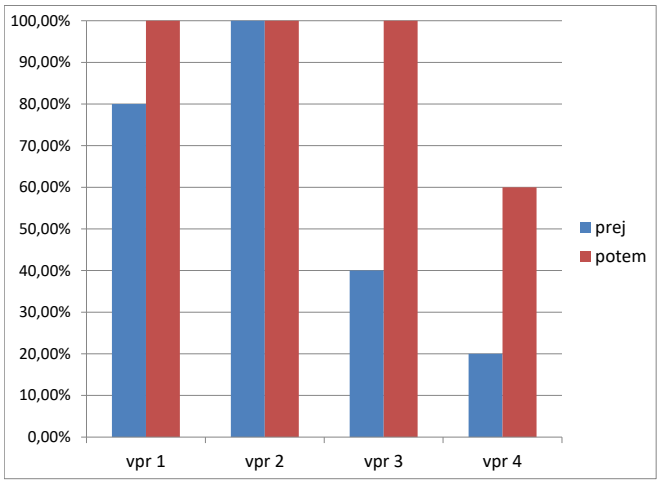
4. Recimo da imamo seznam stvari, ki jo moramo kupiti v marketu. V kakšnem vrstnem redu bi bilo najbolje si zapisati te stvari, da bi jih kupili kar se da hitro?

Tabela 4.2: Rezultati preverjanja znanja z delavnice urejanje.

	Ali je pomembno, da imamo stvari urejene po vrsti? Zakaj?	Kako bi najhitreje razporedil karte po velikosti?	Ali poznaš še kakšen način? Če ja, kateri način je hitrejši?	Kaj pa če bi karte s slike razdelili na 2 dela in jih potem nekako uredili?
Pred delavnico	8/10	10/10	4/10	2/10
Po delavnici	10/10	10/10	10/10	6/10

Tabela 4.3: Statistika preverjanja znanja z delavnice urejanje.

	Število (N)	Aritmetična sredina	Standardni odklon	Standardna napaka
Pred delavnico	10	6,0000	3,6515	1,1547
Po delavnici	10	9,0000	2,0000	0,6326



Slika 4.1: Grafični prikaz rezultatov preizkusa znanja z delavnice urejanje.

Vrednotenje rezultatov in delavnice

V tabeli 4.4 so prikazani rezultati učencev na preverjanju znanja pred delavnico in po njej, v tabeli 4.5 pa je navedena statistika rezultatov. Grafikon 4.2 predstavlja rezultate v grafični obliki.

Raziskovalno vprašanje: Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo zaporedje?

McNemarjev test je pokazal, da se med vzorcema pojavljajo statistično pomembne razlike (p=0,0002). Iz rezultatov lahko ugotovimo, da so učenci preizkus znanja v povprečju reševali bolje po delavnici (M=7,7) kot pred njo (M=4,5).

Na koncu smo prišli do naslednjih ugotovitev:

1. Vpeljava rahlo zapletenejših nalog se je izkazala za dobro, saj smo dobili vsaj na videz verodostojnejše podatke. Na podlagi izboljšave pravilnih odgovorov pred in po delavnici smo opazili, da so učenci v splošnem izboljšali svoje znanje. Opazimo pa slab rezultat in slabo izboljšavo pri zadnji nalogi. Naše mnenje je, da je to slabo zastavljena naloga in so jo zato učenci slabo reševali.

2. Razlaga snovi z uporabo listkov s hrano se je izkazala za zmedeno. Pri učencih smo namreč opazili, da niso zmogli povezati naloge in koncepta zaporedja, kakršno

Tabela 4.4: Rezultati preverjanja znanja z delavnice zaporedje.

	Kaj je zaporedje?	Naštejejte nekaj zaporedij. Ali je vrstni red stvari važen?	Ali si vedel, da računalniki tudi uporabljajo zaporedja? Zakaj misliš da je tako?	V kakšnem vrstnem redu bi bilo najbolje zapisati si hrano na listek, da bi jo nakupili kar se da hitro?
Pred delavnico	7/10	7/10	3/10	1/10
Po delavnici	10/10	8/10	9/10	4/10

je predstavljeno v računalniku. Zaradi tega smo učno pripravo prilagodili tako, da smo dodali razlago, kjer učitelj uporabi tablo za izris podatkovnih struktur (denimo, da za strukturo povezan seznam na tablo nariše elemente in jih poveže v vrsto), s katero bodo učenci bolje razumeli, kako so te strukture zapisane v računalniku.

4.2.4 Delavnica 3: Množica

Za delavnico smo se odločili, da se osredotočimo predvsem na razlago abstraktne podatkovne strukture množica ter na njeno implementacijo z dvojiškim iskalnim drevesom. Python set smo omenili zgolj za opis uporabnosti funkcij unija in presek preko krajše na-

Tabela 4.5: Statistika preverjanja znanja z delavnice zaporedje.

	Število (N)	Aritmetična sredina	Standardni odklon	Standardna napaka
Pred delavnico	10	4,5000	3,0000	0,9487
Po delavnici	10	7,7500	2,6300	0,8317

loge, pri implementaciji množice s tabelo pa smo se osredotočili na časovne zahtevnosti osnovnih operacij kot uvod v naslednji del, dvojiško drevo. Slednjemu smo, kot že rečeno, namenili več časa. Po razlagi strukture smo se spoznali z osnovnimi operacijami in njihovimi časovnimi zahtevnostmi ter pri tem znanje utrdili s številnimi nalogami, ki so jih učenci sprejeli zelo dobro.

Pri izdelavi preizkusa znanja smo se odločili za vprašanja osnovnega tipa: poznavanje pojma množica, primere zanj ter razlike med množico in zaporedjem. Zadnje vprašanje učenci izpolnijo le po delavnici, saj se neposredno nanaša na dvojiška drevesa, ki pa jih učenci pred delavnico zelo verjetno ne poznajo.

Ime in priimek:

Preverjanje znanja

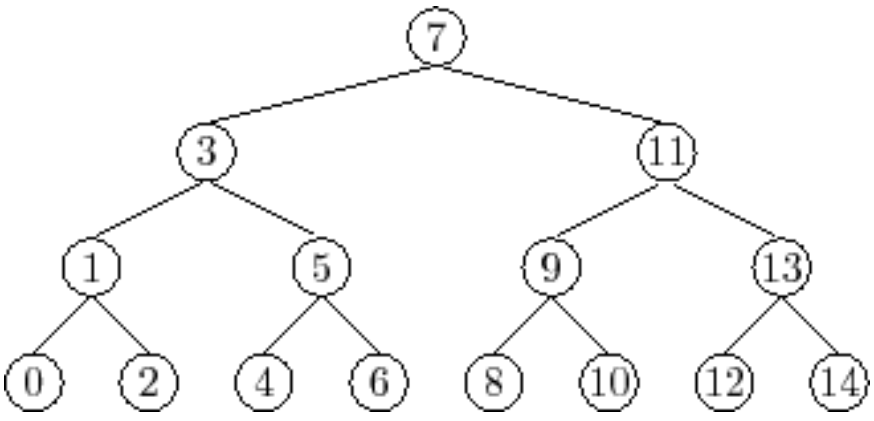
1. Kako bi definiral pojem množica? Kaj velja za elemente v njej?

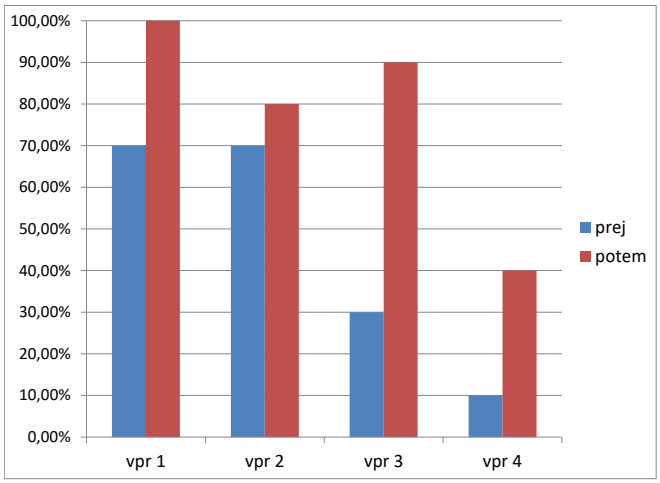
2. Kje se v vsakdanjem življenju srečamo z množicami? Naštejte nekaj primerov.

