

UNIVERZA V MARIBORU
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO
Oddelek za matematiko in računalništvo

DIPLOMSKO DELO

Gregor Nemeč

Maribor, 2016

UNIVERZA V MARIBORU
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO
Oddelek za matematiko in računalništvo

DIPLOMSKO DELO

Spletni portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanju

Mentor:
doc. dr. Igor Pesek

Kandidat:
Gregor Nemeč

Maribor, 2016

ZAHVALA

Citat

*verz (možna opcija)(*ustrezno spremeniti*)*

*Zahvala...(*ustrezno spremeniti*)*

*Posebna hvala še...(*ustrezno spremeniti*)*

*Vsem iskreno hvala.(*ustrezno spremeniti*)*

UNIVERZA V MARIBORU
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

IZJAVA

Podpisani Gregor Nemeč, rojen 15. oktobra 1983, študent Fakultete za naravoslovje in matematiko Univerze v Mariboru, študijskega programa Fizike in Računalništva, izjavljam, da je diplomsko delo z naslovom

Spletni portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanju

pri mentorju doc. dr Igorju Pesku avtorsko delo. V diplomskem delu so uporabljeni viri in literatura korektno navedeni; teksti in druge oblike zapisov niso uporabljeni brez navedbe avtorjev.

Maribor, 2016

Gregor Nemeč
(*podpis kandidata*)

Nemec, G.: Spletни portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanju. Diplomsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Oddelek za matematiko in računalništvo, 2016.

Povzetek

V zadnjem času so se pojavili številni spletni portali, ki ponujajo učenje programiranja. V diplomske delu nas zanima, kateri so ti spletni portali in kako jih lahko umestimo v pouk v osnovnih in srednjih šolah. V ta namen smo pregledali spletne portale, ki so nastali v akademskem okolju ter smo določili kriterije, ki so bili glavno vodilo pri klasifikaciji in vrednotenju spletnih portalov. Novonastalih portalov je precej, zato smo izbiro omejili s kriteriji, kot je brezplačen dostop do gradiv in da so vrste vsebin napredno kombinirane. Pregled portalov je pokazal, da vsak izmed njih prinaša številne zmožnosti pri uporabi, izpostavili smo tudi nekatere slabosti. S konkretnima primeroma pokažemo, kako bi lahko uporabili spletni portal pri pouku.

Ključne besede: Učenje programiranja, spletni portali, programski jeziki ...

Nemec, G.: angleški naslov. Graduation Thesis, University of Maribor, Faculty of Natural Sciences and Mathematics, Department of Computer Science, 2016.

Abstract

Key words:

Kazalo

1 Uvod	10
2 Uporaba računalnika v izobraževanju	11
2.1 Splošnoizobraževalno področje	11
2.2 Strokovno izobraževalno področje	12
2.3 Programiranje v OŠ	13
2.3.1 Izbirni predmet računalništva	13
2.3.2 Neobvezni izbirni predmet računalništva	14
2.4 Programiranje v SŠ	17
2.4.1 Informatika - splošni gimnazijski program	17
2.4.2 Računalništvo - tehniška gimnazija	18
3 Računalniška znanost in programiranje	20
3.1 Zgodovina programskih jezikov	20
3.2 Osnovni pojmi	21
3.2.1 Program	21
3.2.2 Algoritem	21
3.2.3 Programiranje in kodiranje	22
3.2.4 Urejevalnik besedil	23
3.2.5 Integrirano razvojno okolje	23
3.3 Programske paradigmе	24
3.3.1 Objektno orientirano programiranje	24
3.4 Programski jeziki	25
3.4.1 Java	25
3.4.2 C++	26

3.4.3 Java Script	27
3.4.4 Python	27
3.5 Osnovni koncepti programiranja	28
3.5.1 Izpis in računske operacije	29
3.5.2 Spremenljivke	30
3.5.3 Pogojni stavki in vejitve	31
3.5.4 Zanke	31
4 Pristopi in strategije poučevanja	32
4.1 Model aktivnega učenja	32
4.2 Strategije reševanja problemov	33
4.2.1 Razumevaje problema	34
4.2.2 Načrtovanje rešitve	34
4.2.3 Preverjanje rešitve	35
4.2.4 Refleksija	35
5 Spletni portali za učenje programiranja	37
5.1 Razlogi za nastanek spletnih portalov	37
5.2 Primeri implementacije in sistemska arhitektura	39
5.3 Pregled delovanja in interakcija s SPUP	40
5.4 Rezultati izvedenih rešitev SPUP	42
5.5 Značilnosti SPUP	43
6 Kriteriji za klasifikacijo spletnih portalov	44
6.1 Vrsta vsebine	44
6.1.1 Tekstovni vodiči	44
6.1.2 Video vodiči	45

6.1.3	Spletna aplikacija za programiranje	45
6.1.4	Spletne igre	46
6.1.5	Kombinirane vrste vsebin	47
6.2	Jezik spletnega portala	48
6.3	Ponujena znanja	49
6.4	Programski jeziki	49
6.5	Težavnostna stopnja	49
6.6	Upoštevanje učnih načel	50
6.7	Uporaba ocenjevanja dosežkov, značilnih za igre	50
6.8	Dodajanje lastnih vsebin	51
6.9	Upravljanje razreda	51
6.10	Dostop do gradiv	51
7	Pregled spletnih portalov	53
7.1	Code.org	53
7.1.1	Ura kode (<i>ang. Hour of code</i>)	53
7.1.2	Code studio	55
7.1.3	Samostojni projekti	56
7.1.4	Uporaba za učitelje	58
7.1.5	Povzetek	58
7.2	Codeacademy	59
7.2.1	Uporaba za učitelje	62
7.2.2	Povzetek	64
7.3	Scratch	66
7.3.1	Uporaba Scratcha	67
7.3.2	Deljenje in raziskovanje projektov	68

7.3.3	Povzetek	68
7.4	Repl.it	69
7.4.1	Ustvarjanje razredov in nalog	70
7.4.2	Povzetek	70
7.5	Tutorialspoint	71
7.5.1	Coding ground	72
7.5.2	Povzetek	74
7.6	Thimble	74
7.6.1	Povzetek	76
7.7	Code combat	76
7.7.1	Upravljanje razreda	81
7.7.2	Povzetek	82
7.8	Codingame	83
7.8.1	Povzetek	86
8	Možni načini uporabe spletnih portalov pri pouku	87
8.1	Primer uresničevanja ciljev učnega načrta v osnovni šoli	87
8.2	Primer uresničevanja ciljev učnega načrta v srednji šoli	89
9	Zaključek	91

Tabele

1	Operativni cilji, dejavnosti in vsebine izbirnega predmeta računalništvo za III. dodatno enoto. [2]	13
2	Prikaz dvojnih, togih in mehkikh orisov idej na primeru spremenljivke [6]. . . .	28
3	Primerjava starosti in stopnje šolanja šolskega sistema v ZDA in Sloveniji [31]. .	50

Slike

1	Sistemska arhitektura spletnega portala za učenje programiranja, kot so jo naredili na OUHK [19].	39
2	Prikaz interakcije med študentom in mentorjem s spletnim portalom [19].	40
3	Zaslonski posnetek poglavja z vodiča <i>The Python Tutorial</i> s primerom [22].	45
4	Uvodna stran enega izmed video vodičev za učenje Pythona [24].	46
5	Zaslonska slika spletne aplikacije za programiranje <i>Python Fiddle</i> [25].	46
6	Zaslonska slika spletne strani Fightcode [28].	47
7	Zaslonska slika spletne strani <i>w3School</i> [29] v poglavju HTML.	48
8	Zaslonska slika spletne strani <i>Codeschool</i> [30].	48
9	Zaslonska slika dela začetne spletne strani <i>code.org</i> [35], iz katerega je razvidna razdeljenost vsebin.	54
10	Spletna aplikacija Code studio, v kateri so zgrajene vadnice [35].	54
11	Podstran Code studio, kjer lahko nadaljujemo na različnih vsebinskih sklopih [37].	55
12	Podstran vsebinskega sklopa [37].	56
13	Samostojni aplikaciji za programiranje, primerni za osnovno šolo [37].	57
14	Spletна aplikacija za programiranje - ustvarjanje aplikacij. Primer: preprosto računalno [37].	57
15	Zaslonska slika spletne strani <i>Codeacademy</i> [38]. Začetna, nadzorna stran po prijavi, od tu nadaljujemo na vsebinske sklope, ki smo jih že začeli.	59
16	Zaslonska slika spletne strani <i>Codeacademy</i> [38]. Seznam znanj/veščin programskih jezikov, ki jih ponuja spletni portal.	60
17	Zaslonska slika podstrani spletne strani <i>Codeacademy</i> [38], na kateri lahko pregledujemo posamezne teme in nadaljujemo tam, kjer smo ostali.	61
18	Zaslonska slika <i>Codeacademy</i> [38] vadnice, ki jo sestavlja urejevalnik z navodili in oknom za izpis v programu.	61
19	Plakat s povezavami enot različnih tematskih sklopov po stopnjah, ki vodijo do posameznih ciljev (<i>ang. Level prograsion mapa</i>) [38].	63

20	Zaslonska slika <i>Codeacademy</i> [38]. Prikazuje tabelo za sledneje napredku učenjem.	64
21	Zaslonski posnetek glavne strani <i>Scratch</i> [39].	66
22	Zaslonska slika <i>Scratch</i> [39].	67
23	Podstran za deljenje projekta na <i>Scratchu</i> [39].	68
24	Zaslonska slika spletnne aplikacije za programiranje <i>repl.it</i> [41].	69
25	Zaslonska slika pogleda mentorja v načinu priprave naloge [41].	70
26	Del seznama oz. knjižnica vodičev, ki ga ponuja spletna stran <i>Tutorials point</i> [43].	72
27	Zaslonski izrez vodiča za Python3. Slike je razvidno kazalo in gumb za preizkus [43].	72
28	Podokno za preizkus primera programske kode [43].	72
29	Del seznama različnih programskih jezikov, ki jih lahko uporabljamo s spletno aplikacijo za programiranje <i>Codingground</i> [44].	73
30	Spletna aplikacija za programiranje - <i>Codingground</i> [44].	73
31	Urejanje projekta na strani <i>Thimble</i> [45].	75
32	Izbira junaka in programskega jezika [46].	77
33	Izbor zemljevida, na katerem bomo reševali naloge [46].	77
34	Podroben zemljevid za izbiro nalog [46].	78
35	Oprema junaka in njen opis [46].	78
36	Primerjava med prejšnjo različico opreme in njeno nadgradnjo, ki jo lahko zamenjamo junaku [46].	79
37	Postavitev igre [46].	80
38	Uspešno končan izid igre z napisano programsko kodo [46].	80
39	Končni rezultat in pregled nad dobljenimi dosežki ob koncu igre [46].	80
40	Učiteljev pogled na upravljanje razreda [46].	81
41	Podstran <i>Codingame</i> [48] - sestavljanke.	84
42	Reševanje sestavljanke <i>The Bridge</i> [48]	85

43	Primerjava programskih blokov, napisanih v Scratchu , in enačic klica metod, napisanih v Pythonu na spletni strani <i>Codecombat</i>	88
44	Miselni vzorec s povzetkom pregledanih spletnih portalov s predstavniki in značilnostmi za posamezno vrsto vsebine.	93

Pregled uporabljenih simbolov in označb

1. **OŠ** - osnovna šola,
2. **SŠ** - srednja šola,
3. **IKT** - informacijsko-komunikacijska tehnologija,
4. **IRO** - integrirano razvojno okolje (*ang. Integrated development environment IDE*),
5. **OUHK** - Odprta univerza Hong Kong,
6. **USVB** - Univerze Strathclyde Velika Britanija,
7. **QUTA** - Queensland University of Technology, Australia,
8. **OO** - objektno orientirano,
9. **SPUP** - spletni portal za učenje programiranja,
10. **SPZP** - spletna aplikacija za programiranje,
11. **AVP** - aplikacijski programski vmesnik (*ang. application programming interface API*).

1 Uvod

V svetovnem merilu se pojavlja trend popularizacije programiranja in kodiranja. V zadnjem obdobju so se na spletu pojavili številni portali, kot je na primer *Codeacademy* [38], ki ponujajo učenje programiranja. Zanima nas, kateri so ti portali, katere vsebine ponujajo, na kakšen način so vsebine predstavljene in ali so uporabni za uporabo pri pouku ter na kakšen način bi jih lahko uporabili.

Najprej bomo raziskali, kako je definirana računalniška pismenost in uporaba računalnika pri pouku. Ogledali si bomo, kje se uči programiranje na osnovni (OŠ) in srednji šoli (SŠ), pri katerih izbirnih vsebinah, predmetih in kakšna je vsebina, ki jo predvideva učni načrt. Podrobneje bomo preučili vse tiste pojme, ki se pojavljajo v računalniški znanosti in programiranju, ki nam bodo pomagali bolje razumeti spletnne portale in vsebino, ki jo predstavljajo. Preučili bodo tudi sodobne pristope in strategije, ki se uporabljajo pri učenju programiranja, saj bomo tako lažje ocenili, ali jih spletni portali tudi znajo upoštevati.

Zanimali nas bodo **novinci** in njihove težave pri začetnih korakih učenja programiranja. Ko govorimo o novincih, imamo v mislih vse tiste učence, dijake in študente, ki se šele srečujejo s programiranjem, ne glede na spol. Prav tako nas bodo zanimali **mentorji**, s katerimi imamo v mislih učitelje in profesorje.

V ta namen bomo pregledali nastanek spletnih portalov za namen učenja programiranja, ki so se pojavili v akademskem okolju na posameznih univerzah. Zanimal nas bo razlog za nastanek takšnih spletnih portalov na univerzah, zato bomo pregledali literaturo in poskusili ugotoviti, zakaj in kako se na višešolskem področju uporablja spletnne tehnologije za poučevanje programiranja in kako so skušale premostiti nekatere težave, ki jih imajo novinci pri učenju programiranja.

Izluščili bomo predlagane rešitve za uporabo spletnih tehnologij pri učenje programiranja. Na podlagi pregledanega bomo lahko določili kriterije, s katerimi bomo lahko klasificirali spletnne portale in jih tudi ovrednotili ter uspešno umestili v pouk.

2 Uporaba računalnika v izobraževanju

Model uporabe računalnika v izobraževanju je Gerlič [1] razdelil na tri področja. V **primarno področje** lahko uvrstimo učenje računalništva in programiranja, saj sem prištevamo aktivnosti, s katerimi želimo uporabnike seznaniti z delovanjem in uporabo računalnika oz. sodobne informacijsko-komunikacijske tehnologije (**IKT**). Računalnik je tista učna vsebina, ki jo obravnavamo. **Sekundarno področje** predstavlja vse tiste aktivnosti, ki so vezane neposredno na izobraževalni proces katerega koli predmetnega področja. Računalnik in IKT nastopata kot učno sredstvo ali pripomoček v oblikah tradicionalnih računalniško podprtih učnih sistemov ali inteligentnih ekspertnih sistemov. V **terciarno področje** spadajo vse aktivnosti, ki spremljajo izobraževanje. Sem se štejejo aktivnosti izobraževanja, vodenja in upravljanja izobraževalnega sistema.

V tem diplomskem delu nas bo zanimalo le **primarno področje** uporabe računalnika v izobraževanju, ki je razdeljeno v dva pomembna področja [1]:

- kot element **splošne izobrazbe**,
- kot element **ožje strokovne** - poklicne izobrazbe oz. usposabljanja.

2.1 Splošnoizobraževalno področje

V današnjem času se računalnik kot element splošne izobrazbe kaže kot velika potreba oz. se zdi znanje njegove uporabe samoumevno. Že pri najmlajših otrocih računalnik vzbuja zanimanje in interes. Računalnik je postal intelektualno orodje in pripomoček v vsaki sferi človekove dejavnosti in je prodrli že dolgo nazaj tudi v šolo. Tako imenovana **računalniška pismenost** postaja nuja in zajema vse to, kar bi človek moral znati o računalniku, in to, kako je potrebno z njim delati, da bo uspešno živel v družbi, ki je osnovana na informacijah oz. v informacijski družbi. V zvezi z definicijo in pojmovanjem **računalniške pismenosti** se kažeta dve usmeritvi, Gerlič [1] navaja številne avtorje obeh usmeritev in povzema:

Prva poudarja **sposobnost računalniškega programiranja** in opredeljuje s pojmom pismenosti sposobnost branja in pisanja podobno, kot je to značilno za jezikovno pismenost. Tako zagovorniki te smeri poudarjajo, da je cilj računalniške pismenosti učenje in večina programiranja z novim načinom mišljenja in strategijami ugotavljanja in popravljanja računalniških programov.

Druga smer poudarja **splošno usposobljenost** za delo z računalnikom in to, da ni smiselno, da vsakdo postane programer zaradi tega, ker se bo računalnik uporabljal v najširšem smislu v praksi. Pomembno za učenca je, da razume delovanje računalnika in se zaveda njegovega vpliva na razvoj družbe. Učencu moramo pomagati, da dejanske probleme identificira in jih

lahko reši že z narejeno komercialno programsko opremo.

V zgodnjih letih sta bili značilni obe usmeritvi. Pozneje je prišlo do preobrata leta 1987 po mednarodnem simpoziju na Univerzi v Stanfordu. Eden od sklepov simpozija je bil ta, da se v splošnoizobraževalnih programih ne uči več programiranja, še posebej ne strukturiranih različic programskega jezika, kot je na primer **BASIC**, temveč naj se uči uslužnostno programsko opremo, kot je urejevalnik besedil, orodja za delo s podatkovnimi bazami, grafična orodja itd. Ta dejstva je Gerlič [1] povzel leta 2000 in je predlagal isto usmeritev z naslednjim navedkom:

“Učencem vseh stopenj želimo ob čim večjem številu ur praktičnega dela z računalnikom, ob določenem problemu in ob uporabi ustreznih komercialnih programov seznaniti z osnovami računalništva in informatike.”

Zanima nas, koliko je računalniške znanosti in programiranja v splošnoizobraževalnem področju v učnih načrtih za **OŠ** in **SŠ**. Novi trendi in potrebe industrije kažejo na potrebo povečanja po znanjih programiranja. Ali bi bilo morda zaradi razširjenosti računalniške tehnologije, ki je na vsakem koraku, potrebno definicijo računalniške znanosti, ki jo je povzel Gerlič, posodobiti in ji dodati nazaj osnove znanja računalništva in programiranja ter uporabe odprtokodne programske opreme. Predvsem pri mlajših generacijah obstaja potreba in trend, da se programiranje ponudi v širšem kontekstu tudi na splošnem izobraževalnem nivoju.