3. Razmislite in napišite kako se množica razlikuje od zaporedja.

4. (Odgovorite po delavnici) Ali se vam je zdela razlaga dvojiškega drevesa zanimiva, uporabna?

Za drevo na spodnji sliki napišite po kateri poti se moramo sprehoditi, da najdemo število 10.





Slika 4.2: Grafični prikaz rezultatov preizkusa znanja z delavnice zaporedje.

Vrednotenje rezultatov in delavnice

V tabeli 4.6 so prikazani rezultati učencev na preverjanju znanja pred delavnico in po njej, v tabeli 4.7 pa statistika rezultatov. Grafikon 4.3 predstavlja rezultate v grafični obliki.

Raziskovalno vprašanje: Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo množica?

McNemarjev test je pokazal, da se med vzorcema pojavljajo statistično pomembne razlike (p=0,0156). Iz rezultatov lahko ugotovimo, da so učenci preizkus znanja v povprečju reševali bolje po delavnici (M=7,2), kot pred njo (M=4,7).

Na koncu smo prišli do naslednjih ugotovitev:

1. Skoraj vsi učenci so na zadnje vprašanje odgovorili pozitivno. Dvojiško drevo se jim zdi zanimiva in/ali uporabna snov. Rezultat vprašanja v zgornji tabeli se nanaša zgolj na njihovo reševanje naloge in je zelo spodbuden. Zaradi tega smo se odločili učno pripravo popraviti tako, da smo dali razlagi dvojiškega drevesa še rahlo več poudarka.
2. Najslabše so se učenci odrezali pri tretjem vprašanju. Zanimiva je tudi korelacija prvega in tretjega vprašanja – če je več kot polovica učencev pred delavnico znala pravilno definirati pojem množice, pa so imeli težave pri razlikovanju le-te od pojma

Tabela 4.6: Rezultati preverjanja znanja z delavnice množica.

	Kako bi definiral množico? Kaj velja za njene elemente?	Kje se srečamo z množicami?	Kako se množica razlikuje od zaporedja?	Za spodnje drevo napišite, koliko skokov potrebujemo in po kateri poti se sprehodimo, da najdemo število 10.
Pred delavnico	5/9	8/9	1/9	/
Po delavnici	7/9	9/9	5/9	8/9

Tabela 4.7: Statistika preverjanja znanja z delavnice množica.

	Število (N)	Aritmetična sredina	Standardni odklon	Standardna napaka
Pred delavnico	9	4,6667	3,5119	1,1706
Po delavnici	9	7,2500	1,7078	0,5693

zaporedja. Eden od možnih odgovorov je, da učenci niso točno vedeli kaj zaporedje je (oziroma so pozabili, saj smo delavnico o zaporedju že izvedli), zaradi česar niso bili zmožni primerjave z množico. Vseeno smo v uvodnem delu učne priprave namenili več časa razlagi podatkovne strukture množica.

4.2.5 Delavnica 4: slovar

Delavnico smo izvedli kot smo jo opisali v podpoglavju 4.1.6 z izjemo, da smo več časa namenili razlagi razpršene tabele in reševanja sovpadanja elementov.

Po istem vrstnem redu kot je predstavljena snov, so zapisana tudi vprašanja v preizkusu znanja. V prvem vprašanju učence vprašamo po vsakdanjih primerih iskanja po-

datkov, pri čemer jim v pomoč v navodilih vprašanja pojasnimo en tak primer. Drugo vprašanje se navezuje na primere uporabe podatkovne strukture slovar. V zadnjem vprašanju nas zanima, kako učenci rešujejo problem sovpadanja elementov v strukturi slovar (tj. ko želimo dva različna elementa shraniti v isto polje v slovarju).

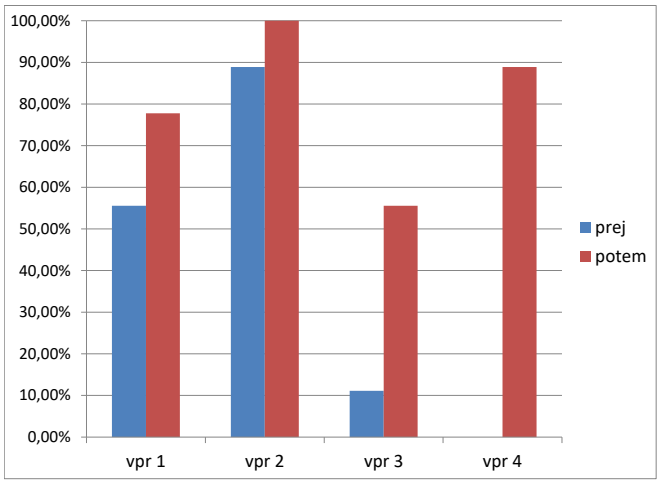
Ime in priimek:

Preverjanje znanja

1. V življenju se velikokrat spopademo s problemom iskanja podatkov. Primer: zdravnik splošne medicine hrani kartoteko vsakega pacienta. Da med njimi vsakič hitro najde željeno, je pametno, da jih ima urejene po imenu in priimku. Ali poznate še kakšen primer, kjer je pomembno hitro iskanje podatkov?

2. Slovar definiramo kot množico parov ključ - vrednost. Takšno je na primer kazalo v knjigi. Ključi so naslovi poglavij, vrednosti pa stran v knjigi, na kateri se začne poglavje. Se spomnite še kakšnega primera?

3. V računalniku lahko hranimo slovarje, pri čemer se včasih zgodi, da želi računalnik shraniti 2 vnosa na isto mesto v slovarju – temu pravimo kolizija. Ker ne moremo imeti 2 vnosov na istem mestu, razmislite in napišite, kako bi vi rešili ta problem. Na katero mesto bi zapisali drugi vnos?



Slika 4.3: Grafični prikaz rezultatov preizkusa znanja z delavnice množica.

Vrednotenje rezultatov in delavnice

V tabeli 4.8 so prikazani rezultati učencev na preverjanju znanja pred delavnico in po njej, v tabeli 4.9 pa statistika rezultatov. Grafikon 4.4 predstavlja rezultate v grafični obliki.

Raziskovalno vprašanje: Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo množica?

McNemarjev test je pokazal, da se med vzorcema pojavljajo statistično pomembne razlike (p=0,0078). Iz rezultatov lahko ugotovimo, da so učenci preizkus znanja v povprečju reševali bolje po delavnici (M=6,3), kot pred njo (M=3,7).

Na koncu smo prišli do naslednjih ugotovitev:

1. Izpostavili bi, da so bili učenci rahlo zmedeni, zakaj smo jih učili o delovanju razpršenih tabel, če pa za naše potrebe že obstaja slovar v jeziku Python. Tu pride do izraza eden od ciljev te magistrske naloge in projekta NAPOJ, in sicer da učimo računalniško mišljenje, ne zgolj uporabo tehnologij. Odgovorili smo, da je treba razumeti snov, saj jo lahko nato uporabimo nadalje v življenju, kar je tudi smisel šolanja.

2. Izvedbo delavnice ocenjujemo kot uspešno, saj je potekala gladko in brez večjih zapletov, na kar kažejo tudi rezultati preizkusa znanja, posebno tretjega vprašanja.

Tabela 4.8: Rezultati preverjanja znanja z delavnice slovar.

	Ali poznate še kak primer, kjer je potrebno hitro iskanje podatkov?	Slovar definiramo kot množico parov ključ-vrednost. Ali poleg kazala v knjigi poznate še kakšen primer slovarja	Kolizija je, ko računalnik želi shraniti 2 podatka v isto celico v tabeli. Kako bi vi rešili ta problem?
Pred delavnico	6/8	3/8	2/8
Po delavnici	7/8	5/8	7/8

Tabela 4.9: Statistika preverjanja znanja z delavnice slovar.