Ne glede na to, kako bo v prihodnje definirana računalniška pismenost, želimo v diplomskem delu pokazati, da spletni portali za učenje programiranja zaradi tehnološkega napredka omogočajo lažjo pot k učenju programiranja. Zanimalo nas bo tudi, kje je umeščeno programiranje v učnem načrtu za **OŠ** in **SŠ**.

2.2 Strokovno izobraževalno področje

Sem prištevamo vse tiste aktivnosti, s katerimi želimo udeležence izobraževanja usposobiti na različnih ravneh tako, da se bodo ti z računalnikom in informacijskimi sistemi ukvarjali na poklicni ravni. Lahko povemo, da sem spadajo vse ozko usmerjene računalniške in informacijske smeri srednjih šol, visokih, višjih ter univerzitetnih študijev. Zanima nas področje programiranja, kar je sicer predmet strokovnega izobraževanja, vendar nas bo zanimalo programiranje na splošnem izobraževalnem področju. Spletne portale za učenje programiranja bomo predstavili predvsem kot vstopno točko za začetnike in novince, ki jih najdemo prav tako na vseh stopnjah.

2.3 Programiranje v OŠ

Pouk računalništva v OŠ poteka kot **izbirni predmet** ali kot **neobvezni izbirni predmet**. Za začetek bomo navedli, kako je definirano računalništvo v učnem načrtu za izbirne predmete v osnovni šoli [2]: "*Računalništvo je naravoslovno-tehnični izbirni predmet, pri katerem se spoznavanje in razumevanje osnovnih zakonitosti računalništva prepleta z metodami neposrednega dela z računalniki, kar odpira učencem in učenkam možnost, da pridobijo tista temeljna znanja računalniške pismenosti, ki so potrebna pri nadalnjem izobraževanju in vsakdanjem življenju*".

2.3.1 Izbirni predmet računalništva

Izbirni predmeti računalništva so sestavljenih iz treh predmetov in jih učenke in učenci lahko izberejo v tretjem trilerju, v 7., 8. in/ali 9. razredu. Prvi od teh predmetov je **računalništvo - urejanje besedil**, kjer si učenci pridobijo osnovna znanja, ki so potrebna za razumevanje in temeljno uporabo računalnika. Naslednja dva predmeta sta **računalniška omrežja** in **multimedija**, kjer se ta znanja spiralno nadgradijo.

Zanima nas, kje se pri teh predmetih računalništva pojavlja programiranje. Vsak izmed teh predmetov ima operativne učne cilje razdeljene tako, da programiranje najdemo v tretji enoti, ki spada med dodatne vsebine. Posamezne operativne cilje, dejavnosti in vsebino prikazuje tabela 1.

Tabela 1: Operativni cilji, dejavnosti in vsebine izbirnega predmeta računalništvo za III. dodatno enoto. [2]

OPERATIVNI CILJI	DEJAVNOSTI	VSEBINE
<ul style="list-style-type: none">• Napisati algoritem z odločitvijo, ki reši preprost vsakdanji problem,• izdelati in spremeniti računalniški program z odločitvijo.	<ul style="list-style-type: none">• Analizirati preprost problem,• uporabljati osnovne korake programiranja.	<ul style="list-style-type: none">• Risanje diagrama poteka za problem z odločitvijo,• izdelava računalniškega programa.

Programiranje se pojavlja le kot dodatna vsebine, kar se kaže v uresničitvi smeri računalništva, ki zagovarja **splošno usposobljenost**, kot smo jo opredelili v poglavju 2.1. Vendar se tu učiteljem računalništva ponuja izjemna priložnost, da z učenci in učenkami naredijo korak v osnove računalništva in programiranja pri vseh treh predmetih.

2.3.2 Neobvezni izbirni predmet računalništva

Opredelitev predmeta v učnem načrtu [3] pravi, da neobvezni izbirni predmet računalništva učence seznanja z različnimi področji računalništva, računalniškimi koncepti ter procesi in jih ne učijo dela s posameznimi programi. Učenci se seznanjajo s tehniko in metodami reševanja problemov, razvijajo algoritmičen način razmišljanja. Opredelitev zadostuje prvi smeri računalniške pismenosti, ki zagovarja **sposobnost računalniškega programiranja**. Ker se v tem diplomskem delu še posebej posvečamo programiranju, nam učni načrt tega predmeta dosti bolj ustreza in si ga bomo podrobneje pogledali, saj nam bo pregled operativnih ciljev služil kot vodilo pri nadalnjem delu. Najprej preglejmo splošne cilje, ki jih povzemamo po učnem načrtu za neobvezni izbirni predmet računalništvo [3] in jih morajo učenci doseči:

- spoznavajo temeljne koncepte računalništva,
- razvijajo algoritmični način razmišljanja in spoznavajo strategije reševanja problemov,
- razvijajo sposobnost in odgovornost za sodelovanje v skupini ter si krepijo pozitivno samopodobo,
- pridobivajo sposobnost izbiranja najustreznejše poti za rešitev problema,
- spoznavajo omejitve človeških sposobnosti in umetne inteligence,
- se zavedajo omejitev računalniških tehnologij,
- pridobivajo zmožnost razdelitve problema na manjše probleme,
- se seznanjajo z abstrakcijo oz. poenostavljanjem,
- spoznavajo in razvijajo zmožnost modeliranja, strokovno terminologijo,
- razvijajo ustvarjalnost, natančnost in logično razmišljanje,
- razvijajo in bogatijo svoj jezikovni zaklad ter skrbijo za pravilno slovensko izražanje in strokovno terminologijo.

Neobvezni izbirni predmet je namenjen učencem 4., 5. in 6. razreda. V učnem načrtu so operativni cilji predstavljeni tako, da so temeljni označeni **krepko** in izbirni *poševno*. Navedli bomo tiste, ki so navezujejo na programiranje in strategije reševanja problemov. Operativni cilji so razdeljeni na vsebinske sklope.

Vsebina/sklop: Algoritmi

- **Razumejo pojem algoritma,**
- **znajo vsakdanji problem opisati kot zaporedje korakov,**
- **znajo z algoritmom predstaviti preprosto opravilo,**
- **algoritem predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku,**
- **sledijo algoritmu, ki ga pripravi nekdo drug,**
- **znajo v algoritmu vključiti vejitev (če) in ponavljanje (zanke),**
- **znajo algoritem razgraditi na gradnike (podprograme),**

- **znajo povezati več algoritmov v celoto, ki reši neki problem,**
- *razumejo vlogo testiranja algoritma in vedo, da je testiranje orodje za iskanje napak in ne za potrjevanje pravilnosti,*
- *primerjajo več algoritmov za rešitev problema in znajo poiskati najustreznejšega glede na dana merila,*
- *znajo uporabiti nekatere ključne algoritme za sortiranje in iskanje,*
- *poznajo osnovne algoritme za iskanje podatkov.*

Vsebina/sklop: Programi

- **Znajo slediti izvajanju tujega programa,**
- **znajo algoritem zapisati s programom,**
- **znajo v program vključiti konstante in spremenljivke,**
- *razumejo različne podatkovne tipe in jih znajo uporabiti v programu,*
- *znajo spremenljivkam spremeniti vrednost s prireditvenim stavkom,*
- **znajo v programu prebrati vhodne podatke in jih vključiti v program,**
- **znajo izpisovati vrednosti spremenljivk med izvajanjem programa in izpisati končni rezultat,**
- **v program vključijo logične operatorje,**
- **znajo uporabiti pogojni stavek in izvesti vejitev,**
- **razumejo pojem zanke in ga znajo uporabiti za rešitev problema,**
- *razumejo kompleksnejše tipe podatkov (nizi, seznamy/tabele) in jih znajo uporabiti v programu,*
- **prepoznajo in znajo odpraviti napake v svojem programu,**
- *znajo popraviti napako v tujem programu,*
- *znajo spremeniti program, da dosežejo nov način delovanja programa,*
- *znajo rezultate naloge zapisati v datoteko,*
- **se seznanijo z dogodkovnim programiranjem,**
- **so zmožni grafične predstavitve scene (velikost objektov, ozadje, pozicioniranje),**
- **so zmožni sinhronizacije dialogov/zvokov,**
- **so zmožni razumeti in realizirati interakcije med liki in objekti,**
- **so zmožni ustvarjati animacije.**

Vsebina/sklop: Podatki

- **Razlikujejo podatek in informacijo,**
- **razumejo dvojiški sistem zapisovanja različnih podatkov,**
- **razumejo kodiranje podatkov,**
- **razumejo, da obstajajo podatki v različnih pojavnih oblikah (besedilo, zvok, slike, video),**
- *poznajo načine predstavitev določenih podatkov in odnose med njimi (dvojiška drevesa*

in grafi),

- *vedo za stiskanje podatkov in vedo, da je stiskanje lahko brez izgub ali z izgubami,*
- **pojasnijo razliko med konstantami in spremenljivkami v programu,**
- *poznajo osnovne algoritme za iskanje podatkov.*

Vsebina/sklop: Reševanje problemov

- *Znajo uporabiti različne strategije za reševanje problema,*
- **znajo našteti faze procesa reševanja problema,**
- *znajo postavljati vprašanja in ugotoviti, kateri podatki so znani,*
- *znajo za podano nalogu izluščiti bistvo problema,*
- *znajo najti ustrezno orodje, s katerim rešijo problem,*
- *znajo problem razdeliti na več manjših problemov,*
- **znajo načrtovati in realizirati rešitev,**
- *za podano rešitev znajo oceniti posledice in vpliv na "okolje",*
- *znajo uporabiti znano strategijo v novih okoliščinah,*
- *znajo ustvariti nov algoritem za kompleksnejše probleme,*
- **znajo učinkovito sodelovati v skupini in rešiti problem z uporabo informacijsko-komunikacijske tehnologije,**
- *znajo ceniti neuspešne poskuse reševanja problema kot del poti do rešitve,*
- *znajo kritično ovrednotiti rešitev in ugotoviti, ali rešitev uspešno reši dani problem,*
- *znajo kritično ovrednotiti strategijo reševanja problema,*
- *zavedajo se omejitve informacijsko-komunikacijske tehnologije pri reševanju problemov.*

Vsebina/sklop: Komunikacija in storitve

- **Znajo uporabiti ustrezna orodja in metode za iskanje po spletu,**
- *znajo uporabiti različne iskalne strategije v iskalnikih,*
- **poznajo omejitve pri rabi na spletu najdenih informacij (zavedajo se pojma intelektualna lastnina).**

Kot smo že povedali, smo nekatere cilje izvzeli, čeprav je znanje računalniških omrežij pomembno, ga z vidika programiranja lahko izpustimo. Lahko povemo, da nam je večina operativnih ciljev ostala, saj smo jih lahko izključili le malo. Tako pridemo do spoznanja, da se v računalniški znanosti pretežni del znanja vrti okoli programiranja in reševanja problemov, kar je zajeto v tem učnem načrtu. Iz ciljev lahko spoznamo tudi samo zahtevnost snovi. Lahko si upamo napovedati, da jih bomo s pravimi orodji, kot so spletni portali za učenje programiranja, tudi lažje dosegli.

2.4 Programiranje v SŠ

Pregledali bomo predmet **informatike** splošnega gimnazijskega programa in **računalništvo** tehniške gimnazije.

2.4.1 Informatika - splošni gimnazijski program

Predmet **informatike** se poučuje v prvem letniku splošne, klasične in strokovne gimnazije. Če povzamemo opredelitev iz učnega načrta [4], ta pravi naslednje: *"Informatika je splošnoizobraževalni predmet, pri katerem se teorija poznavanja in razumevanja osnovnih zakonitosti informatike prepleta z metodami neposrednega iskanja, zbiranja, hranjenja, vrednotenja, obdelave in uporabe podatkov z digitalno tehnologijo z namenom oblikovanja relevantnih informacij za dogajevanje lastnega znanja in za njegovo predstavitev oziroma posredovanje drugim."*

Poučevanje informatike obsega *70 ur* v 1. letniku, izbirni predmet obsega *210 ur* in maturitetni predmet ima *70 + 210 ur*, pri čemer je *70 ur* namenjenih projektnemu delu.

Cilji vsebine predmeta so razporejeni na dve ravni:

- **splošna znanja**, v katerih dijaki razvijajo temeljnje digitalne kompetence, ki so potrebne za učinkovito uporabo digitalne tehnologije pri razvijanju lastnega znanja in za njegovo predstavitev oziroma posredovanje drugim;
- **posebna znanja**, s katerimi dijaki znanje, veščine, osebnostne in vedenjske značilnosti, prepričanja, motive in druge zmožnosti splošnega znanja spiralno nadgradijo.

Za nas pomembne enote najdemo na drugi ravni **posebnih znanj** v tematskem sklopu **obdelava podatkov**. Zbrali bomo cilje, ki izpostavljajo programiranje. Dijaki:

- **računalniška obdelava podatkov**
 - poznajo vlogo računalniškega programa in razložijo pomen programiranja;
- **algoritmi**
 - opredelijo algoritem in poznajo temeljne zahteve za algoritmom,
 - poznajo temeljne gradnike algoritma, razvijejo algoritmom za problem z vejiščem in zanko (do 15 gradnikov),
 - uporabijo diagram poteka in uporabljeno rešitev utemeljijo,
 - analizirajo algoritmom, ki reši zahtevnejši problem, in ga ovrednotijo;
- **programski jezik**
 - opredelijo programski jezik in razložijo njegovo funkcijo,
 - poznajo temeljne gradnike izbranega programskega jezika,
 - razložijo njihovo funkcijo in razlagajo ponazorijo s primeri,

- opredelijo strukturirano, objektno in dogodkovno programiranje,
- ločijo med prevajalnikom in tolmačem in razliko razložijo;
- **programiranje**
 - za dani algoritem izdelajo računalniški program,
 - opredelijo dokumentiranje programa in razložijo njegov pomen,
 - analizirajo program in ovrednotijo rezultate, dobljene s programsko rešitvijo.

2.4.2 Računalništvo - tehniška gimnazija

Cilje učenja programiranja pri tem maturitetnem predmetu najdemo pod poglavjem **Programski jeziki in programiranje**. Cilji so naslednji, dijaki [5]:

- poznajo pomen načrtovanja in sistematične gradnje programa,
- poznajo pojem algoritma in opredelijo njegove lastnosti (razumljivost, končnost, enoumnost, razčlenjenost),
- opišejo načine zapisa algoritma in jih prikažejo na primeru, izdelajo algoritem za lažji problem,
- opredelijo pojem programskega jezika in razložijo njegovo funkcijo,
- poznajo različne vrste programskih jezikov in jih razvrstijo po namenu in uporabi,
- poznajo temeljne gradnike postopkovnega programskega jezika, razložijo njihove funkcije in razlago ponazorijo s primeri (zaporedje, vejitev, iteracija),
- opredelijo strukturirano in objektno programiranje,
- ločijo med prevajanjem in tolmačenjem in razliko razložijo,
- poznajo osnovno razvojno okolje,
- poznajo pojme deklaracija, inicializacija, postopek, konstanta, spremenljivka, rezervirana beseda, operator, prioriteta ...
- ločijo med pojmi izvorna koda, izvršljiva koda in vmesna (byte) koda,
- pridobljeno znanje o tipih in algoritmih prenesejo v programski jezik in samostojno rešujejo probleme s pomočjo višjega programskega jezika Java,
- poznajo in uporabijo osnovne in sestavljeni tipe podatkov,
- uporabijo pogojna stavka (if, switch),
- iteracijo realizirajo z zankami (do, for, while),
- uporabijo tokove podatkov,
- pripravijo testne podatke, testirajo delovanje programa in beležijo rezultate testiranj,
- uporabijo razhroščevalnik,
- napišejo sled programa,
- izdelajo dokumentacijo programa,
- uporabijo metode za delo z objekti razredov Math, String, StringBuffer, Integer, Double
- ...

- napišejo definicijo razreda (lastnosti, metode ...),
- deklarirajo in uporabijo objekte,
- poznajo vlogo in način izvajanja konstruktorja,
- uporabijo enkapsulacijo, dedovanje in polimorfizem,
- poznajo načine za prestrezanje in obravnavo izjem,
- napišejo programe za preproste probleme iz okolja,
- analizirajo program in ovrednotijo rezultate, dobljene s programsko rešitvijo (različni algoritmi urejanja podatkov, različni načini zapisovanja podatkov v datoteke ...),
- predstavijo delovanje programa,
- analizirajo in kritično vrednotijo rešitve,
- *poznajo zahtevnejše tehnike programiranja,*
- *napišejo program z uporabo zahtevnejših tehnik programiranja.*

Cilji določajo zelo podrobno obravnavo snovi programskih jezikov in programiranja, ki na prvi pogled presega uporabo spletnih portalov za učenja programiranja.

3 Računalniška znanost in programiranje

Računalniška znanost ima številne razlike definicij, v grobem jo lahko strnemo v naslednjih trditvah [6].

- Ukvarja se z značilnostmi tistega, kar je izračunljivo.
- Je znanost, ki izhaja iz več področij in ima korenine v matematiki, znanosti in inženirstvu.
- Ima mnoga podpodročja in je interdisciplinarna z biologijo, ekonomijo, medicino, zabavo.
- Ime računalništvo ali računalniška znanost nas lahko tudi zavede in jo zamenjamo s področjem uporabe računalnika.

3.1 Zgodovina programskih jezikov

Uporaba računalništva v izobraževanju je bila deležna številnih sprememb. Sama uporaba računalnika v izobraževanju je tesno povezana z razvojem računalnikov. V začetnem obdobju, leta 1960 so bili računalniki zelo dragi in veliki glavni računalniki (*ang. mainframe*), na njih se je učilo programiranja, a so se uporabljali tudi za druga področja. V tem obdobju se je za učenje programiranja uporabljal **FORTRAN** ali **asembler**. Programi so bili majhni in preprosti zaradi fizičnih omejitev takratnega delovnega pomnilnika. Temu začetnemu obdobju pravimo **terminalsko obdobje** [1].