	Število (N)	Aritmetična sredina	Standardni odklon	Standardna napaka
Pred delavnico	8	3,6667	2,0817	0,6939
Po delavnici	8	6,3333	1,1547	0,3849

Če jih je pred delavnico le peščica znala odgovoriti (odgovori so bili recimo »na prvo prosto mesto v slovarju« in »na naslednje prosto mesto«), jih je po delavnici 90 % znalo odgovoriti bodisi z odprtim bodisi zaprtim razprševanjem, nekaj jih je pa tudi napisalo oba načina. Tako učne priprave nismo spreminjali.

4.2.6 Delavnica 5: Osnovni pojmi algoritmike

Tako kot je opisano v učni pripravi tega sklopa, smo tudi izvedli delavnico. Na začetku smo podali nalogo, skozi katero smo spoznali postopek reševanja z modelom abstrakcije. Ta sestoji iz razumevanja in analize problema (sem spada tudi razumevanje in uporaba abstrakcije) ter izdelave ustrezne rešitve. Med razlago smo spoznali pojme algoritem,

programerski problem, psevdokoda, pravilnost algoritma in ustavljivost algoritma.

Pripravljeni preizkus znanja sestavljajo štiri vprašanja, pri čemer morajo učenci pri prvi nalogi poiskati skupni problem podanih nalog, drugo vprašanje sprašuje po osebnih navadah učencev glede pristopa do problemov, pri čemer smo s predzadnjim podvprašanjem želeli nakazati, da je za reševanje problema najprej bolje imeti plan, šele nato ga rešimo. Tretje vprašanje se nanaša na prejšnje. Tu smo želeli analizirati, na kakšen način se učenci v praksi lotijo reševanja naloge. Zadnje vprašanje velja kot ≫bonus≪, saj nas bolj zanima, kako se bodo pri njem učenci odrezali po delavnici.

Ime in priimek:

Preverjanje znanja

1. Naslednje 3 naloge se nanašajo na isti problem. Ga lahko zapišete?

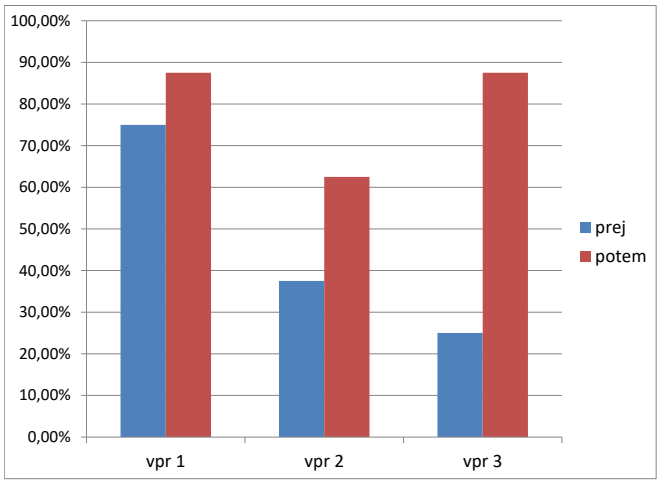
1) Iz škatle smo izvlekli kroglice s številkami 25, 20 in 17. Katera od teh številk je najmanjša?
2) Trije prijatelji se med seboj primerjajo po starosti. Prvi je rojen leta 1999, drugi 1998, tretji pa 2001. Katerega leta je bil rojen najmlajši med njimi?
3) Dijaki tretjih letnikov gimnazije Tecigrad so tekmovali v teku na 1000 metrov. Dijaki razreda A so navedeno razdaljo v povprečju pretekli v 230 sekundah, dijaki razreda B v 260 sekundah, dijaki razreda C pa v 250 sekundah. Kolikšen povprečni čas je dosegel najhitrejši razred?

2. Opišite, kako se lotite reševanja problema, na primer učenja za pisni preizkus znanja. Ali se je reševanja problema pametno lotiti s takojšnjim delom na njem, ali je bolje začeti s načrtovanjem, kako bomo problem rešili? Zakaj?

3. Recimo, da ste programerji in imate podano spodnjo nalogo. Kako bi se lotili reševanja te naloge? Bi se usedli za računalnik in takoj začeli programirati? Ali bi najprej naredili plan?

1) Za spletni iskalnik Google Chrome sprogramirajte delovanje gumba »nazaj«, da nam bo ob kliku na njega odprl zadnjo obiskano spletno stran.

4. Ali lahko razložite kaj pomeni abstrakcija?



Slika 4.4: Grafični prikaz rezultatov preizkusa znanja z delavnice slovar.

Vrednotenje rezultatov in delavnice

V tabeli 4.10 so prikazani rezultati učencev na preverjanju znanja pred delavnico in po njej, v tabeli 4.11 pa statistika rezultatov. Grafikon 4.5 predstavlja rezultate v grafični obliki.

Raziskovalno vprašanje: Ali se v skupini pojavljajo statistično pomembne razlike v preverjanju znanja pred in po delavnici na učno temo osnovni pojmi algoritmike?

McNemarjev test je pokazal, da se med vzorcema pojavljajo statistično pomembne razlike (p=0,0001). Iz rezultatov lahko ugotovimo, da so učenci preizkus znanja v povprečju reševali bolje po delavnici (M=7,3), kot pred njo (M=3,8).

Na koncu smo prišli do naslednjih ugotovitev:

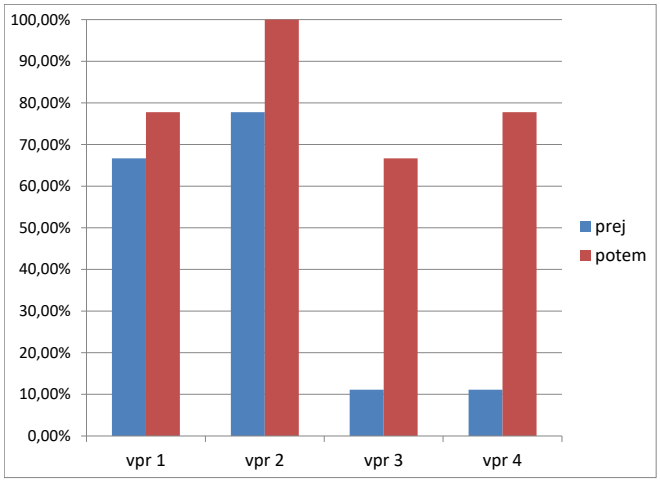
1. Analiza preverjanja znanja kaže zanimive rezultate. Pri prvem vprašanju se število učencev s pravilnim odgovorom ni bistveno izboljšalo. Tretje in četrto vprašanje kaže na velik napredek učencev, s čimer smo zelo zadovoljni.
2. Zaradi slabše izboljšave rezultatov prvega vprašanja iz preizkusa znanja smo se odločili, da bomo učno pripravo izboljšali tako, da bomo bolj poudarili razumevanje pojmov problem in primerek problema.

Tabela 4.10: Rezultati preverjanja znanja z delavnice osnovni pojmi algoritmike.

	Poiščite skupni problem podanim nalogam.	Opišite, kako se lotite reševanja problema.	Kako bi se kot programerji lotili reševanja podane naloge?	Ali lahko razložite kaj pomeni abstrakcija?
Pred delavnico	6/9	7/9	1/9	1/9
Po delavnici	7/9	9/9	6/9	7/9

Tabela 4.11: Statistika preverjanja znanja z delavnice osnovni pojmi algoritmike.

	Število (N)	Aritmetična sredina	Standardni odklon	Standardna napaka
Pred delavnico	9	3,7500	3,2016	1,0672
Po delavnici	9	7,2500	1,2583	0,4194



Slika 4.5: Grafični prikaz rezultatov preizkusa znanja z delavnice osnovni pojmi algoritmike.

Poglavje 5

Sklepne ugotovitve

Cilj magistrskega dela je izdelava učnih priprav in programskih nalog v storitvi TOMO za predmet informatika, ki se izvaja na slovenskih splošnih, klasičnih in strokovnih gimnazijah, ter ovrednotenje učnih priprav. Začeli smo z raziskovanjem stanja poučevanja informatike v Sloveniji in tujini ter ju primerjali. Ugotovljeno je bilo, da je stanje v Sloveniji bistveno slabše kot v tujini. Nadaljevali smo s pripravo na izdelavo učnih priprav. Spoznali smo, kako se uči računalniško mišljenje, izbrali ustrezno programsko okolje za poučevanje algoritmov, določili vrstni red sklopov oziroma učnih tem za poučevanje ter določili strukturo učne priprave. Sledila sta izdelava učnih priprav, za katere smo opisali postopek izdelave vsake od njih, ter ovrednotenje priprav z izvedbo petih delavnic, od katerih je vsaka namenjena eni učni enoti. Pri ovrednotenju smo si pomagali s pisnim preverjanjem znanja učencev, pri čemer smo preverili njihovo znanje pred in po vsaki delavnici. Z ovrednotenjem rezultatov smo ugotovili uspešnost izpeljave vsake delavnice ter po potrebi izboljšali učne priprave. Na koncu smo učne priprave naložili na spletno učilnico projekta NAPOJ, kjer so na voljo učiteljem informatike v Sloveniji.

5.1 Didaktični prispevki

Zaradi ohlapno definiranih učnih ciljev v učnem načrtu je poučevanje informatike v Sloveniji dokaj nekonsistentno in nekonkurenčno tujim državam. Izdelali smo učne priprave, ki bodo na voljo vsem učiteljem informatike v Sloveniji. Ti jih bodo lahko uporabili v svojih učnih urah ter s tem pripomogli k zvišanju ravni znanja računalništva in informatike, s čimer bo Slovenija na omenjenem področju v svetu postala kompetentnejša.

5.2 Predlogi izboljšav

Učne priprave bi lahko večkrat ovrednotili na večjem vzorcu učencev, s čimer bi jih še dodatno izboljšali. Tudi preizkusi znanja, ki smo jih uporabili na delavnicah, bi bili lahko izdelani z večjo pozornostjo, saj se je po pregledu rezultatov v nekaj primerih izkazalo, da so bila določena vprašanja slabo zastavljena in iz njih nismo pridobili smiselnih ugotovitev.

Učne priprave bi bile lahko ovrednotene tudi na drugačne načine, na primer, da bi jih pregledali učitelji oziroma profesorji informatike, ki imajo več izkušenj pri izdelavi učnih priprav in poučevanju predmeta. Z razvijanjem področja računalništva in informatike ter načina poučevanja pa je mogoče učne priprave venomer izpopolnjevati in nadgrajevati.

Z izdelavo magistrske naloge smo pridobili nova znanja s področja računalništva in informatike, vpogled v stanje izobraževanja v Sloveniji in svetu, spoznali smo se z izdelavo učnih priprav in njihovim ovrednotenjem ter nenazadnje pridobili praktične izkušnje glede samega poučevanje informatike v razredu. Pridobljena znanja bodo prispevala k nadaljnjemu izobraževanju in udejstvovanju avtorja magistrske naloge na področju izobraževanja v Sloveniji.

Dodatek A

Rezultati anketiranja

Odgovori	**Kakšno je vaše mnenje glede predmeta informatika?**
Odgovor 1	Na trenutke brezvezno, a najbrz uporabno.
Odgovor 2	Zdi se mi, da bi bilo logično v prvem letniku predstaviti HTML, s katerim bi se v enem šolskem letu naučili ustvariti spletno stran. Python pa v drugem letniku za tiste, ki jih to zanima in so se pripravljeni poglobiti v programiranje. Povečini pa se mi snov ni zdela prezahtevna. Ocenjevanje se mi je zdelo pravično. Najbolj neprijeten spomin mi je ostal v povezavi z skupinsko projektno nalogo, pa ne zaradi skupine, ampak zaradi kriterijev, ki jih je bilo težko razumeti in izpolniti. Glede na to, da je bil najverjetneje za večino to prvi tovrsten projekt, bi bil vesel, če bi mu posvetili več pozornosti. Razumljivo mi je, da smo iz prostorskih razlogov delovali v dveh učilnicah, to pa je imelo stranski učinek, da profesorja nista bila usklajena in je pogosto prihajalo do zmede. Posebno pri tistih, ki informatike nismo vešči iz osnovne šole. Všeč pa mi je bilo, da snovi nismo jemali na slepo iz učbenika. Ideja o razlagi na spletni učilnici se je izkazala za zelo uporabno.
Odgovor 3	Moje osebno mnenje je, da je informatika predmet oziroma področje, ki me ne zanima preveč, a je verjetno zelo dobro da se jo naučimo, saj na tem temelji današnja znanost in tako rekoč tudi svet.
Nadaljevanje na naslednji strani.	

Tabela A.1 – nadaljevanje

Odgovori	**Kakšno je vaše mnenje glede predmeta informatika?**
Odgovor 4	Informatika mi je bila kar všeč, saj ni bila preveč »zategnjena«. Predvsem na začetku leta mi je bilo zanimivo programiranje. Morda je bilo nekatero snov težje razumeti (predvsem za tiste, ki se s tem spopadamo prvič) in bi bila potrebna ponovna razlaga. Všeč mi je bil način pisanja testov, vendar mislim, da bi bilo bolje programe pisati na računalnik. Všeč mi je bilo tudi to, da smo lahko imeli s seboj zapiske in smo se tako izognili nepotrebnemu učenju na pamet. Malo manj so mi bile všeč projektne naloge, saj se mi je zdel predvsem praktičen del prezahteven za tiste, ki nimamo izkušnje s programiranjem. Pametno se mi zdi, da smo domače naloge oddajali na ta način, saj smo bili tako prisiljeni delati sproti. Rahlo nesmiselne so se mi sicer zdele tiste proti koncu šolskega leta, ko je bilo zahtevano neko teoretično znanje in iskanje manj pomembnih podatkov po internetu (npr. sestava IP paketka).
Odgovor 5	Pouk informatike se mi je zdel zelo zanimiv in praktičen. Naučili smo se nekaj osnovnih veščin, ki pridejo prav pri vseh predmetih, (pisanje wordovih dokumentov, powerpointov, excel tabele, pisanje projektne naloge itd.) in tudi bolj specifične veščine (npr. programeranje).
Odgovor 6	Informatika se mi zdi zelo zanimiv predmet, hkrati pa tudi zelo uporaben tudi naprej v življenju. Vseeno pa se mi zdi, da smo v tem letu snov jemali malo prehitro in je nismo sproti utrjevali, zato se je marsikomu predmet zameril oz. je dobil odpor do njega. Svetovala bi, da bi učencem nerazumljive domače naloge pri pouku še enkrat razložili, ker je bilo včasih zelo težko slediti zaradi povsem novih vsebin, s katerimi se nas večina prej v življenju sploh še ni srečala. Komentar imam tudi na delo v skupini pri projektnih nalogah, in sicer se bi mi zdelo ustrezneje, če bi pri ocenah bolj upoštevali dejansko delo posameznika v skupini, saj se je v velikih primerih, tudi v mojem, zgodilo, da je nekdo naredil skoraj vse, drugi pa nič, potem pa so bili vseeno enako ocenjeni. Čeprav je to skupinsko delo, je včasih zelo težko vplivati na druge in na koncu vse skupaj ni pošteno.
Nadaljevanje na naslednji strani.	