1970 so na trg prišli manjši računalniki, ki so bili tudi cenejši in zmogljivejši. V tem času pride v ospredje strukturirano programiranje. Najpopularnejši programski jezik je bil **PASCAL**. Predstavniki obdobja **mikroričunalnikov** so bili *Commodore 64, Sinclair ZX-81, Apple II*.

1980 so se prvič pojavili **samostojni osebni računalniki**. Programske jezike v tem obdobju so bili strukturirani in močnega tipa (*ang. strong type*). Med te spadajo **Ada, Modul 2, ML in Prolog**.

V naslednjem desetletju, 1990, so v ospredje prišli objektno orientirani programske jeziki, kot sta **C++ in Java** [7].

Metode poučevanja računalništva so se prav tako spreminali. 1960 so računalnike uporabljali samo za poučevanje programiranja. Poudarek pri predmetih programiranja je bil predvsem na detajlih zmožnosti programskega jezika. Programiranje je bilo omejeno le na reševanje preprostih primerov in poudarka na splošnem reševanju problemov ni bilo.

1970 sta reševanje problemov in abstrakcija podatkov postala glavni in najpomembnejši del vseh programerskih predmetov, kar velja še danes. Programi so postali večji, bolj interaktivni

in spremenil se je vnos podatkov iz tekstovnega v grafičnega. Vsebina predmetov računalništva se je hitro razširjala, kakor so se množili številni programski jeziki [7].

3.2 Osnovni pojmi

Preden nadaljujemo, moramo razjasniti nekaj osnovnih pojmov, ki se pojavljajo v računalniški znanosti.

3.2.1 Program

Računalniški program je zbirka navodil, ki opravlja točno določeno nalogu in jo izvajamo na računalniku. **Centralna procesna enota** je tista, ki izvajanje programa omogoča. Računalniški program navadno napiše **programer** v nekem **programskem jeziku**, postopku izdelave programa pravimo **programiranje**. Programskega jezika omogoča, da je program zapisan v takšni obliki, da je berljiv za ljudi. Shranjeni datoteki v taki obliki pravimo, da je **izvorna datoteka**, zapisana v **izvorni kodi**. Da računalnik razume napisan program v izvorni kodi, ga mora prevesti **prevajalnik (ang. compiler)** v **strojno kodo** ali ga sproti prevaja **tolmač**. [8].

3.2.2 Algoritem

Z algoritmom ponazarjamо postopek, ki je zgrajen iz posameznih operacij, ki se izvajajo po posameznih korakih. Algoritem daje rešitev za izračune, procesiranje podatkov, avtomatizacijo postopkov. Beseda “*algoritem*” izvira iz imena **Al-Khwārizmī**, perzijskega matematika, astronoma, geografa in učenjaka [9].

Pri pregedu učnih načrtov v poglavju 2.3.2 smo lahko zasledili cilje, kot so:

- **znajo vsakdanji problem opisati kot zaporedje korakov,**
- **znajo z algoritmom predstaviti preprosto opravilo,**
- **algoritem predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku.**

Prikazali bomo več različnih predstavitev algoritma, najprej bomo rešitev zapisali v besedilni obliki, zatem bomo rešitev podali z diagramom poteka, na koncu bomo rešitev podali še s programskima jezikoma Scratch in Python. Za primer bomo prikazali primer algoritma, ki reši naslednjo nalogu.

Primer 1: Algoritem - Najdi največje število

Algoritem med podanimi števili poišče največje število.

Postopek, zapisan v **besedilu**:

1. Zapišemo prvo **največje število**, ki naj bo na začetku enako 0.
2. Števila v seznamu pregledujemo po vrsti tako, da vsako število primerjamo z **največjim številom**, in če je število s seznama večje,
3. to trenutno **največje število** prečrtamo in napišemo tisto s seznam, ki je novo največje število.
4. Postopek ponavljamo, dokler ne pridemo na konec seznama.

Podprogram v **Scratchu**:



Podprogram v **Pythonu**:

```
1 def najdiNajvecjeStevilo(seznam):  
2     """Funkcija poišče največje stevilo v seznamu."""  
3     i = 0  
4     najvecje_stevilo = 0  
5     for i in range(len(seznam)):  
6         if seznam[i] > najvecje_stevilo:  
7             najvecje_stevilo = seznam[i]  
8     return najvecje_stevilo
```

3.2.3 Programiranje in kodiranje

Izraz programiranje smo že spoznali. Velikokrat na spletu zasledimo tudi izraz **"kodiranje"** (**ang. coding**). Izraz pomeni zapisovanje programa s programskega koda izbranega programskega jezika, ki predstavlja končni izdelek oziroma izvorno datoteko. Lahko rečemo, da je za vsako kodiranje odgovorna oseba, ki ji pravimo **"koder"**. Podobno, kot smo uporabili izraz programer za osebo, ki skrbi za razvoj in izdelavo programa. Programer in koder sta velikokrat dana v isto vlogo, vendar njune naloge niso enake, saj je programer nekdo, ki načrtuje

rešitve na različne načine in z različnimi orodji. Naloga koderja je, da rešitve zapiše na najbolj učinkovit način v programskem jeziku, ki ga zares dobro pozna. Poklica sta zelo povezana in so meje med njima tudi pogosto zbrisane, kar še posebej velja, če je ista oseba v vlogi programerja kot tudi koderja [10].

Pri poučevanju računalništva lahko menimo, da je v prvi vrsti pomembno to, da se uči strategije reševanja problemov, zato mora obstajati težnja, kako poučevati, da bo čim več učencev postalo predvsem dobrih programerjev.

3.2.4 Urejevalnik besedil

Dober urejevalnik besedil lahko programerju lajša delo s številnimi zmožnostmi. Opisali smo nekatere, saj nam bodo ti pomagali lažje prepoznati dober urejevalnik besedil.

- **Barvanje rezerviranih besed prog. kode** je značilno za skoraj vsak novodobni urejevalnik besedil. Z različnimi barvami je olajšano branje programske kode.
- **Samodejno zamikanje vrstic programske kode**. Urejevalnik, ki ima to zmožnost, zna prepoznati, ko sledi nov del programske kode, ki je zamknjen, npr. za stavkom `if/else`.
- **Ponujanje predlog za samodokončanje rezerviranih besed in funkcij prog. jezika**. Ob pisanju programske kode urejevalnik ponuja dokončanje programske kode.
- **Izpis opisa funkcij z atributi**.
- **Samodejno zaključevanje oklepajev**. Odprti oklepaj se samodejno zaključi z zaprtim in smernik za pisanje se postavi v sredino med oba oklepaja.

3.2.5 Integrirano razvojno okolje

Del **integriranega razvojnega okolja (IRO)** (*ang. Integrated development environment IDE*) je dober urejevalnik besedil, kot smo ga opisali v prejšnjem poglavju. Za IRO je značilno, da omogoča **pisanje, testiranje, razhroščevanje in prevajanje končnega programa**. Omogoča povezavo med datotekami projekta in knjižnicami. Svoje zmožnosti nadgrajuje s številnimi zunanjimi orodji, ki jih navadno vklopimo preko vtičnikov. Slabost IRO je ta, da so to veliki programski paketi, ki včasih znajo biti okorni in počasni. Pri nekaterih obsežnih IRO je tudi čas, ki ga potrebujemo, da se ga naučimo uporabljati, dolgotrajen.

3.3 Programske paradigmme

Paradigma je način, kako obravnavamo in gledamo na stvari, je okvir, v katerem leži naša interpretacija realnosti sveta. Paradigma najpogosteje pomeni vzorec delovanja v znanstvenem ali drugem raziskovanju. Izraz programska paradigma je večpomenka, ki povzema mentalne procese, strategije reševanja problemov, povezave med različnimi paradigmami, programske jezike, stil programiranja in še več [11] [6].

Programske paradigmme so hevristike, ki se uporabljajo za reševanje problemov. Programska paradigma analizira problem skozi specifični pogled in na ta način formulira rešitev za dani problem, ki ga razdeli na manjše dele, med katerimi definira razmerja.

Programske paradigmme so na primer proceduralno, objektno orientirano, funkcionalno, logično in istočasno programiranje. V nadaljevanju spoznamo značilnosti programske paradigmme objektnega programiranja, saj je ta v zadnjih dveh desetletjih najbolj razširjena.

3.3.1 Objektno orientirano programiranje

Objetno orientirano (**OO**) programiranje je način, kako programerji razmišljajo o svojem delu. Princip OO model realnosti sveta predstavlja v **razredih ang. class**. S takim načinom zapisa programske kode je ramišljanje o programu dosti bolj naravno [12].

Objekt je predstavnik različnih stvari, ki jih želimo predstavljati v programske razredu. Te stvari so lahko karkoli, od realnih objektov in vse do konceptov. Podajmo primer objekta mačke. Mačka ima številne karakteristike, kot so barva, ime, teža ..., tem lastnostim pravimo, da so lastnosti objekta. Mačka je živo bitje, zato njenemu početju, kot je mjavkanje, spanje, igranje ..., pravimo, da so metode objekta. Pri objektih lahko uporabimo analogijo in objekte poimenujemo s samostalniki, metode so glagoli in vrednosti lastnosti objekta so pridevniki. V nadaljevanju si bomo ogledali nekatere značilnosti, ki definirajo programski jezi kot OO [13].

Razred (ang. Class): lahko v resničnem življenju objekte združuje po nekih določenih kriterijih. Orel in sinička sta oba ptiča, zato jih lahko damo skupaj v razred, ki ga poimenujemo **Ptiči**. Razredi so načrti ali recepti za objekte, tako lahko ustvarimo več objektov iz istega razreda, saj je razred le shema.

Enkapsulacij (ang. Encapsulation): je koncept, ki predstavlja, da so podatki, torej *lastnosti* objektov in opravila, ki jih lahko opravljajo ali *metode* objektov, združeni.

Združevanje (ang. Aggregation): pomeni, da lahko več objektov združimo v en objekt. To predstavlja močno orodje pri razčlenjevanju problemov na manjše podprobleme.

Dedovanje (ang. Inheritance): je eleganten način, kako porabimo eno kodo večkrat. Podajmo primer, imamo splošen razred **Oseba**, ta ima lastnosti, kot so ime, datum rojstva,

in ima napisane metode, ki prestavlja funkcionalnost, kot je, da Oseba lahko govori, hodi, je, spi. Zatem bi želeli bolj specifičen razred, kot je Programer. Lahko bi vso kodo ponovno napisali in ji dodali specifično za programerja. Dedovanje omogoča, da povemo, da Programer deduje od razreda Osebe in si tako prihranimo velik del dela.

Polimorfizem (ang. polymorphism): je način, kako lahko isto ime metode uporablja več različnih razredov in posledično objektov, ne glede na to, da je najverjetneje koda v njem različna.

Programsko paradigma OO programiranja smo povzeli na kratko, da bi lažje razumeli, zakaj je ta programska paradigmata tako popularna. V nadaljevanju bomo spoznali, da je večina programskih jezikov, ki se uporabljajo dandanes, OO ali vsaj vsebujejo nekaj lastnosti OO programskih jezikov.

3.4 Programske jeziki

V tem poglavju bomo povzeli osnovne značilnosti posameznih programskih jezikov. Če na spletu v spletnem iskalniku podamo zahtevo po najpopularnejših programskih jezik, dobimo podobne rezultate večih spletnih strani¹ ²³, in sicer: **JAVA, C, C++, Python, C#**, v najbolj priljubljenih 10 za nas pomembne najdemo še **Java Script**.

Programski jeziki v izobraževanju so se skozi zgodovino menjali, tako kot se je rezvijala računalniška znanost, kar smo že povzeli v poglavju 3.1. Zanima nas, kateri so programski jeziki, ki so najbolj primerni za uporabo učenja programiranja in se uporabljajo danes.

Vsak programski jezik bomo s kratkim primerom tudi predstavili s primerom programske kode, tako bomo dobili lažjo predstavo, kakšna je osnovna razlika v sintaksi.

Večina od zgoraj naštetih programskih jezikov je **OO**, z izjemo **C**-ja, ki je predhodnik **C++**. Za nas bodo z izobraževalnega vidika zanimivi predvsme tisti, ki se uporabljajo pri splošnem izobraževanju pri nas.

3.4.1 Java

Java je večnamenski programski jezik, njegova osnova so *razredi* in je OO. Njegova glavna prednost je, da lahko napisano kodo zaganjam ne glede na platformo, na kateri teče, torej so napisani programi neodvisni od operacijskega sistema, ki ga poganja računalnik. Zato je za vsako platformo prilagojen **virtualni stroj za Java** (ang. *Java Virtual Machine (JVM)*), ki

¹Pridobljeno 27. 4. 2016 iz, http://www.tiobe.com/tiobe_index.

²Pridobljeno 27. 4. 2016 iz, <http://github.info/>.

³Pridobljeno 27. 4. 2016 iz, <http://pypl.github.io/PYPL.html>.

prevedeno kodo poganja. Sintaksa programskega jezika je zelo podobna C++. Popularnost programskega jezika se je še izboljšala zaradi operacijskega sistem za tablice in telefone **Android**, ki prav tako teče na različici (JVM) oz. so programi napisani v Javi [14].

V izobraževanju je Java postala zelo priljubljena prav zaradi zmožnosti poganjanja programov na različnih platformah. Uporablja se na primarnem področju izobraževanja, predvsem v srednjem in visokem šolskem področju. V sekundarnem področju izobraževanj se je uporabila predvsem za pisanje programske opreme, ki dopolnjuje izobraževanje, kot so fizikalne simulacije (*fizleti*) in podobno. Eden od razlogov, da se je Java v preteklosti na tem področju dobro uveljavila, je tudi ta, da omogoča zagon aplikacije iz spletnega brskalnika, vendar je za to potrebna namestitev posebnega vtičnika, ki to omogoča. V zadnje času je zaradi varnosti praktično nemogoče zaganjati Java applete, zato je vedno manj primerna za uporabo v spletnem brskalniku. Primer 2 prikazuje sintakso "*Dobrodošel svet!*", napisano v Javi.

Primer 2: Program napisan v Javi

```
1 class First {
2     public static void main(String[] arguments) {
3         System.out.println("Dobrodošel svet!");
4     }
5 }
```

3.4.2 C++

C++ je vsenamenski programski jezik, ki je OO in je bil zasnovan kot sistemski programski jezik. Večina operacijskih sistemov je danes napisana v kodi C++ in predhodniku C. Kodo programskega jezika mora prevesti prevajalnik, preden jo lahko zaganjam, za vsako platformo moramo prevajati posebej. C++ velja za najhitrejši programski jezik, z njim lahko opravljamo tako naloge, kot je neposredni nadzor nad polnilnikom, kot tudi vse višje funkcije, ki jih omogoča. Zato velja za enega težje učljivih programskih jezikov. Programiranje v C++ se uči predvsem na višjem in univerzitetnem izobraževalnem nivoju [15].

Primer 3: Program napisan v C++

```
1 #include <iostream>
2 int main()
3 {
4     std::cout << "Hello World!" << std::endl;
5     return 0;
6 }
```

3.4.3 Java Script

Java Script (JS) se je razvil za potrebe po bogatejših in dinamičnih spletnih straneh. Začetek spletja so predstavljeni statični dokumenti, ki so bili povezani s hiperpovezavami. Skriptni jezik Java script se je prvič pojavi s spletnim brskalnikom *Netscape 2.0*. Takrat je bilo možno vstavljanje kratkih odsekov kode, ki so spletne strani naredile dinamične. Težnja po standardizaciji skriptnega jezika se je pojavila, ko se je na trgu pojavil *Internet explorer 3.0*, saj je ta imel svojo različico skriptnega jezika *JScript*. Sedaj se standardni jezik imenuje **ECMA Script** oz. točneje **ECMA-262** [13].

Če smo v prejšnjih poglavjih govorili, da sta bila **Java** in **C++** večnamenska jezika, je bil JS eden tistih, ki so tekli znotraj vgrajenega gostiteljskega okolja, kot je spletni brskalnik. Danes imamo tudi okolja, ki omogočajo, da JS teče na strežnikih, namizju in mobilnih napravah. Torej, kljub zgoraj omenjeni omejitvi, postaja prav tako večnamenski skriptni programski jezik. V spodnjem primeru 4 imamo primer programa „*Dobrodošel svet!*“ s **HTML** ogrodjem. Tako programsko kodo odpremo v spletnem brskalniku. Del skriptnega jezika se začne z značko `<script type="text/javascript">` JS koda `</script>`. Povejmo še to, da se koda programskega jezika ne prevaja, temveč jo poganja **tolmač**.

Primer 4: Program napisan v JavaScriptu + HTML ogrodje

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <title>Some Page</title>
5     <script type="text/javascript">
6       alert("Hello World!");
7     </script>
8   </head>
9   <body>
10    <p>The content of the web page.</p>
11  </body>
12 </html>
```

3.4.4 Python

Python je zelo pogosto uporabljen večnamenski programski jezik. Njegovo kodo, podobno kot JS, poganja tolmač. Zasnovan je tako, da je koda čim bolj berljiva in njegova sintaksa omogoča, da programske koncepte zapišemo v čim manj vrsticah, kakor bi jih lahko v Javi ali C++. Če so posamezni odseki ali bloki programske kode pri Javi in C++ označeni z zavitimi

oklepaji ({}), jih v Pythonu izrazimo s tabulatorskim zamikom. V Pythonu lahko uresničimo več programskih paradigem, kot je OO ali proceduralno programiranje. Omogočen je dinamičen tip spremenljivk, ima urejeno samodejno upravljanje s pomnilnikom in ima veliko standardno knjižnico [16].

Python se veliko uporablja tudi v izobraževalne namen. Pri nas se priporoča za začetke učenja programskega jezika na srednjem šolskem izobraževalnem nivoju. Zaredi tega ga bomo v diplomskem delu uporabljali kot glavni demonstracijski programski jezik.

Primer 5: Program napisan v Pythonu HTML ogrodje

```
1 print (''Hello world!'')
```

3.5 Osnovni koncepti programiranja

V naslednjem odstavku se bomo vprašali, kako lahko formuliramo sintakso programskega jezika. In kaj je npr. definicija *kopice*. V ta namen definiramo mehko idejo po Hazzanu [6]. Mehka ideja je koncept, ki mu ne moremo pripisati toge, niti formalne definicije. Mehke ideje ni niti možno opisati s točno določeno aplikacijo. Na tem mestu se postavlja vprašanje, kako lahko defineramo nekaj, kar se odvija po korakih.

Da odgovorimo na zgornji vprašanji, lahko povemo, da so pravila sintakse togorisi pri pisanju programske kode in da so semantična pravila mehke ideje. Opozorimo še na to, da koncepti v računalniški znanosti niso le toga pravila ali samo mehke ideje, temveč skupek obojega. V spodnji tabeli 2 prikazujemo primer spremenljivke.

Tabela 2: Prikaz dvojnih, togih in mehkih orisov idej na primeru spremenljivke [6].

	Togi orisi	Mehki orisi
Ime spremenljivke	Pravilo sintakse.	Potreba po imenu spremenljivke. Katero ime spremenljivke je pomembno in zakaj ga je potrebno določiti.
Vrednost spremenljivke	Pravila tipa spremenljivke. Rezervacija pomnilnika.	Spremenljivka ima eno vrednost, ki se lahko spreminja s časom.
Dodelitev začetne vrednosti	Pravila sintakse.	Pomen dodelitve začetne vrednosti.

V nadaljevanju bomo pregledali in skušali razložiti osnovne koncepte programiranja. S primerom bomo pokazali enega izmed načinov, kako jih prestavimo. Za vodilo bomo uporabili učni načrt za OŠ, ki smo ga pregledali v poglavju 2.3.2 in SŠ, ki je v poglavju 2.4. Primeri

programov, ki jih bomo uporabljali in prilagodili, so povzeti iz knjige in s spletno strani “*Learning python the hard way*” [17].

3.5.1 Izpis in računske operacije

Osnovno interakcijo z računalnikom lahko opišemo na naslednji način. Računalniku damo vhodne podatke, ta podatke po navodilu programa obdelava in nam poda rezultate na izbrano izhodno napravo. Ta izhodna naprava je na primer zaslon in na njem se izpisujejo obdelani podatki. Izpišimo nekaj stakov, pri tem uporabljamo ukaz `print`.

Primer 6: Izpis besedila na zaslon | 01_izpis_na_zaslon.py [17]

Navodilo naloge: Sledi navodilu, ki je zapisano v programske kodi.

```
1 #Del programa, ki nocemo, da ga uposteva tolmac,  
2 #oznacimo z # in ga imenujemo komentar.  
3 print "Pozdravljeni, to je nas prvi izpis na zaslonu."  
4 print "Izpisemo lahko tudi prazno vrstico. \n"  
5 #Za izpis prazne vrstice uporabimo "\n"  
6 print "Pred to vrstico je prazna in za njo.\n"
```

```
$ python 01_izpis_na_zaslon.py  
Pozdravljeni, to je nas prvi izpis na zaslonu.  
Izpisemo lahko tudi prazno vrstico.
```

Pred to vrstico je prazna in za njo.

Ena izmed glavnih nalog računalnikov so računske operacije, zato si poglejmo dva primera izračunov v programskega jezika Python.

Primer 7: Računske operacije | 02_racunske_operacije.py [17]

Navodilo naloge: Sledi navodilu, ki je zapisano v programske kodi.

```
1 #Izračunajmo naslednje izraze in izpišimo njihov rezultat.  
2 #Preden poženemo program, izračunajmo vrednost sami.  
3 print 100 - 5%2 + 3*4 - 22/3  
4 print 4+7 > 13
```

```
$ python 02_racunske_operacije.py  
104  
False
```

3.5.2 Spremenljivke

Spremenljivke so način, kako shranujemo podatke v računalniku. Ime spremenljivke ima podobno vlogo kot imena ljudi ali stvari v vsakdanjem življenju. Ljudje in stvari imajo imena zato, da si jih lažje zapomnimo in se z njimi in o njih lažje pogovarjamo. Podobno je tudi v programiranju, zato si moramo izbrati dobra imena spremenljivk 8. V tem poglavju bomo uresničili naslednje cilje, ki smo jim nekoliko spremenili vrstni red:

- **znajo v program vključiti spremenljivke in jim izpisovati vrednosti med izvajanjem programa in izpisati končni rezultat,**
- **znajo spremenljivkam spremeniti vrednost s prireditvenim stavkom,**
- **razumejo različne podatkovne tipe in jih znajo uporabiti v programu,**
- **znajo v programu prebrati vhodne podatke in jih vključiti v program.**

Primer 8: Izpis besedila na zaslon | 03_uporaba_spremenljivk.py [17]

Navodilo naloge: Na parkirišču je 100 avtomobilov, vsak izmed avtomobilov ima 5 sedežev. S temi avtomobili želimo pripeljati 90 potnikov, od tega jih ima 30 vozniško dovoljenje. Izračunaj in izpiši naslednje podatke.

- Koliko avtomobilov bo ostalo na parkirišču, če bodo vozili vsi šoferji?
- Koliko ljudi lahko prepeljemo z vsemi avtomobili?
- Kakšno je povprečno število potnikov, če vozijo vsi vozniki?

```
1 #1. Določimo spremenljivke:  
2 avtomobili = 100  
3 prostor_v_avto = 5.0  
4 potniki = 90  
5 šoferji = 30  
6 #2. Izračunajmo vrednosti in jih shranimo v spremenljivke:  
7 avtomob_na_park = avtomobili - soferji  
8 kapac_avtomob = avtomobili * prostor_v_avto  
9 avtomob_na_potnikov = potniki/prostor_v_avto  
10 povpr_st_potnikov = potniki/soferji  
11 #3. Izpišimo vse zahtevane podatke.  
12 print ("Na parkirišču bo ostalo",avtomob_na_park, "avtomobilov.")  
13 print ("Z vsemi avtomobili prepeljemo", kapac_avtomob, "potnikov.")  
14 print ("Povpr. st. potnikov je",povpr_st_potnikov, ", če vozijo vsi  
šoferji.")
```

```
$ python 03_uporaba_spremenljivk.py  
Na parkirišču bo ostalo 70 avtomobilov.  
Z vsemi avtomobili lahko prepeljemo 500.0 potnikov.  
Povprečno število potnikov je 3, če vozijo vsi šoferji.
```

3.5.3 Pogojni stavki in vejitve

V tem poglavju bomo uresničili naslednje cilje: **znajo uporabiti pogojni stavek in izvesti vejitev.**

Primer 9: Program odločitve | 05_pogojni_stavki.py [17]

Navodilo naloge: Število potnikov spremeni tako, da bo kombi dovolj za vse potnike.

```
1 potniki = 7
2 if potniki <= 5:
3     print ("Pojdite z avtomobilom.")
4 elif potniki > 5 and potniki <= 9:
5     print ("Pojdite s kombijem. ")
6 else:
7     print ("Ne morem se odločiti.")
```

```
$ python 05_pogojni_stavki.py
Pojdite s kombijem.
```

3.5.4 Zanke

V tem poglavju bomo uresničili naslednje cilje: **razumejo pojem zanke in ga znajo uporabiti za rešitev problema.**

Primer 10: Zanka | 06_zanka.py [17]

Navodilo naloge: Napišite zanko, ki prešteje ter izpiše vsa liha števila.

```
1 #Podan seznam celih števil.
2 cela_st = [11, 41, 2, 32, 12]
3 st_lihih = 0
4 for st in cela_st:
5     if st%2 == 1:
6         print (st)
7         st_lihih += 1
8 print ("Vseh lihih števil:", st_lihih)
```