Tabela A.1 – nadaljevanje

Odgovori	**Kakšno je vaše mnenje glede predmeta informatika?**
Odgovor 7	Predmet informatika mi je zelo všeč. Moti me samo razlaga oziroma učenje programiranja, saj bi si po mojem mnenju morali vzeti več časa, tako da bi vsi znali programirati in razumeli do potankosti kako se programira in kako nek programski jezik deluje. Kljub temu mi je predmet všeč.
Odgovor 8	*čudno se mi zdi, ker profesor Brodnik vika *ne zdi se mi pravično, da obvezno dodatno (kazensko) nalogo dobimo vsi, čeprav jih je klepetalo le nekaj *zelo mi je všeč, da imamo pri testih lahko zapiske, saj se tako nihče ne 'pifla' temveč se učimo z razumevanjem *všeč so mi dodatne naloge pri testu in domačih nalogah, saj predstavljajo poseben izziv
Odgovor 9	Pouk se mi je zdel zanimiv in kar nekaj smo se naučili. V bodoče si želim tudi samostojnega dela pri seminarski nalogi, projektni nalogi, ...ker se lahko kot posameznik dokažeš s svojim znanjem in dobiš oceno, ki si si jo sam zaslužil. Če pa je skupinsko delo pa se vedno nekdo ≫šlepa≪ in dobi enako ali nekaj manjšo oceno od tistega , ki res dela ali pa mu to celo dvigne končno oceno. Skupinsko delo je dobro za povezovanje med nami in za različne poglede na nalogo, če si člani skupine delo res razdelijo in delajo enakovredno.Pa mogoče še to, če so datumi in roki za oddajo nalog jasni in napovedani se ne bi smelo po roku oddajati nalog. Najbrž ste ugotovili, da so posamezniki oddajali domače naloge od drugih z manjšimi popravki ali celo brez njih.
Odgovor 10	Ocene naj bi zaključevali skupaj in tako bi lahko takoj dobili odgovore zakaj taka in ne drugačna ocena. Spraševanje po meilu o tem se mi ne zdi ok. Mogoče bi lahko domače naloge popravljali boj sproti, da bi imeli možnost dopolniti in pojasniti tisto kar je bilo pomanjkljivo.
Odgovor 11	Pri predstavitvi projektne naloge bi bilo mogoče dobro, da bi dobili sprotne informacije kaj je bilo pomanjkljivo narejeno, kaj dobro, katere prosojnice so ok, kako se je točkovalo, ...sam še danes ne vem kaj je bilo narobe z našimi prosojnicami?
Odgovor 12	Meni je bila informatika všeč, vendar so se mi zdele občasno domače naloge prezahtevne v primerjavi z razlago, ki smo jo dobili. Velikokrat se mi je zdelo, da pri pouku nismo še tega povedali.

Dodatek B

Struktura učne priprave

NAPOJ

UČNA PRIPRAVA

NASLOV UČNE PRIPRAVE

DATUM NAPOJ

OSNOVNI PODATKI

Šola:
Letnik:
Datum:
Predmet:
Učna tema:
Učna enota:
Učne oblike: -
Učne metode: -
Predznanje: -
Operativni učni cilji Ob koncu učne ure učenec: •
Učna sredstva: -
Didaktične etape učnega procesa: 1.
Medpredmetne povezave:
Literatura: -
Novi pojmi: -
Priloga: -

POTEK UČNE URE

UVODNI DEL: UVAJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI

GLAVNI DEL: OBRAVNAVANJE UČNE SNOVI / SPROTNO PREVERJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI

ZAKLJUČNI DEL: ZAKLJUČNO PONAVLJANJE / PREVERJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI

Priloge

Dodatek C

Učna priprava urejanje

Ostale učne priprave se nahajajo na spletnem naslovu `https://redmine.lusy.fri.uni-lj.si/documents/276`.

NAPOJ

UČNA PRIPRAVA

Urejanje

DATUM NAPOJ

OSNOVNI PODATKI

Šola:

Letnik: 1. Letnik

Datum:

Predmet: Informatika

Učna tema: Urejanje podatkov

Učna enota: urejanje, koncepti urejanja

Učne oblike:

- frontalno, individualno, skupinsko

Učne metode:

- razlaga, pogovor, demonstracija, reševanje problemov

Predznanje:

- poznajo pojme algoritem, funkcija, tabela, spremenljivka

Operativni učni cilji

Ob koncu učne ure učenec:

- Spozna problem urejanja podatkov
- Razume logično oziroma strukturno razmišljanje v proces urejanja
- Spozna različne koncepte urejanja
- Razume dobre in slabe lastnosti teh konceptov

Učna sredstva:

- Učila: delovni list, računalnik
- Učni pripomočki: računalnik, Python, zvezek, tabla, tehtnica z različno težkimi utežmi oziroma podobno orodje

Didaktične etape učnega procesa:

1. Pripravljanje ali uvajanje
2. Obravnava nove učne snovi ali usvajanje
3. Urjenje
4. Ponavljanje
5. Preverjanje

Medpredmetne povezave: slovenščina, angleščina

Literatura:

- M. A. Weiss, Data Structures and Algorithm Analysis (2nd Edition), Addison Wesley, 1994.
- M. T. Goodrich, R. Tamassia, Data structures and algorithms in Java, 2001.
- G. Anželj, J. Brank, A. Brodnik, P. Bulić, M. Ciglarič, M. Đukić, L. Furst, M. Kikelj, A. Krapež, H. Medvešek, N. Mori, M. Pančur, P. Sterle, Računalništvo in informatika 1, Založba Univerze na Primorskem and Založba Fakultete za računalništvo in informatiko and Založba Fakultete za elektrotehniko and računalništvo in informatiko, 2015. URL https://lusy.fri.uni-lj.si/ucbenik/book/index.html.
- B. Vocking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, D. Wagner, Algorithms unplugged, Springer Science & Business Media, 2010.

- Computer Science Unplugged, Lightest and Heaviest – Sorting Algorithms. Dostopno na: https://classic.csunplugged.org/wp-content/uploads/2014/12/unplugged-07-sorting_algorithms.pdf (pridobljeno 2. 8. 2018).
- E. Knuth D, The art of computer programming, fundamental algorithms (1980).
- D. E. Knuth, The art of computer programming: sorting and searching, Vol. 3, Pearson Education, 1997.
- Tutorials point, Data structures - merge sort algorithm. Dostopno na: https://www.tutorialspoint.com/data_structures_algorithms/merge_sort_algorithm.htm (pridobljeno 2. 8. 2018)
- Leder, Misha, Presentation 4: sorting. Dostopno na: https://sites.google.com/site/childrenandtechnology/Home/presentation-4-sorting (pridobljeno 2. 8. 2018).
- J. Demšar, Urejanje, Dostopno na: http://vidra.si/urejanje, (pridobljeno 3. 8. 2018).

Novi pojmi:
- urejanje z izbiranjem, urejanje z zlivanjem, hitro urejanje

Priloga:
- Tabela z rezultati

POTEK UČNE URE

UVODNI DEL: UVAJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Uvod 2 min	*Učitelj pozdravi učence, zapiše manjkajoče učence in po potrebi prosi dežurnega učenca, da pobriše tablo.*	Odzdravijo, naštejejo manjkajoče učence, dežurni učenec pobriše tablo.	Frontalno.
Motivacija 10 min	Danes se bomo posvetili urejanju. Urejanje je problem, pri katerem hočemo elemente nekega zaporedja prerazporediti v željen vrstni red. Večina ljudi imajo radi stvari urejene oziroma pospravljene. Mi lahko naštejete nekaj takih primerov in razložite, zakaj je v teh primerih dobro imeti stvari urejene? Na primer: • Osebe v imeniku so urejene po abecedi, da hitreje najdemo želeno. • Pri prenosu olimpijskih iger se na koncu teka izpišejo imena tekmovalcev, ki so urejeni po času, da gledalci lažje vidimo kdo je zmagal. • Knjige na polici so urejene ali po naslovu,	Poslušajo. Odgovorijo, naštejejo primere.	Frontalno, razlaga, pogovor.

	ali po pisatelju, spet z namenom da hitreje najdemo želeno. Naslednje vprašanje za vas je, po katerih kriterijih lahko urejamo elemente? Ali je primerjanje števil edini možen kriterij? Drugi možni kriteriji so na primer še: • Odtenek barve • Oblika predmetov • Veselje (nekateri čustvenčki so bolj, nekateri manj veseli) • Moč zvoka (ali je električna kitara glasnejša od orglic) • Mehkosti predmeta Pojdimo raje iz druge smeri: naštejte nekaj stvari, ki jih ne moremo urejati. *Učitelj odgovarja na primere učencev, na kakšen način se jih da urediti.* Vidimo torej, da je veliko več kriterij za urejanje kot pa le po velikosti števil ter da skoraj ni stvari, katere se ne bi dalo urediti glede na nek kriterij.	Odgovorijo, naštejejo druge kriterije. Poskušajo našteti stvari, ki se jih ne da urediti po nekem kriteriju.	