```
$ python 06_zanka.py
11
41
Vseh lihih števil: 2
```

4 Pristopi in strategije poučevanja

V naslednjem poglavju raziščemo, kaj so sodobni pristopi in značilne strategije pri poučevanju računalniške znanosti in programiranja. Izpostavili bomo **model aktivnega učenja** in **strategijo reševanja problemov**. Z razumevanjem tega bomo v nadaljevanju lažje ocenili, ali uporaba spletnih portalov vzpodbuja oba pristopa.

4.1 Model aktivnega učenja

Vsak pouk računalništva naj bi imeti modelno zgradbo in bi upošteval naslednji načeli:

- naj vzpodbuja študente s pozitivno naravnanim poukom in jim naj omogoča okolje, kjer najdejo pomoč.
- Pouk računalništva naj bo grajen na konstruktivnih metodah poučevanja in aktivnem učenju.

Konstruktivizem je kognitivna teorija, ki preučuje naravo procesov učenja. Po tem principu naj bi učenci konstruirali novo znanje na osnovi preurejanja in izpopolnjevanja že obstoječega znanja. Znanje se gradi na obstoječih mentalnih strukturah in na odzivu, ki ga dobi učenec iz učnega okolja. Mentalne strukture so grajene korak za korakom, ena za drugo, lahko pa s to metodo pride tudi do sestopanja ali slepih koncev. Proses je povezan s Piagetovim mehanizmom asimilacije [6].

Pri **aktivnem učenju** je najpomembnejše to, da učenci z lastno aktivnostjo ugotovijo, kako nekaj deluje. Sami si morajo izmisli primere, preiskusiti lastne veščine in reševati neloge, ki so jih že ali jih še bodo spoznali. Učenje je aktivno usvajanje, je gradnja idej in znanja. Za učenje mora biti posameznik aktivno vključen v gradnjo svojih lastnih mentalnih modelov. Model aktivnega učenja je sestavljen iz štirih korakov [6].

- **Sprožilec** je naloga, ki predstavlja iziv za uvod v novo tematiko.
- **Aktivnost.** Študenti izvajajo aktivnost, ki jim je bila predstavljena v sprožilcu. Ta korak je lahko kratek ali lahko zavzame večji del učne ure. To je odvisno od vrste sprožilca in izobraževalnih ciljev.
- **Diskusija** sledi po koncu aktivnosti, kjer se zbere celoten razred, ne glede na obliko dela. V tem koraku študenti izpopolnijo koncepte in ideje kot del konstruktivnega učnega procesa.
- **Povzetek** je lahko izražen v različnih oblikah, kot so zaokrožene definicije, lahko so miselni vzorci ali povezave med temami, ki so jih obravnavali študenti, in med drugimi temami, ki se navezujejo nanje.

Ko se ta model izkaže za primernega, ga lahko uporabimo v številnih učnih urah v različnih variacijah. Zanjo nas, ali znajo pristop aktivnega učenja spletni portali upoštevati.

4.2 Strategije reševanja problemov

Programiranje je proces, pri katerem rešujemo probleme, zato je reševanje problemov v središču poučevanja računalniške znanosti. Reševanje problemov je zahteven mentalni proces. Če na spletu poberemo za strategije reševanja problemov, lahko hitro ugotovimo, da obstajajo različne strategije. Med njimi lahko najdemo **abstrakcijo**, **analogijo**, **brainstorming**, **deli in vladaj** in mnoge druge [18]. Proces in tehnike reševanja problemov se uporablja v mnogih tehničnih in znanstvenih disciplinah [6].

V nekaterih primerih učenci sami razvijejo strategijo, s katero rešijo nek problem. Otroci si na primer sami izmisljijo preprosto seštevanje in odštevanje, še preden se to učijo pri pouku matematike. Toda brez formalne podpore za učinkovito strategijo reševanja problemov spodleti še tako inovativnemu učencu tudi pri preprostih strategijah, kot je **preizkus in napaka**. Zato je pomembno, da se uči strategij za reševanje problemov.

Vsak osnovni proces, ki se ukvarja z reševanjem problemov, ne glede na znanstveno disciplino, se začne z opisom problema. Vsak problem se navadno zaključi z neko rešitvijo, ki je v nekaterih primerih izražena z **zaporedjem korakov** ali **algoritmom**. V računalnik algoritem zapišemo s kodo nekega programskega jezika. Zapisan algoritem testiramo tako, da kodo prevedemo v strojni jezik in jo izvedemo. Za pravilno delovanje programa primerjamo vhodne in želene izhodne podatke. Preden pridemo od opisa problema do podane rešitve, moramo prehoditi kar nekaj težkih korakov. Na te vmesne korake lahko gledamo kot na procese odkrivanja, zato lahko na reševanje problemov gledamo tudi kot na kreativen, umetniški proces. Splošno priznani koraki reševanja procesov so naslednji [6]:

1. *Analiza problema.* Najprej je pomembno, da razumemo, kaj je problem in ga znamo identificirati. Če tega ne znamo, ne moremo priti do nobene rešitve.
2. *Alternativne rešitve.* Razmišljamo o alternativnih rešitvah, kako bi lahko rešili nek problem.
3. *Izbira pristopa.* Izberemo primeren pristop, kako rešiti problem.
4. *Razgradnja problema.* Problem razgradimo na manjše podprobleme.
5. *Razvoj algoritma.* Algoritem razvijamo po korakih, ki smo jih določili v podproblemih.
6. *Pravilnost algoritma.* Preverjanje pravilnosti algoritma.
7. *Učinkovitost algoritma.* Izračunamo učinkovitost algoritma.
8. *Refleksija.* Naredimo refleksijo in analizo za pot, ki smo jo naredili pri reševanju problema in naredimo zaključek s tem, kar lahko izboljšamo za naslednji problem, ki ga bomo reševali.

Točen recept, kako se lotiti reševanja, ne obstaja. Učencem lahko le pokažemo nekatere metode in strategije, ki jim lahko pomagajo pri reševanju problemov. Da bi bolje znali oceniti, kako izrazito spletni portali upoštevajo korake strategije reševanja problemov, poglejmo še podrobnejše nekatere pomembne korake in kako se z njimi spopadajo novinci.

4.2.1 Razumevaje problema

Razumevanje problemov je prva stopnja v procesu reševanja problemov. Pri reševanju algoritemskih nalog moramo najprej prepoznati, kaj so vhodni podatki in kateri podatki naj bi bili izhodni. Če znamo povedati, kaj bodo vhodni podatki, razumemo tudi bistvo samega problema.

4.2.2 Načrtovanje rešitve

Novinci se spopadajo z največjimi težavami na začetni stopnji načrtovanja rešitve za nek problem. V nadaljevanju so predstavljene tri strategije, ki jih lahko uporabimo pri tem koraku reševanja problema.

Definicija spremenljivk problema: Pri rešitvi problema si pomagamo tako, da ugotovimo, kaj morajo biti vhodni in kateri bojo izhodni podatki. S tem razjasnimo problem. V naslednjem koraku definiramo **spremenljivke**, ki so potrebne za rešitev problema.

Postopno izboljševanje (ang. *Stepwise Refinement*): Po tej metodi nas najprej zanima celoten pregled strukture problema in odnosi med posameznimi deli. Zatem se šele poglobimo v specifično in kompleksno implementacijo posameznih podproblemov. Postopno izboljševanje je metodologija, ki poteka od **zgoraj navzdol**, torej od splošnega k specifičnemu. Drugačen pristop je od **spodaj navzgor**. Za oba pristopa velja, da drug drugega dopolnjujeta. V obeh primerih je problem razdeljen na manjše podprobleme ali naloge. Glavna razlika med obema je mentalni proces, ki je potreben za en ali drugi pristop. V nadaljevanju se posvetimo samo pristopu od **zgoraj navzdol**. Rešitev, ki jo poda **postopno izboljševanje**, ima modularno obliko, ki:

1. jo lažje razvijamo in preverjam,
2. jo lažje beremo in
3. nam omogoča, da uporabljam posamezne podrešitve tudi za reševanje drugih problemov.

Algoritemski vzorci: Združujejo matematični pogled in elemente načrtovanja. Vzorec podaja načrt na rešitev, s katero se srečamo mnogokrat. Algoritemski vzorci so primeri elegantnih in učinkovitih rešitev problemov in predstavljajo abstraktni model algoritemskega procesa, ki ga lahko prilagodimo in ga integriramo v rešitve drugim problemom.

Pri tem procesu lahko nastopi težava prepozname vzorca algoritma pri novincih, saj ti niso sposobni prepoznati podobnosti med posameznimi algoritmi ali ne znajo prepozname bistva problema, njihove posamezne komponente in razmerja med njimi, da bi lahko rešili nove probleme. V takih primerih novinci radi ponovno izumijo že njim poznane rešitve, ki bi jih lahko uporabili. Te težave navadno nastanejo zaradi slabe organizacije sistematike znanja o algoritmih.

Proces reševanja problemov z algoritemskim vzorcem se navadno začne s prepoznamenjem komponent, ki vodijo k rešitvi, in iskanjem podobnih problemov, na katere že imamo znane rešitve. Zatem vzorec prilagodimo rešitvi problema in ga vstavimo v celotno rešitev. V večini primerov je treba vstaviti več različnih vzorcev, da dobimo neko novo rešitev.

4.2.3 Preverjanje rešitve

Ko imamo pripravljeno rešitev, moramo preveriti, ali je ta pravilna. Pogled na preverjanje pravilnosti rešitve je lahko teoretične in praktične narave. Razhroščevanje (*ang. debugging*) spada me vrsto aktivnosti, ki nam pomaga pri ugotavljanju pravilnosti rešitve. Splošno velja, da proces razhroščevanja s programom, ki nam pomaga razhroščevati (*ang. debugger*), ali brez njega poglablja razumevanje računalniške znanosti. S tem ko učenci razmišljajo, kako bodo preverjali, ali njihov program deluje pravilno, hkrati v njih poteka miselni proces refleksije o tem, kako so implementirali določen program in kako ga bojo morebiti morali spremeniti.

Na nivoju do srednje šole uporabljam praktične metode ugotavljanja pravilnosti programa, kot je razgloščevanje. Ko želimo znanje pravilnosti delovanja poglobiti, se lahko lotimo tudi teoretične analize.

4.2.4 Refleksija

Refleksija je mentalni proces ali obnašanje, ki nam omogoča, da neko delovanje analiziramo in o njem tudi premislimo. Refleksija je pomembno orodje v splošnem učnem procesu, prav tako spada med kognitivne procese višjega reda. Z refleksijo učenec dobi priložnost, da stopi korak nižje in premisli o svojem razmišljanju in tako izboljša večino reševanja problemov. Refleksivno razmišljanje je proces, ki zahteva veliko časa in vaje. Med procesom reševanja problemov lahko refleksijo uporabimo na različnih stopnjah.

- *Pred* reševanjem problemov. Ko problem preberemo in že načrtujemo rešitev, se splača uporabiti refleksijo in razmisli o tem, ali smo morda že reševali podoben problem in temu primeren vzorec algoritma.

- *Med* reševanjem problema. Ko rešujemo problem, refleksija služi kot pregled, kontrola in nadzor. Na primer, ko nastopijo težave pri načrtovanju rešitve ali morda zaznamo težavo ali napako. Temo procesu lahko pravimo **refleksija v akciji**.
- *Po* reševanju problema. Ko že najdemo rešitev, ki deluje, nam refleksija služi kot orodje, s katerim pregledamo učinkovitost delovanja. Pregledamo strateške odločitve, ki so bile sprejete med načrtovanjem rešitve.

Refleksija je kreativni proces in je pomemben tako za učenca kot za učitelja.

5 Spletni portali za učenje programiranja

Spletne portale za učenje programiranja (**SPUP**) bomo predstavili in spoznali tako, da bomo najprej pregledali, kaj so bili glavni razlogi, da so se pojavili. Spletni portali za učenje programiranja, v nadaljevanju **SPUP**, so nastali takoj po razmahu interneta v začetku novega tisočletja. Najprej so nastali na univerzah. Zanima pa nas, kaj so glavni razlogi za nastanek SPUP. Poleg tehnoloških zmožnosti IKT za nastanek spletnega portala nas zanimajo predvsem težave, ki so jih skušali premostiti z uporabo SPUP.

Spletni portali so nastali na različnih univerzah, ogledali si bomo spletni portal, ki je nastal na *Odprti univerzi v Hong Kongu* (**OUHK**), na *Univerzi Strathclyde v Veliki Britaniji* (**USVB**) in *Queensland University of Technology, Australia* (**QUTA**).

Zanimali nas bodo predmeti, ki veljajo za začetne pri poučevanju računalniške znanosti in programiranja. **Novinci**, kot jih bomo imenovali, so študenti, ki se šele začnejo učiti programiranja. V diplomskem delu nas zanimajo le učenci osnovnih šol in dijaki srednjih, vendar se oni prav tako šele srečujejo s programiranjem, podobno kot študentje, in jih bomo zato vse poimenovali kot **novince**.

Kot je razvidno iz literature, bomo lahko sklepali na nekatere skupne značilnosti vseh novincev, ne glede na težavnostno stopnjo, na kateri se nahajajo, saj je programiranje veščina, ki ni dana naravno in se je mora vsak priučiti.

5.1 Razlogi za nastanek spletnih portalov

Na Odprti univerzi v Hong Kongu (**OUHK**) ponujajo tri računalniške sklope različnih težavnosti za dodiplomske programe. Imajo zelo veliko populacijo študentov, ki se učijo programiranja. Avtorji članka [19] ugotavljajo, da je proces učenja programiranja kompleksen in zahteva veliko vaje. Izkaže se, da praktični del iga poglobitno vlogo v učnem procesu.

Glavna težava, s katero se srečujejo na **OUHK**, je ta, da se število študentov, ki se vpišejo v smeri računalništva, povečuje. Povečanje študentov pomeni manj časa za mentorstvo za posameznega študenta.

Da bi študentje lahko normalno sledili pouku na daljavo, si morajo doma urediti delovno okolje, kjer lahko programirajo. Študenti, dobijo vso potrebno učno literaturo in tudi programsko opremo, ki predstavlja **prevajalnik** in **razvojno okolje**. Izkaže se, da imajo številni težavo nastaviti in se spoznati z integriranim razvojnimi okoljem (ang. *Integrated Development Environment (IDE)*) [19].

Težave pri izobraževanju na daljavo se pojavijo tudi v komunikaciji. Študent, ki se izobražuje

od doma in naleti na neko težavo, ki je ne zna sam rešiti, nima dostopa do svojih kolegov ali mentorjev. Do mentorjev lahko dostopa le preko telefonskih klicev ali elektronske pošte. Če pogledamo še s strani mentorjev, imajo ti težavo s spremeljanjem napredka velikega števila študentov.

Naslednji članek, ki so ga sestavili avtorji z *Univerze Strathclyde iz Valike Britanije (USVB)*, se ukvarja z raziskovanjem vpliva nove strategije kognitivnega pristopa k poučevanju programiranja, ki spreminja mentalni model študentom tako, da v njih ustvari konflikt. V ta namen je bilo razvito tudi spletno okolje, ki implementira uporabo nove kognitivne strategije [20].

Kot pravijo avtorji v članku, s hitrim razvojem IKT narašča tudi potreba po sposobnih programerjih in učenje programiranja postaja globalna skrb. V prvem letu pri predmetih programiranja študenti obvladajo naloge programiranja dosti slabše, kot bi to pričakovali. Slaba uspešnost se pozna predvsem pri tem, da se mnogi izpišejo s smeri računalništva, takih je kar od 30 do 50 %. Kot avtorji poudarjajo in povzemajo po drugih študijah, so za to v glavnem krive težave pri reševanju problemskih nalog, ki nastopajo v programiranju. Nekatere druge študija vidijo krivca za neuspeh tudi v napačnem razumevanju ključnih konceptov pri programiranju, ki so lahko posledično krivi za težave pri reševanju problemov. Tradicionalni učni pristop je za učenje programiranja manj zanesljiv, da bi zagotovil pravilnost v razvoju mentalnih modelov o konceptih programiranja. Študije kažejo, da študenti po enem letu predmeta programiranja še vedno nimajo pravih mentalnih modelov o osnovnih programskeih konceptih.

Na univerzi QUTA se pri začetnih predmetih programiranja srečujejo s podobnimi težavami kot na OUHK in USVB.

1. Namestitev in nastavitev okolja za programiranje.
2. Uporaba urejevalnika besedil.
3. Razumevanje programskih vprašanj in uporabe sintakse jezika pri pisanju programske kode.
4. Razumevanje napak prevajalnika.
5. Razhroščevanje.

Ugotavljajo, da je pri danih vajah programiranja pomembno, da ob težavah novinci dobijo čimprajšni odziv mentorja. V velikih razredih se to izkaže za zelo zahtevno. Tudi začetniki, ki uspešno premagajo začetne ovire in se lotijo takojšnjega programiranja, imajo zelo slabo napisano in konstruirano programsko kodo. Pomagati študentom pisati kakovostno programsko kodo je prav tako časovno zelo zahtevno. Težave programiranja se stopnjujejo, ko se za učenje progrimiranja uporabljam OO programski jeziki, saj ti zahtevajo visoko stopnjo abstraktnega razumevanja programskih konceptov. Za izdelavo spletnega portala za učenje programiranja so na QUTA bili pomembni naslednji cilji [7].

- Omogočiti lažji začetek pri učenju programiranja s pogostim odzivom mentorjev na

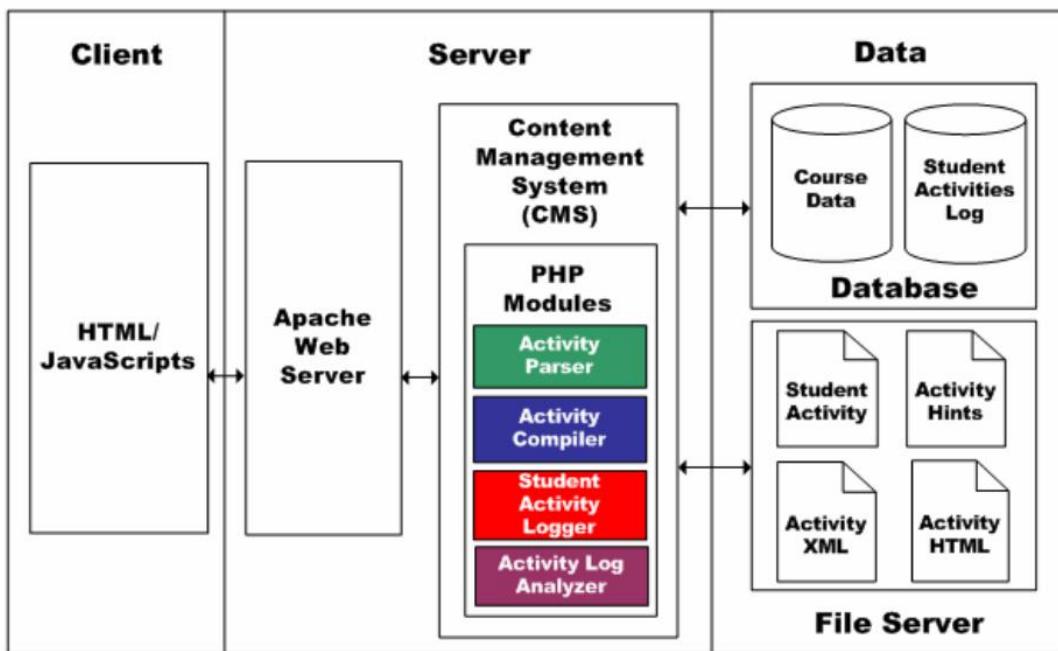
težave novicev. S pomočjo ob pravem času spremenimo odnos novicev do programiranja.

- Izboljšati uspeh začetnih predmetov programiranja.
- Pomoč mentorjem pri učenju in administraciji predmetov programiranja.

5.2 Primeri implementacije in sistemska arhitektura

Zanimalo nas je tudi, kakšna je morebitna sistemska arhitektura takega spletnega portala, zato si pomagamo s primerom arhitekture, ki so ga izdelali na **OUHK**. V nadaljevanju govorimo o *aktivnostih*, ki jih mora študent opraviti, to so naloge, programske rešitve na zastavljene probleme. Študenti na **OUHK** se učijo programiranja v programskem jeziku **Java**.

Kot prikazuje slika 1, je sistem urejanja vsebine (*ang. Content Management System (CMS)*), teče na spletnem strežniku *Apache* z *MySQL* podatkovno bazo. Sistem je narejen iz štirih podmodulov, ki so napisani v skriptnem jeziku *PHP*.



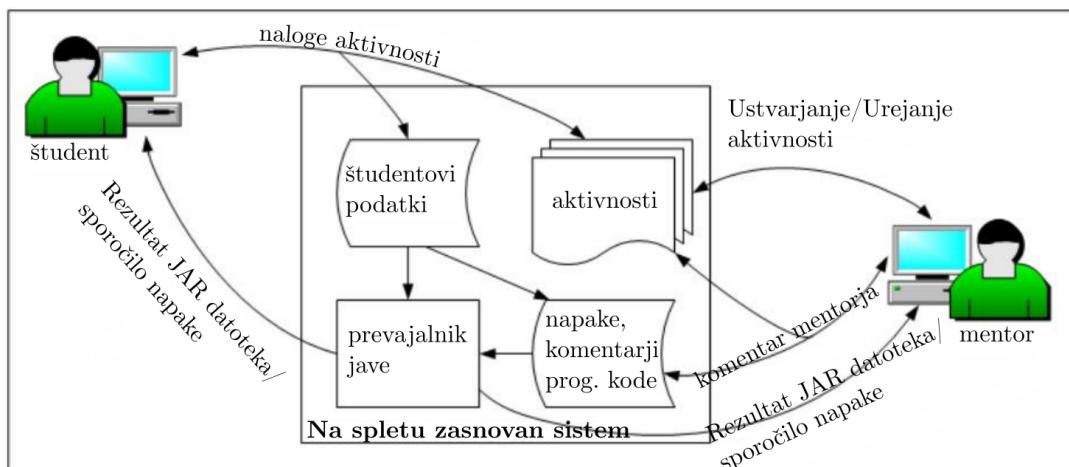
Slika 1: Sistemska arhitektura spletnega portala za učenje programiranja, kot so jo naredili na OUHK [19].

Ti moduli so naslednji: zajem aktivnosti, prevajalnik aktivnosti, dnevnik študentove aktivnosti in analizator dnevnikov aktivnosti. Samo delovanje je naslednje: ko odjemalec pošlje zahtevo za neko aktivnost, se ta naslovi strežniku, ki poišče programsko aktivnost. Z modulom *zajema aktivnosti* strežnik zajame aktivnost, ki je zapisana v obliki **XML**, in naloži vse potrebne datoteke. Zajem aktivnosti prav tako naloži študentovo predhodno delo, ki je shranjeno v datoteki aktivnosti. Ko se vse zajame in naloži, se vsebina pošlje v obliki **HTML** nazaj h klientu.

Strežnik omogoča tudi prevajanje aktivnosti. Ko strežnik dobi prošnjo za prevajanje programske kode, se ta prevede, če v njej ni sintaktičnih napak, in se ustvari datoteka **JAR**, ki jo študent lahko prenese s strežnika. Če so v programu napake, se ustvari dnevnik napake v trenutni aktivnosti, prav tako se napaka izpiše na zaslonu študenta. Vsako aktivnost zajame dnevnik študentove aktivnosti in jo shrani v podatkovno bazo. Z analizatorjem dnevnika študentove aktivnosti mentorji dobijo vpogled v delo študenta in njegovega napredka.

5.3 Pregled delovanja in interakcija s SPUP

Opišimo, kako so si zamislili interakcijo med študentom in mentorjev s spletnim sistemom na **OUHK**. Diagram prikazuje sliko 2. Spletni sistem omogoča študentom in mentorjem spletno okolje za učenje programiranja. Mentorji na spletni portal naložijo snov preko spletnega brskalnika. Mentor lahko naloži datoteke z opisom aktivnosti. Ta datoteka vsebuje osnovne opise in informacije o aktivnostih. Posebej naloži še datoteko, v kateri je predloga za aktivnost. V to predlogo študent rešuje zadano nalogu. V posebno datoteko je naložen tudi namig, ta je študentu v pomoč in ponuja primer izpisa programa.



Slika 2: Prikaz interakcije med študentom in mentorjem s spletnim portalom [19].

Študent lahko pregleduje vse aktivnosti in si naloži katero koli izmed njih. Omogočeno ima, da program prevede na strežniku, ko prevajalnik naleti na napake, strežnik vrne napako na spletno stran. Če študent naleti na težavo, ki je povezana z reševanjem aktivnosti, lahko pošlje prošnjo za pomoč svojemu mentorju. Ko se mentor prijavi v sistem, ima vpogled v napako in na začasno delovno datoteko študenta, mentor lahko zaganja prevajalnik na tem začasnem projektu študenta. Ko mentor popravi programsko napako, odgovori študentu in poda komentar na programsko kodo študenta. Študent ima vpogled v komentarje in predloge, ki jih je posredoval mentor [19].

Na univerzi v US [20] je okrog strategije kognitivnega konflikta nastalo spletno okolje, ki naj bi izboljšalo mentalne modele ključnih programskega konceptov. Učni model je sestavljen iz štirih korakov:

- **Predhodni korak:** mentor razišče, kakšni so predhodni mentalni modeli študentov, in identificira neprimerne.
- **Korak kognitivnega konflikta:** v študentovi predstavi mora sprožiti tak dogodek, ki v študentu izzove neskladje z njegovo predhodno predstavo in s tem se študenta potisne v konfliktno situacijo.
- **Korak konstruiranja modela:** vizualizacija študentu pomaga ustvariti pravo mentalno predstavo.
- **Korak aplikacije:** študent mora rešiti programsko nalogu z na novo ustvarjeno mentalno predstavo.

Spletno učno okolje podpira programski jezik **Java**. Za učenje programskih konceptov je na spletni strani vsak posamezen koncept povezan s potjo, ki predstavlja načrt potovanja. Poti konceptov se povezujejo tako, da se ti nadgrajujejo, saj znanje določenega koncepta potrebuje neko predznanje prejšnjega. Tako za razumevanje določevanja reference najprej potrebujemo predznanje o spremenljivkah ali npr. preden se študenti učijo, kako se podajajo parametri v podprograme, morajo najprej razumeti, kaj je obseg nekega podprograma. Torej je vrstni red spoznavanja programskih konceptov pomemben. Med potmi so gumbi, ki predstavljajo vsak koncept. Na vsakem gumbu je označen rdeč križ, kar pomeni, da študent še ni spoznal koncepta. Ko študent opravi naloge, povezane s posameznim konceptom, se rdeč križ spremeni v zeleno kljukico. Ko študent vstopi v koncept, se izpiše študentova zgodovina z nalogami tega koncepta. Vsaka naloga vsebuje tako vprašanje, ki sproži konfliktno situacijo v mentalnem modelu študenta. Nato študenti dobijo učni material v vidni obliki. Za vizualizacijo uporabljajo orodje **Jeliot**, ki dinamično upodablja izvajanje javanskih programov. Za pravilnost razumevanje mentalnega modela mora študent odgovoriti na dodatna vprašanja. Če študentovi odgovori niso v skladu s podanim mentalnim modelom, dobi študent povratno informacijo o nepravilnem odgovoru. Naslednji korak je ta, da mora študent zagnati vizualizacijo dela programske kode, ki si ga je prej moral predstavljati. Tako ima možnost, da zazna nepravilnost v svojem mišljenju in tako lahko gradi na pravilnem konceptu [20].

V preteklosti je bilo razvitih mnogo orodij, ki so nastala ravno zaradi raziskovanja učenja programiranja, vendar mnoga od teh zahtevajo, da študenti pišejo celotne programe od začetka do konca. Spletni portal v primeru QUTA uči programiranja v programskega jeziku Java in ima naslednje zmožnosti [7].

1. Spletni portal za programiranja, ki omogoča naloge tipa "zapolni prazna mesta".
2. Ogrodje za analizo, ki preverja kakovost in pravilnost, nalog tipa "zapolni prazna mesta".
3. Samodejni sistem za dajanje povratnih informacij, ki sporoča prilagojena sporočila

prevajalnika in formalni odziv študentom in njihovim mentorjem. Poročilo vsebuje kakovost napisanega programa, strukturo in pravilnost glede na programsko analizo.

5.4 Rezultati izvedenih rešitev SPUP

Večina študentov smeri računalništva na OUHK nima predhodnih izkušenj v programiranju s programskim jezikom **Java**. Sistem se uporablja kot spletno okolje za učenje programiranja. Študentom je s tem dana množica aktivnosti oz. nalog, ki jih morajo sami uspešno opraviti. To lahko počnejo kadarkoli in kjerkoli. Študentom ni treba nastavljati programskega okolja, študenti vse programe, ki jih napišejo, lahko takoj prevedejo in jih zaganjajo na svojih računalnikih. Uporaba spletnega portala je pokazala, da so študentje oddali 100 % programskih nalog, napisanih v Javi. Kar kaže na to, da so študentje samozavestno reševali naloge in jih oddajali. Pred uporabo spletnega portala je oddaja nalog bila 80 %.

Kot pravijo avtorji članka in portala [19], je to šele začetek uporabe spletnega portala, ki nudi osnovno funkcionalnost. V nadaljevanju nameravajo dodati še inteligentni sistem, ki bo nadzoroval napredok študentov.

Za izboljšanje mentalnih modelov avtorji predlagajo konstruktivno naravnani učni model, ki vključuje strategijo kognitivnega konflikta in vizualiacijo programov. Zgodnje preizkušanje strategije kognitivnega konflikta pokažejo, da so študenti bolj zavzeti za učni material in jih motivira tako, da si prej ustvarijo pravilno mentalno predstavo [20].

Tudi začetniki, ki uspešno premagajo začetne ovire in se lotijo takojšnjega programiranja, imajo zelo slabo napisano in konstruirano programsko kodo. Pomagati novincem pistati kakovostno programsko kodo je časovno zelo zahtevno opravilo.

Težave programiranja se stopnjujejo, ko se za učenje programiranja uporabljam Objektno orientirani programski jeziki, sej ti zahtevajo visoko stopnjo abstraktnega razumevanja programskih konceptov in so načrtovani predvsem za zahtevne programerje.

Rezultat dela avtorjev spletnega portala QUTA gre še nekoliko naprej od OUHK in v njihov spletni portal vgradijo odziv spletnega portala, ki javlja o pravilnosti programa in o kakovosti. Ogrodje (*ang. framework*) za analizo programske kode vsebuje naslednje komponente [7]:

- sintaktično ali semantično opozarjanje na napake ali napake prevajalnika,
- odziv na kakovost in pravilnost programske kode,
- formalni odziv učitelja oz. komunikacija med učiteljem in učencem.

5.5 Značilnosti SPUP

Ena od osnovnih in glavnih komponent pri vseh SPUP je orodje, ki omogoča pisanje in preizkušanje programske kode. Poimenujemo jo lahko kot **spletna aplikacija za programiranje (SAZP)**. Njene glavne značilnosti so naslednje:

- **urejevalnik besedil**, ki ima lahko osnovne funkcije ali tudi zahtevne, ki so značilne za IRO;
- omogočen je zagon napisanega progama z vhodnimi in izhodnimi podatki;
- omogočena je **povratna informacija**:
 - **sintaktičnih napak**, ki ju vrne prevajalnik ali tolmač;
 - **semantičnih napak**, ki preverjajo želen rezultat napisanega programa oz. pravilno rešitev.

Iz pregleda SPUP, ki so nastali na univerzah, smo se lahko poučili, kaj so nekatere značilnosti spletnih portalov. Strnimo te značilnosti, saj jih bomo pozneje uporabili pri iskanju, kategorizirjanju in vrednotenju. Spletni portal vsebuje naslednje elemente:

- razdelano vsebino z nalogami oz. aktivnostmi,
- **spletno aplikacijo za pisanje programske kode**,
- omogočena je komunikacija med mentorjem in novincem,
- omogočen je pregled nad napredkom novincev oz. tako imenovan *nadzor nad razredom*.

6 Kriteriji za klasifikacijo spletnih portalov

Spletni portali za učenje programiranja imajo različne značilnosti in zmožnosti. V poglavju bomo razdelili posamezne kriterije, s katerimi bomo klasificirali in ovrednotili posamezne spletne portale.

6.1 Vrsta vsebine

Po prvem pregledu in iskanju spletnih portalov lahko ugotovimo, da spletni portali za učenje programiranja ponujajo najrazličnejše vrste vsebin in njihove kombinacije, kot je na primer, **tekstovni vodič in spletna aplikacija za programiranje**. V posebno kategorijo bomo uvrstili tudi spletnе portale, ki ponujajo **spletne igre**, ki učijo programiranje. Različne vrste spletnih portalov, ki jih lahko obravnavamo, so naslednje:

- **tekstovni vodiči**,
- **video vodič**,
- **spletna aplikacija za programiranje**, kot smo jo definirali v poglavju 5.5,
- **spletne igre**,
- **kombinacija** vrst vsebin, ki jih lahko še razdelimo na:
 - **najosnovnejša kombinacija** (*tekstovni vodič + preizkus kode*);
 - **napredna kombinacija** (*različne vrste vodičev + spletna aplikacija za programiranje*), ki tvorijo **vadnice**.

V tej diplomskem delu se ne bomo podrobno ukvarjali s tem, katera izmed vrst vsebin predstavlja boljše zmožnosti za prenos znanja. Vsaka ima svoje prednosti in slabosti, zato bomo za vsako izpostavili le bistvene pozitivne značilnosti in tudi slabosti. Zanimale nas bodo predvsem tiste **kombinirane** vrste vsebin, ki bodo predstavljale čim bolj celovit spletni portal za učenje programiranja, kot smo ga definirali v poglavju 5.5.

6.1.1 Tekstovni vodiči

Spletni vodiči veljajo za starejše metode podajanja znanja na spletu. Za njih je značilno, da uporabnika vodijo **po korakih** do nekega določenega cilja, kako nekaj narediti, ali specifičnega znanja. Besedilo, ki podaja znanje, je opremljeno s **primeri**. [21]

Značilni predstavnik takih vodičev je spletna stran <https://docs.python.org>, na kateri najdemo vso dokumentacijo programskega jezika **Python**. Na strani najdemo tudi vodiča z naslovom *The Python Tutorial* [22].

3.1.1. Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators +, -, * and / work just like in most other languages (for example, Pascal or C); parentheses (()) can be used for grouping. For example:

A screenshot of a computer screen displaying the Python Tutorial. The code shown is:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5 # division always returns a floating point number
1.6
```