GLAVNI DEL: OBRAVNAVANJE UČNE SNOVI

VSEBINSKI POUDARKI	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Uvod v urejanje 15 min	Danes bomo spoznali 3 koncepte urejanja. Še prej pa sem pripravil pripomočke za današnjo uro. Z njimi se bomo tudi spoznali s temi koncepti. Vidimo, da imam tehtnice in lončke, ki so različno težki. *Opomba učitelju: izbira ustreznih učnih pripomočkov (v tej učni pripravi so uporabljeni tehtnica in različno težki lončki (lahko bi uporabili namesto tehtnice kar roke)) je prepuščena učitelju. Važno je, da učenci ne morejo urediti elementov z metodo ostrega pogleda, saj se s tem izgubi razumevanje, kako računalnik primerja elemente.* Vaša naloga je, da se razporedite v skupine. Vsaka skupina bo dobila eno tehtnico in nekaj različno težkih lončkov. Eden iz skupine jih bo moral urediti glede na težo od najlažjega do najtežjega, drug učenec iz skupine bo pa zapisoval število primerjanj. Naj vas opozorim, da lahko hkrati primerjate zgolj 2 lončka, saj boste tako bolje kako	Spremljajo razlago, si zapisujejo. Učenci se razdelijo v skupine in uredijo lončke po teži.	Frontalno, razlaga, skupinsko delo.

	deluje računalnik. Pa kar začnite. *Učitelj medtem na tablo nariše tabelo iz priloge 1.* Prosim, da mi sedaj vsaka skupina pove, koliko primerjanj je potrebovala za ureditev lončkov. *Učitelj zapiše številke v tabelo pod »prosto urejanje«.* Čestitke najboljši ekipi! Nam lahko zaupate, po kakšnem sistemu ste karte uredili? Ste imeli težave z izbiranjem sistema ali ste jih uredili intuitivno? Kaj pa ostale skupine, ste uredili karte na isti način? Kot vidimo je izbira dobrega načina urejanja oziroma algoritma pomembna. Za urejanje lahko izbiramo med različnimi algoritmi, pri čemer se ti razlikujejo po preprostosti implementacije in hitrosti urejanja. Danes bomo spoznali enega enostavnejša, a počasnejšega algoritma – urejanje z izbiranjem, ter 2 hitrejša, a bolj zapletena – urejanje z zlivanjem in hitro urejanje. Uro bomo nadaljevali tako, da bom za vsak algoritem razložil njegovo delovanje, vsaka skupina pa bo z uporabo teh algoritmov uredila lončke po teži, pri čemer boste seveda prešteli število primerjav. Te bom nato napisal na tablo, da bomo na koncu videli, kateri algoritmi so boljši in kateri slabši. Boljši so seveda tisti z manjšim številom primerjanj. Vsak algoritem bomo tudi implementirali v Pythonu ter poiskali časovno zahtevnost.	Skupine sporočijo število primerjanj. Učenci odgovorijo glede na lastno izkušnjo. Učenci poslušajo in si zapisujejo.	
Urejanje z izbiranjem 13 minut	Začnimo z urejanjem z izbiranjem. Ta algoritem je zelo enostaven: vsak obhod elementov poiščemo najmanjšega oziroma najlažjega in ga postavimo na začetek vrste. Kako? Vzamemo prvi lonček in ga primerjamo z naslednjim. Če je naslednji lažji, je sedaj ta najmanjši in njega primerjamo z nadaljnjimi lončki. Ko pridemo do konca, ta lonček postavimo na začetek vrste in ponovimo postopek z naslednjim lončkom v vrsti. Vzemimo na primer zaporedje [5,8,7,2,3,10,9]. Postopek urejanja je sledeč: [2,5,8,7,3,10,9] Se sprehodimo čez elemente, najdemo najmanjšega, tj. 2, ter ga postavimo na začetek. [2,3,5,8,7,10,9] najmanjše je 3, jo postavimo na indeks 1 [2,3,5,7,8,10,9] najmanjše je 5, jo postavimo na indeks 2	Spremljajo razlago in si zapišejo v zvezek.	Frontalno, razlaga, pogovor, skupinsko, demonstracija.

	[2,3,5,7,8,9,10] najmanjše je 7, jo postavimo na indeks 3 [2,3,5,7,8,9,10] najmanjše je 8, jo postavimo na indeks 4 [2,3,5,7,8,9,10] najmanjše je 9, jo postavimo na indeks 5 [2,3,5,7,8,9,10] najmanjše je 10, jo postavimo na indeks 6 Sedaj je na vas, da uredite lončke, in ne pozabite preštet število primerjanj. Sedaj vas prosim, da mi sporočite, koliko primerjanj ste dobili. *Učitelj izpolni v tabeli stolpec Urejanje z izbiranjem.* Kaj mislite, zakaj ste vsi dobili enako število primerjav? Tako, ker ob vsakem obhodu vedno obiščemo vse karte. Zapišimo algoritem v Pythonu. *Učitelj razloži kodo.* `def uredi(sez):` `  for i in range(len(sez)):` `  iMin = i` `  for j in range(i+1, len(sez)):` `  if sez[j] < sez[iMin]:` `  iMin = j` `  sez[i], sez[iMin] = sez[iMin], sez[i]` `  return sez`	Skupine uredijo lončke z uporabo urejanja z izbiranjem. Skupine sporočijo število primerjanj in odgovorijo na vprašanja.	
Urejanje z zlivanjem 18 min	Pojdimo na urejanje z zlivanjem. Tu pridemo do zanimivega koncepta, ki se imenuje deli in vladaj. Pri tem razdelimo problem na dva podproblema, ju rešimo, ter ju združimo v rešitev prvotnega problema. Urejanje z zlivanjem poteka takole: • 1. del (deli): razdeli lončke na dva enaka dela, vsako del tudi na pol, vsako četrtino tudi na pol, ... dokler ne dobimo, da je vsak del velik po 0 ali 1 element. • 2. del (vladaj): združujemo po dva dela tako, da so v novem seznamu lončki urejeni. Za lažje razumevanje si na primeru. Imamo seznam [7,4,5,2]. Najprej ga razdelimo na pol, vsak del pa še na pol. Tako dobimo sezname [7], [4], [5] in [2]. Nato združimo prva dva seznama tako, da primerjamo prvi element iz obeh seznamov ter manjšega prenesemo v združen seznam. To delamo, dokler imamo še kaj elementov v prvem	Sledijo razlagi in si zapisujejo.	Frontalno, razlaga, pogovor, skupinsko, demonstracija.

ali drugem seznamu.

Sez1	Sez2	združenSez
[7]	[4]	[]
[7]	[]	[4]
[]	[]	[4,7]

Enako storimo z naslednjima dvema podseznama.

Sez1	Sez2	združenSez
[5]	[2]	[]
[5]	[]	[2]
[]	[]	[5,2]

Na koncu združimo še ta dva seznama.

Sez1	Sez2	združenSez
[4,7]	[5,2]	[]
[4,7]	[5]	[2]
[7]	[5]	[2,4]
[7]	[]	[2,4,5]
[]	[]	[2,4,5,7]

Uredite sedaj lončke z uporabo urejanja z zlivanjem, pa ne pozabite prešteti število primerjanj.

Skupine uredijo lončke z uporab urejanja z zlivanjem.

Sedaj vas prosim, da mi poveste koliko primerjanj ste dobili. Kaj mislite, zakaj ste dobili enako število primerjav?

Skupine sporočijo število primerjanj in odgovorijo na vprašanja.