Two arrows point from the text "primer" and "razlaga" to the code line "20" and the explanatory note "# division always returns a floating point number" respectively.

Slika 3: Zaslonski posnetek poglavja z vodiča *The Python Tutorial* s primerom [22].

Spletne vodiče pri pouku uporabljamo na podoben način, kot bi vodiča, napisanega na učnem listu z **metodo dela s tekstrom**. Sicer je pomembno, da učenci usvojijo uporabo spletnih vodičev, vendar so spletni vodiči marsikdaj prezahtevni za uporabo, sploh na osnovnošolskem nivoju, z vso tehnično dokumentacijo. Negativna stran spletnih vodičev je še ta, da ne moramo neposredno s spletno strani preizkušati primerov programske kode, kar je slabo tudi z motivacijskega vidika.

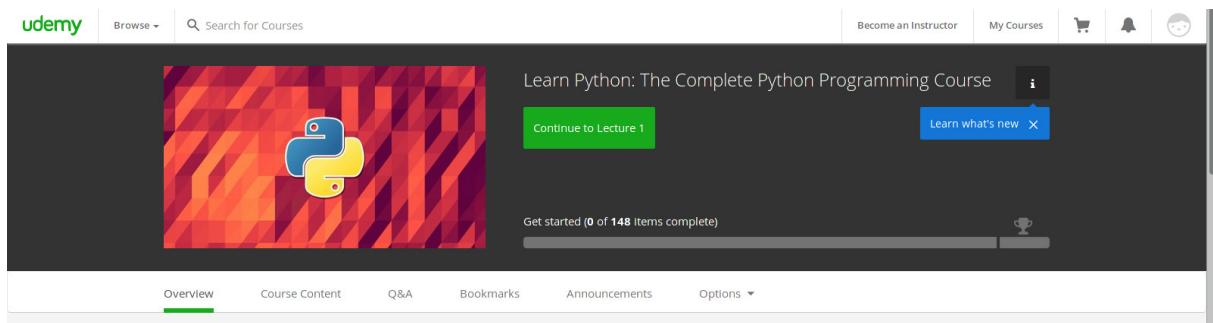
6.1.2 Video vodiči

Z razmahom video vsebin na spletu so marsikateri spletni vodič dopolnili oz. jih zamenjali video vodiči. Popularno je postalo zajemanje oz. **snemanje lastnega namizja**. Video vodiče najdemo za številna področja, od uporabe določene programske opreme in vse do programiranja. Ena izmed prednosti video vodičev je ta, da ti omogočajo nazornejši prikaz nekega postopka po korakih. Preden sami opravimo nek postopek, lahko v video posnetku opazujemo vsak korak, potek miške in poleg tega poslušamo razlago, če je ta vključena. Številne študije kažejo, da je učenje z multimedijo, torej kombinacijo zvoka in slike, učinkovitejše kot samo poslušanje ali branje teksta [23]. V razredu je uporaba video vodičev lahko koristna pri samostojnjem in domačem delu. Uporaba video vodičev ima tudi slabe strani, v njih lahko predstavimo dosti manj vsebine in iskanje po vsebini v video ni preprosto, kot je to pri tekstu.

Eden izmed spletnih portalov, ki je specializiran za podajanje znanja z video vodiči, je *Udemy* [24]. Na njem lahko vsakdo postane učitelj in pripravi učne ure z različnih področij, ne samo z računalniške znanosti. Nekateri sklopi učnih ur so v celoti brezplačni, kljub temu je večina plačljivih.

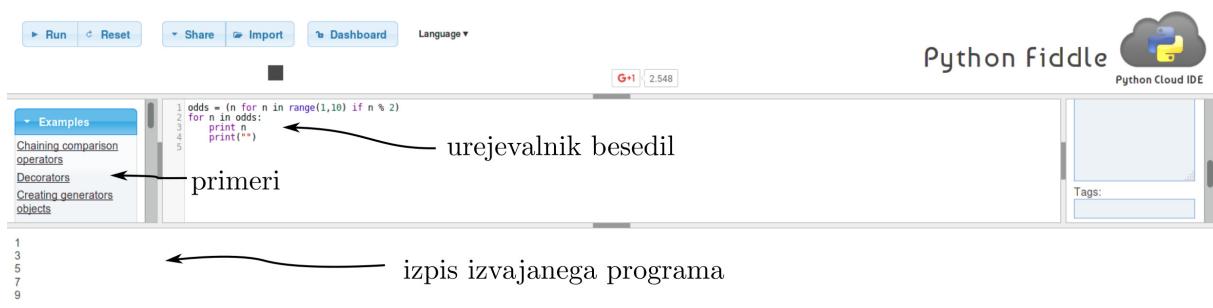
6.1.3 Spletna aplikacija za programiranje

Nekatere spletne strani ponujajo le spletno aplikacijo za programiranje, kot smo jo povzeli v poglavju 5.5. Taki spletni portali ne ponujajo vsebine, ponujajo le aplikacijo, ki jo uporabimo kot **orodje**. Ali pa ponujajo le toliko vsebine, kot je potrebno, da se uporabnik nauči uporabljati spletno aplikacijo. Kljub temu da nas zanimajo celoviti spletni portali, ki ponujajo tudi



Slika 4: Uvodna stran enega izmed video vodičev za učenje Pythona [24].

vsebino, nas bodo podrobneje zanimala tudi orodja, saj so ta ključna za nekatere prednosti, ki jih ponujajo spletni portali za učenje programiranja. Predstavnik takega orodja je *Python Fiddle* [25]. Omogoča osnovni urejevalnik besedila (slika) z barvanjem programske kode, s predlogami za samodokončanja izpisa vgrajenih funkcij. Uvozimo lahko datoteke in jih delimo. Zaganjamо napisane programe, izhodni podatki se izpišejo v konzoli.



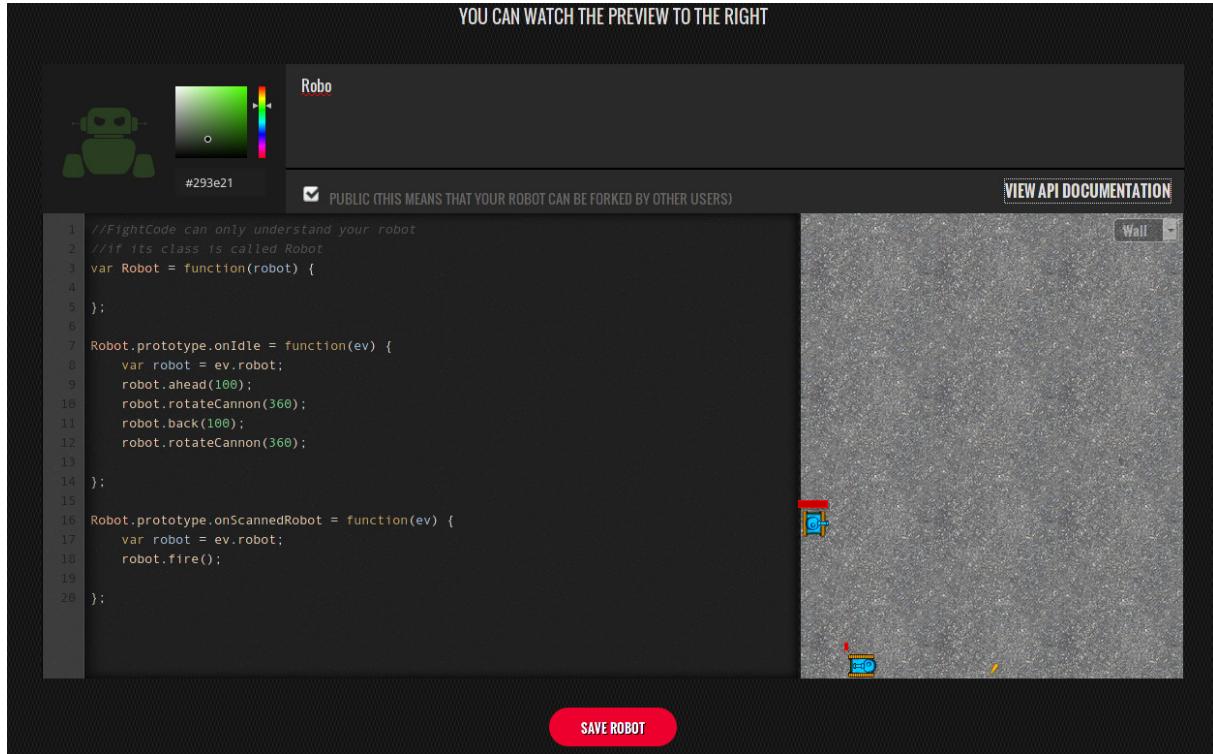
Slika 5: Zaslonska slika spletnne aplikacije za programiranje *Python Fiddle* [25].

Spletne tehnologije so danes zelo napredovale in že nekaj let se aplikacije in podatki selijo v **oblak**. Te aplikacije in podatki so dostopni kjer koli. V **oblaku** se razvijajo številna profesionalna okolja **IRO**, ki omogočajo delo na večjih projektih in ponujajo napredne funkcije IRO, ki smo jih drugače lahko imeli le z namiznimi aplikacijami. Navedimo dva primera: *Codenavy* [26] in *Cloud9*[27]. Oba ponujata profesionalni IRO v oblaku. Za uporabo v šoli sta ti dve okolji preveč zahtevni in jih ni smiselno uporabljati pri poučevanju novincev. S tem jim otežimo učenje programiranja, saj potrebujejo čas, da spoznajo in se naučijo uporabljati IRO.

6.1.4 Spletne igre

Na spletu obstajajo številni spletni portali, ki učijo in spodbujajo k učenju programiranja z igram podobnimi vsebinami. Vsebina je razdeljena na stopnje. Igralci napredujejo iz stopnje v stopnjo in pri tem nabirajo izkušnje, nove veštine in dosežke. Za igranje igre ne upravljamo gibanja lika v igri s tipkovnico in miško, temveč pišemo programsko kodo, ki upravlja njihovo početje. Takšni spletni portali dajejo zelo dobro motivacijsko osnovo, saj se novinci spoznajo

na osnovne principe igranja iger. Primer spletnega portala je *Fightcode* [28] (slika 6). Igralci programirajo robota v programskem jeziku JavaScript. Vsak izmed igralcev lahko izzove drugega igralca v boj med roboti. Za vsako zmago se igralec pomika navzgor po lestvici najboljših robotov.



Slika 6: Zaslonska slika spletnje strani Fightcode [28].

V nadaljevanju si bomo še podrobneje ogledali značilnosti nekaterih drugih spletnih iger, ki učijo programirati.

6.1.5 Kombinirane vrste vsebin

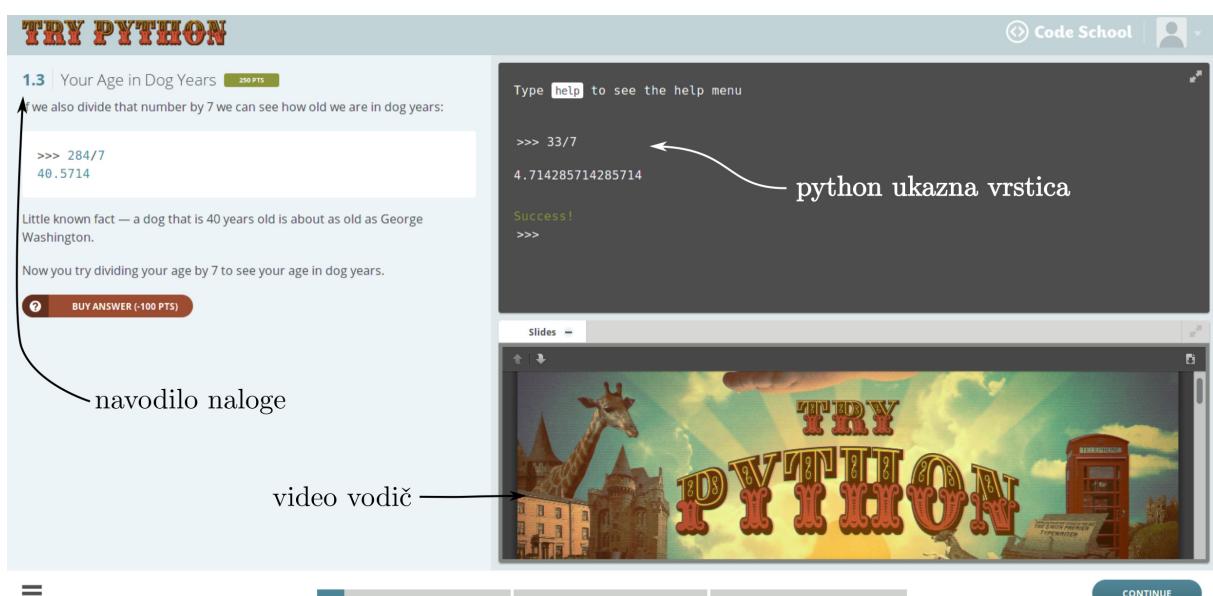
V prejšnjih poglavjih smo opisali **osnovne vrste** spletnih portalov. Zanimale nas bodo predvsem **kombinirane vrste**, ki so sestavljene iz osnovnih. Na spletu najdemo številne kombinacije spletnih portalov. **Osnovno kombinirano vsebino** predstavljajo spletni portali, kot je *w3School* (slika 7) [29]. Sestavljeni so iz **tekstovnih vodičev** in **najosnovnejšega preizkusa programske kode**. Vsak primer v vodiču je opremljen s primerom, ki ga lahko zaženemo in preizkusimo. Za izvajanje primera pritisnemo na gumb **Preizkusi!** (*ang. Try it!*) Programsko kodo primera lahko tudi spremojamo in jo ponovno izvajamo.