Sedaj zapišimo algoritem v Pythonu.

```
def zlijZaporedji(sez1, sez2):
  zlito = []
  i = 0
  j = 0

  while i < len(sez1) or j <
len(sez2):
    if j >= len(sez2):
      zlito.append(sez1[i])
```

	``
        i += 1
    elif i >= len(sez1):
        zlito.append(sez2[j])
        j += 1
    elif sez1[i] < sez2[j]:
        zlito.append(sez1[i])
        i += 1
    else:
        zlito.append(sez2[j])
        j += 1
  return zlito

def UrejanjeZZlivanjem(sez):
  n = len(sez)
  if n <= 1:
      return sez

  levo = sez[:n // 2]
  desno = sez[n // 2:]

  levoUrejeno =
UrejanjeZZlivanjem(levo)
  desnoUrejeno =
UrejanjeZZlivanjem(desno)
  zdruzeno =
zlijZaporedji(levoUrejeno,
desnoUrejeno)
  return zdruzeno
```<br><br>*Učitelj razloži kodo.*<br><br>Funkcija UrejanjZZlivanjem() razdeli prejet seznam na podsezname ter kliče funkcijo zlijZaporedji() in mu poda 2 taka podseznama, funkcija pa ju združi. | | |
| **Hitro urejanje**<br><br>17 min | Pojdimo še na hitro urejanje oziroma po angleško Quicksort. Tudi ta algoritem deluje po načinu deli in vladaj, a na malo drugačen način. Na začetku vzamemo prvi element, ki ga imenujemo pivot. Nato gremo čez vse ostale elemente in jih razporedimo v nove sezname glede na to, ali so manjši ali večji od pivota. Če je element manjši, gre v prvi seznam (levo od pivota), če večji, pa v drugega (desno od pivota). Postopke ponovimo z obema kupoma. Primer:<br><br>[4,2,5,7,3,1,8] vzamemo 4 za pivot, v levi kup postavimo manjše, v desni pa večje elemente<br><br>[2,3,1], [4], [5,7,8] iz levega kupa vzamemo 2 za pivot, iz desnega pa 5<br><br>[1], [2], [3], [4], [], [5], [7,8] iz skrajno desnega vzamemo za pivot 7<br><br>[1], [2], [3], [4], [], [5], [7], [8] združimo sezname<br><br>[1,2,3,4,5,7,8] rešitev | Sledijo razlagi in si zapisujejo. | Frontalno, razlaga, pogovor, skupinsko, demonstracija. |
```

	Pa kar začnite z urejanjem in ne pozabite prešteti število primerjanj. Sedaj vas prosim, da mi sporočite koliko primerjanj ste dobili. Zapišimo algoritem v obliki psevdokode: `def hitro_uredi(sez):` `  manjsi = []` `  enaki = []` `  vecji = []` `  if len(sez) > 1:` `  pivot = sez[0]` `  for st in sez:` `  if st < pivot:` `  manjsi.append(st)` `  elif st == pivot:` `  enaki.append(st)` `  else:` `  vecji.append(st)` `  leviSez = hitro_uredi(manjsi)` `  desniSez = hitro_uredi(vecji)` `  return leviSez + enaki + desniSez` `  else:` `  return sez` *Učitelj razloži kodo. F*unkcija hitro_uredi() razdeli elemente prejetega seznama v 3 sezname ter uredi levi in desni seznam.	Skupine uredijo lončke z uporabo urejanja z zlivanjem. Skupine sporočijo število primerjanj in odgovorijo na vprašanja.	
Časovna zahtevnost 13 min	Kot ste opazili, je urejanje z izbiranjem enostavnejše kot urejanje z zlivanje ali hitro urejanje. Kaj pa glede časovne zahtevnosti? Kateri algoritem je najhitrejši? Mogoče nam je že iz števila primerjanj jasno, a to ne nujno kaže realne slike, zato uporabimo naše znanje iz poglavja Poraba časa in časovna zahtevnost. Če pogledamo kodo urejanja z izbiranjem, vidimo, da imamo for zanko, znotraj nje pa še eno for zanko, pri čemer gresta obe preko vseh elementov seznama. Kakšna je torej časovna zahtevnost algoritma? Pri urejanju z zlivanjem moramo malo razmisliti. Funkcija vsakič kliče samo sebe, a pri tem vedno razdeli seznam na dve polovici. Tako imamo ob prvem klicu 2 polovici, ob drugem klicu 4 četrtine, ob tretjem klicu 8 osmin, itd., dokler ne pridemo do seznamov velikosti 1 ali 0. Temu pravimo dvojiški logaritem. Za delitev torej potrebujemo log(n) časa. Kaj pa za združevanje seznamov? Ker pri tem pregledamo vse elemente iz obe seznamov, je časovna zahtevnost kar n. Skupaj je torej časovna zahtevnost urejanja z zlivanjem	Sledijo razlagi in si zapisujejo. Odgovorijo: O(n^2).	Frontalno, razlaga, pogovor.

	O(n*log(n)). Hitro urejanje deluje na podoben način. Tudi tu razdelimo seznam glede na pivot na 2 podseznama ter ju uredimo. Časovna zahtevnost je tudi tukaj O(n*log(n)).		

ZAKLJUČNI DEL: ZAKLJUČNO PONAVLJANJE / PREVERJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
2 min	Spoznali smo 3 algoritme za urejanje elementov, pri čemer je eden počasnejši a enostavnejši, druga dva pa hitrejša, a zapletenejša za implementacijo. Pomembno si je torej zapomniti, da ima vsak postopek svoje prednosti in slabosti. *Učitelj se zahvali učencem za sodelovanje, jim da napotke za domačo nalogo. Dežurni učenec naj po potrebi pobriše tablo. Učenci naj ugasnejo računalnike.*	Poslušajo. Poslušajo, ugasnejo računalnike, dežurni učenec pobriše tablo.	Frontalno.

Priloge

Priloga 1: Tabela z rezultati

	Prosto urejanje	Urejanje z izbiranjem	Urejanje z zlivanjem	Hitro urejanje
[Ime_skupine 1]				
[Ime_skupine 2]				
…				
[Ime_skupine n]				

Literatura

[1] J. Hromkovič, Contributing to general education by teaching informatics, in: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, Springer, 2006, pp. 25–37.

[2] E. Shein, Should everybody learn to code?, Communications of the ACM 57 (2) (2014) 16–18.

[3] Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Projekt NAPOJ, Dostopno na: `https://lusy.fri.uni-lj.si/node/205`, (pridobljeno 17. 8. 2018).

[4] G. Anželj, J. Brank, A. Brodnik, P. Bulić, M. Ciglarič, M. Đukić, L. Fürst, M. Kikelj, A. Krapež, H. Medvešek, N. Mori, M. Pančur, P. Sterle, Računalništvo in informatika 1, Založba Univerze na Primorskem and Založba Fakultete za računalništvo in informatiko and Založba Fakultete za elektrotehniko and računalništvo in informatiko, 2015.
URL `https://lusy.fri.uni-lj.si/ucbenik/book/index.html`

[5] Univerza v Ljubljani, Fakulteta za matematiko in fiziko, Projekt TOMO, Dostopno na: `https://www.projekt-tomo.si`, (pridobljeno 17. 8. 2018).

[6] D. P. Groth, J. K. MacKie-Mason, Why an informatics degree?, Commun. ACM 53 (2) (2010) 26–28.

[7] G. A. Marco, Two false dogmas of information science, New Library World 97 (7) (1996) 11–14.

[8] W. Bernard, H. Reinder, H.-N. Robert, 1917 karl steinbuch 2005, IEEE Computational Intelligence Society.

[9] T. Matti, The Science of Computing, Chapman and Hall/CRC, 2015.

[10] Boston university, What is computer science, Dostopno na: `https://www.cs.bu.edu/AboutCS/WhatIsCS.pdf`, (pridobljeno 17. 8. 2018).

[11] Mississippi state university, Computer science, Dostopno na: http://www.cse.msstate.edu/computer-science/, (pridobljeno 17. 8. 2018).

[12] D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, P. R. Young, P. J. Denning, Computing as a discipline, Communications of the ACM 32 (1) (1989) 9–23.

[13] A. Tucker, F. Deek, J. Jones, D. McCowan, C. Stephenson, A. Verno, A model curriculum for K–12 computer science, Final Report of the ACM K-12 Task Force Curriculum Committee, Computer Science Teachers Association (2003).

[14] University of Edinburgh, What is informatics?, dostopno na: https://www.ed.ac.uk/files/atoms/files/what20is20informatics.pdf (pridobljeno 18. 8. 2018) (julij 2016).

[15] University of Washington, What is informatics?, Dostopno na: https://ischool.uw.edu/programs/informatics/what-is-informatics, (pridobljeno 18. 8. 2018).

[16] Indiana university, school of informatics and computing, Informatics defined, Dostopno na: https://soic.iupui.edu/about/what-is-informatics/, (pridobljeno 18. 8. 2018).

[17] N. M. Luscombe, D. Greenbaum, M. Gerstein, What is bioinformatics? a proposed definition and overview of the field, Methods of information in medicine 40 (04) (2001) 346–358.

[18] K. Jawad, Uses of computers in various fields, Dostopno na: http://www.byte-notes.com/uses-computers-various-fields, (pridobljeno 18. 8. 2018).

[19] B. S. Lerner, How computer science advances other disciplines, Dostopno na: https://www.mtholyoke.edu/~blerner/Applications.html, (pridobljeno 18. 8. 2018) (2014).

[20] K. MacPherson, 3-D computer simulations help envision supernovae explosions, Dostopno na: https://www.princeton.edu/news/2010/09/16/3-d-computer-simulations-help-envision-supernovae-explosions (pridobljeno 18. 8. 2018) (september 2010).

[21] Gospodarska zbornica Slovenije, Digitagenda 2016: 30 priporočil za nova digitalna delovna mesta, Dostopno na: https://vrhgospodarstva.gzs.si/vsebina/Arhiv/VSG-2016/DigitAgenda-2016, (pridobljeno 20. 8. 2018) (november 2016).

[22] A. Krapež, V. Rajkovič, V. Batagelj, R. Wechtersbach, Razvoj predmeta računalnistvo in informatika v osnovni in srednji šoli, Zbornik posvetov 26 (2001).

[23] Strokovna delovna skupina za analizo prisotnosti vsebin računalništva in informatike v programih osnovnih in srednjih šol ter za pripravo študije o možnih spremembah, Snovalci digitalne prihodnosti ali le uporabniki?, Ministrstvo za izobraževanje, znanost in šport (maj 2018).

[24] R. Kranjc, A. Drinovec, A. Brodnik, I. Pesek, I. Nančovska Šerbec, J. Demšar, A. Žerovnik, M. Lokar, Računalništvo za 4., 5. in 6. razred, učni načrt, Ministrstvo za izobraževanje znanost in šport, Zavod RS za šolstvo, 2013.

[25] V. Batagelj, R. Wechtersbach, I. Gerlič, A. Krapež, S. Zamuda, S. Muršec, Računalništvo za 7., 8. in 9. razred, učni načrt, Ministrstvo za izobraževanje znanost in šport, Zavod RS za šolstvo, 2002.

[26] N. Kristan, A. Brodnik, Primerjava učnih ciljev in kurikulumov računalništva in informatike, in: Vzgoja in izobraževanje v informacijski družbi, Informacijska družba IS 2013, 2013, pp. 136–143.

[27] V. Guerra, B. Kuhnt, I. Blochliger, Informatics at school–worldwide: An international exploratory study about informatics as a subject at different school levels, in: An international, Hasler Foundation, University of Zurich, 2012, pp. 12–30.

[28] J. Fraillon, J. Ainley, W. Schulz, T. Friedman, Preparing for life in a digital age: The IEA International Computer and Information Literacy Study international report, Springer, 2014.

[29] I. Gerlič, Stanje in trendi uporabe informacijsko komunikacijske tehnologije v slovenskem izobraževalnem sistemu, Vzgoja in izobraževanje v informacijski družbi 13 (2010) 111–118.

[30] Dosežki držav na tekmovanju IOI, Dostopno na: `http://stats.ioinformatics.org/countries/`, (pridobljeno 24. 8. 2018).

[31] Google Inc. & Gallup Inc., Trends in the state of computer science in U.S. K-12 schools, Dostopno na: `http://goo.gl/j291E0`, (pridobljeno 21. 8. 2018) (2016).

[32] F. Olsen, Computer scientist says all students should learn to think "algorithmically", The Chronicle of High Education 46 (2000).

[33] T.-C. Hsu, S.-C. Chang, Y.-T. Hung, How to learn and how to teach computational thinking: Suggestions based on a review of the literature, Computers & Education 126 (2018) 296–310.

[34] G. Futschek, Algorithmic thinking: the key for understanding computer science, in: International conference on informatics in secondary schools-evolution and perspectives, Springer, 2006, pp. 159–168.

[35] V. Barr, C. Stephenson, Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?, ACM Inroads 2 (1) (2011) 48–54.

[36] L. Perković, A. Settle, S. Hwang, J. Jones, A framework for computational thinking across the curriculum, in: Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, ACM, 2010, pp. 123–127.

[37] F. Buitrago Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo, G. Danies, Changing a generation's way of thinking: teaching computational thinking through programming, Review of Educational Research 87 (4) (2017) 834–860.

[38] J. Hromkovič, T. Kohn, D. Komm, G. Serafini, Examples of algorithmic thinking in programming education, Olympiads in Informatics 10 (1-2) (2016) 111–124.

[39] S. Grover, Classroom strategies, CSTA Voice 12 (1) (2016) 7–8.

[40] R. Laucius, Issues of selecting a programming environment for a programming curriculum in general education, in: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, Springer, 2006, pp. 169–178.

[41] L. Grandell, M. Peltomäki, R.-J. Back, T. Salakoski, Why complicate things?: introducing programming in high school using python, in: Proceedings of the 8th Australasian Conference on Computing Education-Volume 52, Australian Computer Society, Inc., 2006, pp. 71–80.

[42] K. K. Agarwal, A. Agarwal, Python for CS1, CS2 and beyond, Journal of Computing Sciences in Colleges 20 (4) (2005) 262–270.

[43] K. K. Agarwal, A. Agarwal, M. E. Celebi, Python puts a squeeze on java for CS0 and beyond, Journal of Computing Sciences in Colleges 23 (6) (2008) 49–57.

[44] J. Heliotis, R. Zanibbi, Moving away from programming and towards computer science in the CS first year, Journal of Computing Sciences in Colleges 26 (3) (2011) 115–125.

[45] M. A. Weiss, Data Structures and algorithm analysis (2nd edition), Addison Wesley, 1994.

[46] M. T. Goodrich, R. Tamassia, Data structures and algorithms in Java, Wiley, 2001.

[47] M. Kovač, J. Strel, Učna priprava, Nekatera poglavja iz didaktike športne vzgoje v 1. in 2. triletju devetletne osnovne šole (2004) 120–134.

[48] B. Haberman, O. Muller, Teaching abstraction to novices: pattern-based and ADT-based problem-solving processes, in: Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, IEEE, 2008, pp. F1C–7.

[49] N. Dale, H. M. Walker, Abstract data types: specifications, implementations, and applications, Jones & Bartlett Learning, 1996.

[50] Desmos Inc., Graphing calculator, Dostopno na: `https://www.desmos.com/calculator`, (pridobljeno 1. 10. 2018).

[51] C. A. Shaffer, A practical introduction to data structures and algorithm analysis, NJ: Prentice Hall, Upper Saddle River, 1997.

[52] J. Gal-Ezer, T. Vilner, E. Zur, Teaching algorithm efficiency at CS1 level: a different approach, Computer Science Education 14 (3) (2004) 235–248.

[53] T. J. Rolfe, Classroom exercise demonstrating linked list operations, ACM SIGCSE Bulletin 38 (4) (2006) 83–84.

[54] B. Vöcking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, D. Wagner, Algorithms unplugged, Springer Science & Business Media, 2010.

[55] J. Demšar, Urejanje, Dostopno na: `http://vidra.si/urejanje`, (pridobljeno 26. 9. 2018).

[56] C.-C. Li, An immediate approach to balancing nodes in binary search trees, Journal of Computing Sciences in Colleges 21 (4) (2006) 238–245.

[57] M. Singh, D. Garg, Choosing best hashing strategies and hash functions, in: Advance Computing Conference, 2009. IACC 2009. IEEE International, IEEE, 2009, pp. 50–55.