Napredno kombinirano vsebino predstavljajo spletni portali, ki so sestavljeni iz **tekstovnega vodiča in/ali video vodičev ter spletnne aplikacije za programiranje**. Omenimo naslednjega predstavnika *Codeschool* [30]. Taki napredni kombinirani vsebini pravimo **vadnica**. V njej je navadno podana razlaga, zastavljen problem oz. naloga, ki ga rešujemo v urejevalniku besedil.



Slika 7: Zaslonska slika spletnne strani *w3School* [29] v poglavju HTML.

V vadnici imamo navadno še preverjanje rešitev, torej sintaktično preverjanje pravilnosti napisane programske kode, ter pravilnost rešitve naloge.



Slika 8: Zaslonska slika spletnne strani *Codeschool* [30].

Različne kombinirane vsebine imajo različne zmožnosti, ki jih bomo spoznali na konkretnem primeru v podrobнем pregledu.

6.2 Jezik spletnega portala

Ugotovimo lahko, da večina spletnih portalov uporablja **angleščino** kot primarni jezik. Nekateri ponujajo tudi druge jezike, vendar je **slovenščina** zaradi majhnosti le malokrat zajeta, razen v redkih primerih. Angleščina je glavni jezik spletja in računalniške znanosti, zato je

pomembno, da učenci oz. dijaki spoznajo tudi angleške izraze in jih povežejo s pravilnimi slovenskimi. Kljub temu da je učenje v slovenskem jeziku predpisano, lahko vsako uro z uporabo spletnih portalov za učenje programiranja medpredmetno povežemo z angleščino. Zanimalo nas bo, kateri jezik uporabljajo spletni portali: **angleščina (da/ne), slovenščina (da/ne), drugi (da/ne)**.

6.3 Ponujena znanja

Spletne strani za učenje programiranja navadno ponujajo znanja oz. veščine programiranja z določenim programskim jezikom. Nekatera svojo ponudbo širijo tako, da ponujajo številne druge projekte, ki združujejo prej naučeno znanje. Na primer izdelava **interaktivne spletne strani**. Za posamezen spletni portal nas bo zanimalo, ali ponuja samo:

- **znanja/veščine programiranja** oz. učenje določenega programskega jezika,
- **znanje algoritmov** ali tudi
- **druga projektna znanja/veščine** (npr. izdelava spletne strani).

6.4 Programski jeziki

Zanimalo nas bo, katere programske jezike ponuja nek spletni portal. Najpogostejše programske jezike smo opisali v poglavju 3.4, v prvi vrsti nas bodo zanimali spletni portali, ki ponujajo najpopularnejše programske jezike in tiste, ki se v izobraževanju uporabljajo pri nas. Prednost bodo imeli tisti, ki ponujajo Python. Večina takšnih spletnih portalov ponuja več programskih jezikov.

6.5 Težavnostna stopnja

Vsak spletni portal je namenjen svojemu občinstvu, zato se razlikujejo tudi po težavnostni stopnji, čeprav govorimo o novincih. Glavna težavnostna razdelitev bo na **osnovo in srednjo šolo**. Po potrebi bomo podrobnejše razdelili že osnovno šolo, ki je razdeljena na triletja. Večina spletnih portalov izhaja iz Združenih držav Amerike, zato smo povzeli njihove stopnje šolanja (3), saj nekatere strani uporabljajo **K-12** formulacijo za definicijo težavnosti oz. prilagoditev učnemu načrtu. V podrobнем pregledu bomo ocenili, kateri težavnostni stopnji ustreza spletni portal.

Tabela 3: Primerjava starosti in stopnje šolanja šolskega sistema v ZDA in Sloveniji [31].

Leta	ZDA K-12 naziv	SI Primerjava
6-10	<i>Elementary school</i>	1. in 2. triletje OŠ
10-14	<i>Middle school</i>	3. triletje OŠ
10-14	<i>High school</i>	Srednja šola

6.6 Upoštevanje učnih načel

Za uspešno delo in uporabo SPUP v razredu je dobro, da vsebine, ki jih najdemo na spletu, sledijo že v samem jedru nekaterim **načelom**, ki jih upoštevamo tudi drugače pri pouku.

Problemski pristop (Da/Ne) , zanima nas ali spletni portali, ki ponujajo vsebino, uporabljam problemsko zasnovane naloge, torej imajo v začetku obravnavanja snovi podano oz. predstavljeno, kateri problem bomo znali na koncu neke vadnice rešiti. Čeprav je vsebina računalniške znanosti in programiranja že po naravi problemsko zasnovana, marsikdaj ni najbolje predstavljeno, za kaj je neka stvar dobra.

Načelo sistematičnosti (Da/Ne) , lahko potrdimo za tiste spletne portale, kjer so posamezni vsebinski sklopi povezani v nekem logičnem zaporedju. Kot smo že ugotovili pri pregledu spletnih portalov na univerzah, je pomembno, da novinci spoznajo nekatere koncepte prej kot druge. Tiste spletne portale, ki bodo imeli neko rdečo nit v povezavi vsebine, bomo potrdili kot take.

Načelo postopnosti (Da/Ne) , pripišemo lahko spletнемu portalu, ki podaja snov v posameznem vsebinskem sklopu tako, da bo razlago in program nadgrajeval postopoma in se težavnost stopnjuje.

6.7 Uporaba ocenjevanja dosežkov, značilnih za igre

V izobraževanju se uveljavlja trend ocenjevanja napredka in dosežkov, ki je tipičen za video igre. To metodo ocenjevanje so poimenovali *ang. Gamification*. Vsako snov ali nalogu, ki je v osnovi toga, popestrimo z načinom ocenjevanja tako, da vsako nalogu predstavimo z različnimi izzivi. Vsaki nalogi oz. izzivu sledijo različne nagrade, ki jih učenci zbirajo in jim pravimo dosežki [32].

Dosežki v video igrach so prisotni že vrsto let. Dosežki se razlikujejo po kompleksnosti, vse od zmagoslavne glasbe ob končani stopnji ali igri pa vse do kompleksnega sistema dosežkov z zbiranjem značk. Značko igralec dobi, ko na primer zbere dovolj predmetov ali razišče določen odstotek ozemlja. Poznamo več načinov nagrajevanja dosežkov. Kot smo že omenili, lahko dosežke predstavimo kot **značke** ali za posamezne izzive pripravimo sistem točkovanja. Z zbranih točk se lahko sestavijo **lestvice ali uvrstitve**. V razredu morajo biti slednje skrbno

načrtovane, da ne pride do prevelikih razlik med učenci, kjer bi se eni lahko počutili nadrejeni in drugi podrejeni.

Zanimalo nas bo, ali spletni portali, ki učijo programiranja, uporabljajo kakršen koli sistem ocenjevanja dosežkov, saj za tiste, ki ga uporabljajo, lahko rečemo, da imajo dodaten motivacijski faktor. Zapisali bomo **uporabo dosežkov (da/ne)** in tipe **značke, lestvice, zbiranje točk za izkušnje, napredovanja ...**

6.8 Dodajanje lastnih vsebin

Nekateri spletni portali omogočajo, da pripravimo lastne vsebine, ki jih potem delimo. Navadno je **spletna aplikacija za programiranje** ali **vadnica** razširjena tako, da omogoča sestavljanje programskega nalog. Večina teh je tako, da pripravimo **spremno besedilo, začetni program ali ogrodje programa, končno različico in pomoč ali namig**. Lahko so dodani tudi **testni vhodni in izhodni podatki**. S podatki, ki smo jih vnesli, imamo avtomatizirano nalogo, ki jo lahko posredujemo oz. delimo z novincem. Zanimalo nas bo, ali kateri spletni portal omogoča to zmožnost, torej **dodajanje lastnih vsebin (da/ne)**.

6.9 Upravljanje razreda

Zmožnost upravljanja razreda je velika prednost in olajša administrativno delo razreda za mentorja. Osnovni način delovanja je naslednji. Mentor ustvari razred ali predmet podobno, kot je to možno pri sistemih spletnih učilnic, kot je *moodle* [33], in v učilnico povabi učence. Učitelj s spletno učilnico **spremlja napredek in dosežke posameznega učenca**. Pri nekaterih portalih je **omogočena komunikacija** med mentorjem in novincem. Spletni portali spremeljanje učencev navadno ponujajo kot plačljivo storitve za šole, kar navadno ni najbolj poceni. Zanimalo nas bo, ali spletni portal ponuja **upravljanje razreda (da/ne)** in ali je ta storitev **plačljiva ali brezplačna**.

6.10 Dostop do gradiv

Veliko vsebin na spletu je brezplačnih in jih pri pouku lahko uporabimo. Mnogo vsebin je tudi plačljivih. Spletni portali, ki imajo plačljive vsebine, uporabljajo navadno model **plačevanja naročnine** za dostop do vsebin. Uporabnik mora na **letni ali mesečni** ravni odšteti različne zneske. Nekateri izmed portalov, kot je *Codeacademy*, imajo plačljive le nekatere zahtevnejše vsebine in storitve. Drugi portali imajo vso vsebino plačljivo. Obstaja tudi vrsta portalov, kot je *Udemy*, kjer je potrebno plačati za posamezen učni sklop ali temo. Plačljivost dostopa do

gradiv lahko razvrstimo na naslednji način, tako da je dostop:

- brezplačen,
- polplačljiv (*nekatere so brezplačne, druge plačljive*),
- popolnoma plačljive vsebine.

7 Pregled spletnih portalov

V prejšnjem poglavju smo nastavili kriterije, po katerih bomo lažje vrednotili spletne portale. Preden se lotimo tega opravila, določimo še omejitve, katere spletne portale bomo dali v ožji izbor in jih pregledali. Te določitve bodo temeljile na uporabi pri pouku v srednji in osnovni šoli. Omejitve za izbor spletnega portala so naslednje:

- spletni portal vsebuje **spletno aplikacijo za programiranje**, ki lahko nastopa tudi samostojno brez vsebine in jo uporabljam kot **orodje**,
- **vrsta vsebine** naj bo sestavljena iz osnovnih vrst oz. naj bo **kombinirana** vrsta vsebine, ki je lahko **osnovna ali napredna** oz. vsebuje **vadnice**,
- spletni portal ima dosegljivo vsebino **brezplačno ali polplačljivo**.

7.1 Code.org

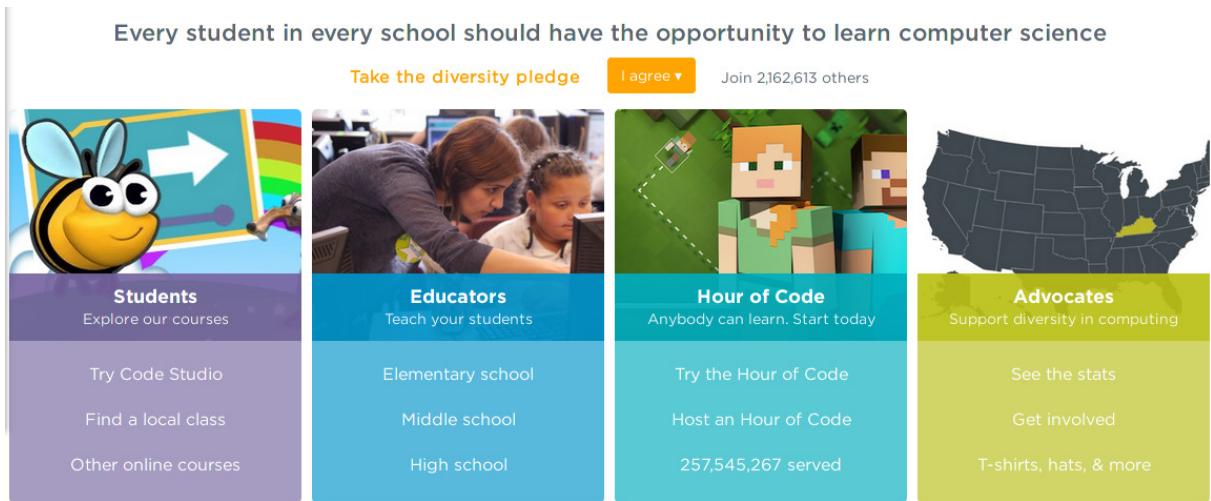
V predstavitvi spletne strani je predstavljena statistika, ki je povzeta iz raziskave v ZDA. Iz nje je razvidno, da bo v prihodnje primanjkovalo kadra za računalniško in informacijsko tehnologijo ter da premalo šol poučuje računalniško znanost [34]. Na *code.org* [35] pravijo, da so neprofitna organizacija, ki se je posvetila širjenju dostopa do poučevanja računalniške znanosti, s poudarkom, ki temelji na nerasni diskriminaciji in povečanju ženskega spola pri učenju računalništva [36].

Spletni portal je v osnovi sestavljen iz štirih glavnih podvsebin (slika 9), ena je namenjena **študentom**, druga **učiteljem** in tretja **Uri kode**. Četrти vsebinski sklop je namenjen **promociji drugih spletnih portalov, aplikacijam in strojni opremi**, ki pripomorejo k učenju računalniške znanosti in programiranja ter so del projekta **Ura kode**. Najprej si bomo ogledali slednjega.

7.1.1 Ura kode (*ang. Hour of code*)

Ura kode so krajši projekti, ki jih lahko organizacije, kot so šole, izvedejo v času ene do dveh ur. Celoten projekt je namenjen promociji računalniške znanosti in programiranju. Vsebine, ki so v okviru tega projekta, so navadno uvodne vsebine.

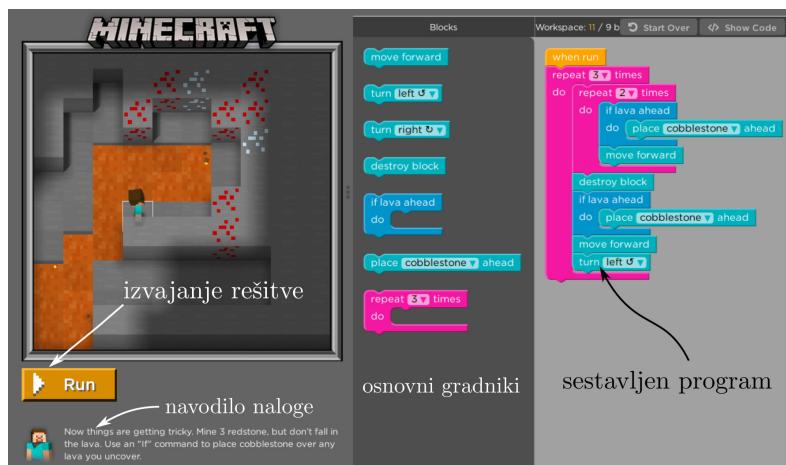
Povezava na portalu **Ura kode** *code.org* nas vodi do zbirke vadnic, ki so del spletnega portala **code**, prav tako najdemo povezave do številnih drugih spletnih portalov in vsebin, ki so vključene v projekt, nekatere bomo opisali tudi v nadaljevanju. Vadnice, ki so del portala, ponujajo številne začetne vsebine, ki so grajene na neki znani temi iz sveta računalniških iger ali animiranih filmov.



Slika 9: Zaslonska slika dela začetne spletnne strani *code.org* [35], iz katerega je razvidna razdeljenost vsebin.

Primer vadnice, ki smo si jo izbrali, je tematsko povzeta iz znane video igre **Minecraft**. V uvodu vsake vadnice najprej sledi video uvod znane osebe, v tem primeru je to glavni razvijalec omenjene igre. V tem uvodnem delu izvemo, zakaj je postal programer, kaj bomo počeli v tej uri ter kaj se bomo naučili. V predstavitvi je predstavljen tudi uporabniški vmesnik vadnice. Uvodni del predstavitve lahko gledamo kot video ali izberemo zapisan povzetek predstavitve. Snov, ki je razložena v vadnici, je **uporaba ukazov, zank in vejitev**. Vadnica je sestavljena iz več enot. Med uvedbo nove snovi spet sledi video predstavitev ali različica v besedilu.

Vadnice so sestavljene z aplikacijo za programiranje **Code studio** (slika 10). Pisanje programske kode v njej je omogočeno z **zlaganjem gradnikov** ali načinom *ang. Blockly*, kjer z metodo **vleci in spusti** sestavljamo oz. zlepljamo programsko kodo. Podoben programski jezik je tudi **Scratch**, ki smo ga opisali v poglavju 7.3.



Slika 10: Spletna aplikacija Code studio, v kateri so zgrajene vadnice [35].

Pod temi zlepiljenimi gradniki se ustvarja programska koda v jeziku **JavaScript**. Posamezne

enote gradijo na znanju sistematično in postopoma z večanjem težavnostne stopnje. V vadnicah so omogočeni samo tisti gradniki, ki jih potrebujemo za rešitev naloge. Večina vsebinskih sklopov je narejena tako, da lahko v zadnji vadnici vsebinskega sklopa prosto uporabljamo vse gradniki, ki smo se jih naučili uporabljati ter nam ni več potrebno izpolniti konkretnega cilja naloge, da jo opravimo.

7.1.2 Code studio

Code studio ni samo spletna aplikacija, ki omogoča vadnice in pisanje programske kode, temveč je podstran, na kateri novinci najdejo zbrane vsebinske sklope (slika 11), ki so razdeljeni po težavnosti, ki se ji spreminja spodnja meja starosti, zgornja ostaja ista pri 18 letih. Posebna značilnost spletne strani je tudi ta, da ponuja računalniške vsebine, pri katerih ne potrebujemo računalnika, tako imenovano **računalništvo brez računalnika** ali **izključene vsebine** (*ang. Unplugged lessons*). Te vsebine so navadno predstavljene s predstavitvenimi videi in gradivi, ki jih lahko natisnemo.



Slika 11: Podstran Code studio, kjer lahko nadaljujemo na različnih vsebinskih sklopih [37].

S klikom na posamezni vsebinski sklop preidemo na stran s povzetkom. Na tej strani sledimo tudi lastnemu napredku. Vsebine so razdeljene na manjše tematske sklope, ki so spet sestavljeni iz posameznih enot oz. vadnic (slika 12). Med posameznimi tematskimi sklopi najdemo tudi **izključene vaje**, ki jih predelujemo brez računalnika.

Reševanje naloge oz. pisanje programa poteka na podoben način, kot smo ga že predstavili pri enotah **Ure kode**, vendar je teh enot več in na vsaki naslednji enoti ponujajo več gradnikov.

Slika 12: Podstran vsebinskega sklopa [37].

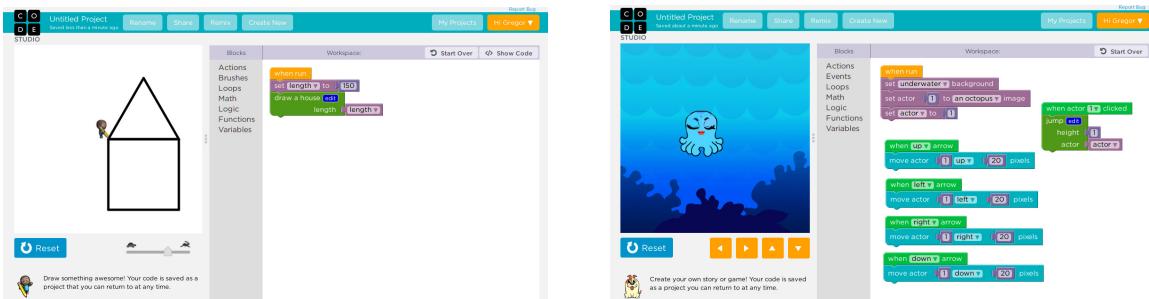
7.1.3 Samostojni projekti

Čeprav nekoliko skrito na dnu podstrani **Code studio**, najdemo gumb, na katerem piše **Moji projekti** (ang. *My projects*). S pritiskom na gumb preidemo na stran, na kateri lahko ustvarimo nove projekte, samodejno shranjene projekte ponovno odpremo ali jih izbrišemo. Ustvarimo lahko naslednje nove projekte, **nariši, ustvari igro in ustvari aplikacije**. Ti projekti so novi in niso omejeni z nobeno vsebino, zato imajo na voljo vse gradnike.

Nariši (slika 13a) deluje na podobnem principu kot programski jezik **Logo**. Imamo pisalo, ki pušča sled, mi mu s programsko kodo določamo potek, barvo itd. Lahko ustvarjamo lastne funkcije in uporabljamo že obstoječe, ki jih lahko spremojamo. Takšna je npr. funkcija **nariši hišo**, ki sprejme argument **velikost**. Vsi gradniki, ki jih lahko uporabimo, so omejeni izključno na uporabo pisala.

Podobno kot nariši uporabljamo **ustvari igro** (slika 13b). Vendar tu ne uporabljamo pisala, temveč lahko gradimo igre in zgodbe. Na delu, kjer teče aplikacija, lahko vstavljam številne prednastavljeni figure, ki se odzivajo na dogodke. Sklopi ukazov pri obeh oblikah projektov, risanje in igre, so naslednji: **akcija, dogodki, zanke, matematika, logika, funkcije in spremenljivke**. Kljub številnim možnostim programske logike so omejitve še vedno prisotne, kot je na primer izbira figur, ki je omejena, saj lastnih figur ne moremo vstavljati. Uporaba teh dveh možnosti je namenjena predvsem osnovni šoli.

Naprednejše možnosti za programiranje smo našli v primeru **ustvarjanja aplikacij** (slika 14). V tej spletni aplikaciji za programiranje se lahko odločamo, ali programsko kodo sestavljamo z gradniki ali jo pišemo tekstovno. Glavna značilnost je ta, da moramo poleg pisanja programske



(a) Aplikacija za risanje.

(b) Aplikacija za ustvarjanje iger.

Slika 13: Samostojni aplikaciji za programiranje, primerni za osnovno šolo [37].

kode ustvariti še **uporabniški vmesnik**, ki je omejene velikosti. Sestavljanje uporabniškega vmesnika je podobno sestavljanju mobilnih aplikacij. S programskim jezikom nismo več tako omejeni in lahko uporabljamo vse vgrajene funkcije JavaScripta. Vgrajena je tudi dodatna možnost ukaznega izpisa in osnovni pomočki za razhroščevanje, **prekini**, **preskoči**, **vstopi** in **izstopi**.

ustvarjanje uprabniškega vmesnika

posamezni gradniki

skupine gradnikov

cestavljanje prog. kode

```

onEvent("button1", "click", function(event) {
    var A = parseInt(getText("text_input1"));
    var B = parseInt(getText("text_input2"));

    if (getText("dropdown1") == "+") {
        //Sestevanje
        setText("text_input3", A+B);
    }
    if (getText("dropdown1") == "-") {
        //odstevanje
        setText("text_input3", A-B);
    }
    if (getText("dropdown1") == "*") {
        //Mnozenje
        setText("text_input3", A*B);
    }
    if (getText("dropdown1") == "/") {
        //Deljenje
        setText("text_input3", A/B);
    }
})

```

Slika 14: Spletna aplikacija za programiranje - ustvarjanje aplikacij. Primer: preprosto računalno [37].

7.1.4 Uporaba za učitelje

Celotna vsebina spletnega portala je zasnovana na ameriškem učnem načrtu. Učitelji na strani najdejo podrobne učne načrte s cilji in idejami za njihovo realizacijo.

7.1.5 Povzetek

Spletni portal velja za glavnega pobudnika **Ure kode** in s tem popularizacije učenja računalniške znanosti in programiranja. Na portalu najdemo ogromno pripravljenih in razdelanih vsebin. Primerna je predvsem za čiste začetnike OŠ in jo lahko uporabljamo pri najmlajših učencih. Za SŠ so pripravljene vsebine morda prelahke in nekoliko otroče, zato jih v ta namen ne bi priporočali, čeprav zgornje omejitve vadnic ni. Vsebinski sklopi so dobro razdelani na podrobne enote in ti postopoma stopnjujejo težavnost. Postopnost je včasih celo pretirana in bi kakšna naloga lahko zajela dve enoti, saj so posamezni koraki nezahtevni oz. prelahki. Problemska zasnova je tako, da naloge, ki jih je treba rešiti, izvajajo in rešujejo junaki iz filmskega oz. sveta video iger. To je pomembno predvsem za motivacijo najmlajših. Nekatere vadnice so prevedene v slovenski jezik, vendar je delež prevedenih zanemarljiv. V splošnem lahko rečemo, da večina spletnega portala ni prevedena. Jezik uporabe spletnega portala pri pouku ne bi smel ovirati, saj so naloge razdelane na majhne dele in so navodila preprosta, tako jih lahko učitelj podaja sproti ustno ali jih ima pripravljene na učnih listih po posameznih korakih. Celotna razporeditev na splettem portalu včasih deluje nestrukturirana in neurejena.

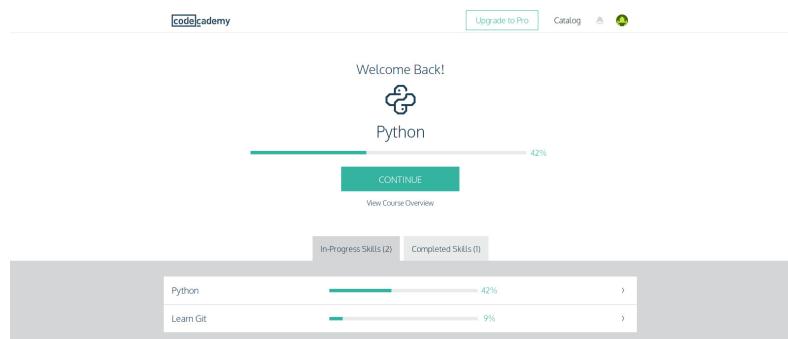
Kot samostojne aplikacije v primeru **risanja in ustvarjanja iger** so uporabne, vendar so po zmožnostih dokaj omejene, zato lahko v ta namen z večjo svobodo in izbiro uporabljamo Scratch. **Ustvarjanje aplikacij** je primerno predvsem za srednje šole, omogoča pisanje programske kode, prav tako lahko posamezne dele kode nazorno predstavimo grafično z gradniki. Uporabna je predvsem v uvodu gradnje aplikacij, saj ustvarjamo uporabniški vmesnik in programsko kodo sami od začetka.

Code | code.org

Vrsta vsebine	Napredna kombinirana vsebina: vadnica (video vodič + navodila (nalog) + spletna aplikacija za programiranje).
Jezik spletne strani	Angleščina: da, slovenščina: da (nekatere vadnice), drugi: ne.
Ponujena znanja	Osnove rač. znanosti in programiranja.
Programski jeziki	Blocky (podobno kot Scratch), JavaScript.
Težavnostna stopnja	Osnovna šola (vadnice, risanje, ustvarjanje iger), srednja šola (ustvarjanje aplikacij).
Upoštevanje načel	Upošteva načelo sistematičnosti: da, postopnosti: da, problemski pristop: da.
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Ne.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačno.

7.2 Codeacademy

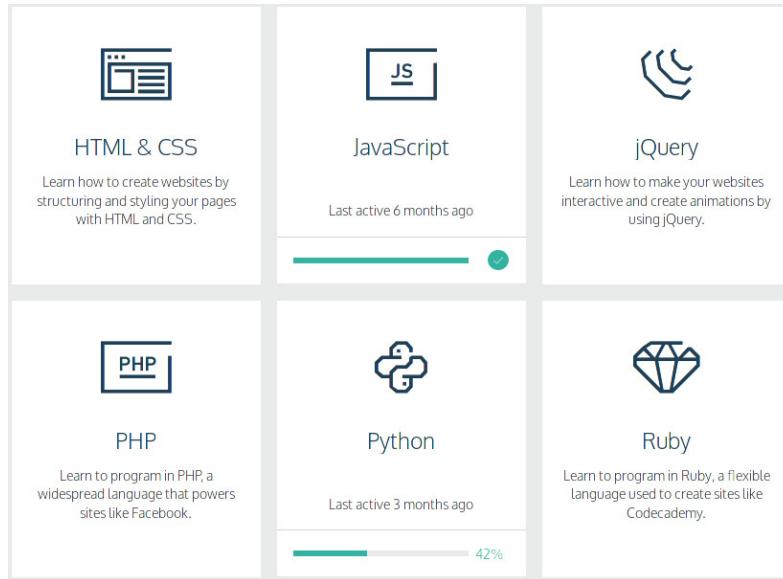
Spletni portal je tipični predstavnik novonastalih portalov za učenje programiranja. Sami o sebi pravijo, da so ameriško podjetje, ki se ukvarja z izobraževanjem. Njihov tim se ukvarja z ustvarjanjem spletne strani *Codeacademy* ter se uči in poučuje, saj želijo ustvariti najboljšo spletno izobraževalno izkušnjo za prihodnost, ki domuje na spletu [38]. Po registraciji in prijavi nas čaka naslednja spletna stran (slika 15). Z začetne, nadzorne strani lahko izbiramo nov vsebinski sklop ali nadaljujemo z že začetimi.



Slika 15: Zaslonska slika spletne strani *Codeacademy* [38]. Začetna, nadzorna stran po prijavi, od tu nadaljujemo na vsebinske sklope, ki smo jih že začeli.

Jezik spletne strani je v angleščini, drugih jezikov ni mogoče izbrati. Če še podrsamo po

spletni strani navzdol, najdemo vsebinske sklope, ki učijo programske jezike in so naslednji: **HTML+ CSS, JavaScript, JQuery, PHP, Python, Ruby** in že v prejšnjem odseku so bile na voljo osnove **Jave**.



Slika 16: Zaslonska slika spletnne strani *Codecademy* [38]. Seznam znanj/veščin programskih jezikov, ki jih ponuja spletni portal.

Nad zbirkom osnovnih programskih jezikov najdemo druga **ponujena znanja**. V tem delu najdemo nekatere vsebine, kot je na primer, učenje **SQL**, uporaba **ukazne vrstice** ali uporaba spletnega orodja za nadzor različice **GIT**. Nekatere vsebine so sestavljene kot projekti, taki sta na primer **naredi spletno stran** ali **naredi spletno stran interaktivno**. Strnemo lahko, da spletni portal ne ponujajo le znanja in veščine **programiranja in programskih jezikov**, temveč tudi druga znanja. S klikom na želeno vsebino pridemo na stran (slika 17), s katere lahko nadaljujemo tam, kjer smo ostali ali pregledujemo posamezne teme, ki smo jih že opravili ali tiste, ki nas še čakajo.

Razvidno je, da so teme sistematicno razporejene, zato lahko ugotovimo, da je **načelo sistematičnosti upoštevano**. S pritiskom na gumb za nadavaljevanje (*ang. Continue*) odpremo urejevalnik (slika 18) na temi in podenoti, na kateri smo ostali.

Vsaka tema vsebinskega sklopa je razdeljena na več enot oz. vadnic, ki so sestavljene tako, da postopoma dograjujejo program. Delo, ki smo ga opravili v predhodni vadnici, se samodejno prenese naprej, ko je to potrebno. Lahko povemo, da je **načelo postopnosti upoštevano**. Pri nekaterih tematskih sklopih sledi najprej ponovitev že naučenega. Na primer pri temi *seznamami in funkcije* najprej sledi pregled osnovnega upravljanja s seznamimi in pisanjem funkcij. Uporabniški vmesnik (slika 18) je urejen tako, da imamo na desni strani podano snov, ki je sestavljena s **primerom določene programske strukture in navodili**, kaj moramo dograditi v programu. Na levi strani imamo **urejevalnik besedil**, v katerega pišemo programsko kodo.

Want more practice and review? Upgrade for the complete experience.

8 Projects 9 Quizzes 1 Final Project

UNIT 1: PYTHON SYNTAX

- Lesson: Python Syntax
- Lesson: Tip Calculator
- Quiz: Python Syntax

UNIT 2: STRINGS AND CONSOLE OUTPUT

Slika 17: Zaslonska slika podstrani spletnne strani *Codeacademy* [38], na kateri lahko pregledujemo posamezne teme in nadaljujemo tam, kjer smo ostali.

For the Record

Excellent. Now you need a hard copy document with all of your students' grades.

```
animal_sounds = [
    "cat": ["meow", "purr"],
    "dog": ["woof", "bark"],
    "fox": [],
]
print(animal_sounds["cat"])
```

The example above is just to remind you how to create a dictionary and then to access the item stored by the "cat" key.

Instructions

for each student in your students list, print out that student's data, as follows:

- print the student's name
- print the student's homework
- print the student's quizzes
- print the student's tests

UREJEVELNIK

```
lloyd = {
    "name": "Lloyd",
    "homework": [90.0, 97.0, 75.0, 92.0],
    "quizzes": [88.0, 40.0, 94.0],
    "tests": [75.0, 90.0]
}
alice = {
    "name": "Alice",
    "homework": [100.0, 92.0, 98.0, 100.0],
    "quizzes": [92.0, 83.0, 91.0],
    "tests": [88.0, 97.0]
}
tyler = {
    "name": "Tyler",
    "homework": [87.0, 75.0, 22.0],
    "quizzes": [0.0, 75.0, 78.0],
    "tests": [100.0, 100.0]
}
students = [lloyd, alice, tyler]
```

Izpis programa.

Programska koda prenešena s prejšnje enote.

Dodan del programa, zahtevan v navodilih enote.

Shrani in oddaj programsko kodo.

Slika 18: Zaslonska slika *Codeacademy* [38] vadnice, ki jo sestavlja urejevalnik z navodili in oknom za izpis v programu.

Urejevalnik zna barvati programsko kodo in samodejno predviditi zamike besedila. V zgornjem desnem kotu je **okno za izpis**, v katerem se izpisujejo **izhodni podatki** iz programa in napake sintakse **Python tolmača**. Spodaj je gumb za **shrani in oddaj programsko kode** (*ang. Save and Submit Code*).

Vsaka v uvodnem delu predstavi novo problematiko, ki jo potem v posameznih enotah rešujemo. Vsebina je predstavljena **problemško**. Samo delo z **vadnico** poteka tako, da napišemo program v **spletno aplikacijo za programiranje**, ki je zahtevan v navodilih. Ko menimo, da imamo pravilno rešitev, pritisnemo na gumb **shrani in oddaj programsko kode**. Program najprej preverja **sintaktično** pravilnost. Napako program vrne nad gumbom za oddajo programske kode, podrobna napaka **Pythonovega tolmača** se izpiše v **oknu za izpis**. Zatem sledi **semantično** preverjanje pravilnosti rešitve naloge. V posamezni enoti program samodejno vrši osnovno preverjanje programa s točno določenim rezultatom. Dokler test napisanega programa ne da pravega rezultata, ne moremo nadaljevati na naslednjo enoto. Ko se nam zatakne, spletna stran ponuja **forum**, na katerem najdemo odgovor ali lahko postavimo vprašanje. Forum je razdeljen na posamezne teme in enote, tako da lahko hitro najdemo zahtevano vprašanje.

Za uspešno premagovanje enot je uporabnik nagrajen z **značkami**, ki so vidne na strani njegovega profila. Na tej strani se beležijo tudi predelane vsebine. Spletni portal torej uporablja nagrajevanje z **dosežki**.

Spletni portal ponuja tudi plačljive vsebine, čeprav je osnova vsebinskih sklopov brezplačna. Za dostop do plačljivih storitev ponujajo model zakupa z naročnino za **19\$/mesec**. Za naročnino uporabnik pridobi dostop do **naslednjih dodatnih storitev**:

- personaliziran učni načrt,
- dostop do kvizov,
- dostop do realnih projektov,
- dostop pomoči v živo preko sporočil.

Ugotovimo, da je spletni portal **polplačljivi** in da dodatne storitve, ki jih ponuja spletni portal, niso potrebne za uporabo pri pouku, saj vse omenjene dodatne storitve lahko zagotovi učitelj.

7.2.1 Uporaba za učitelje

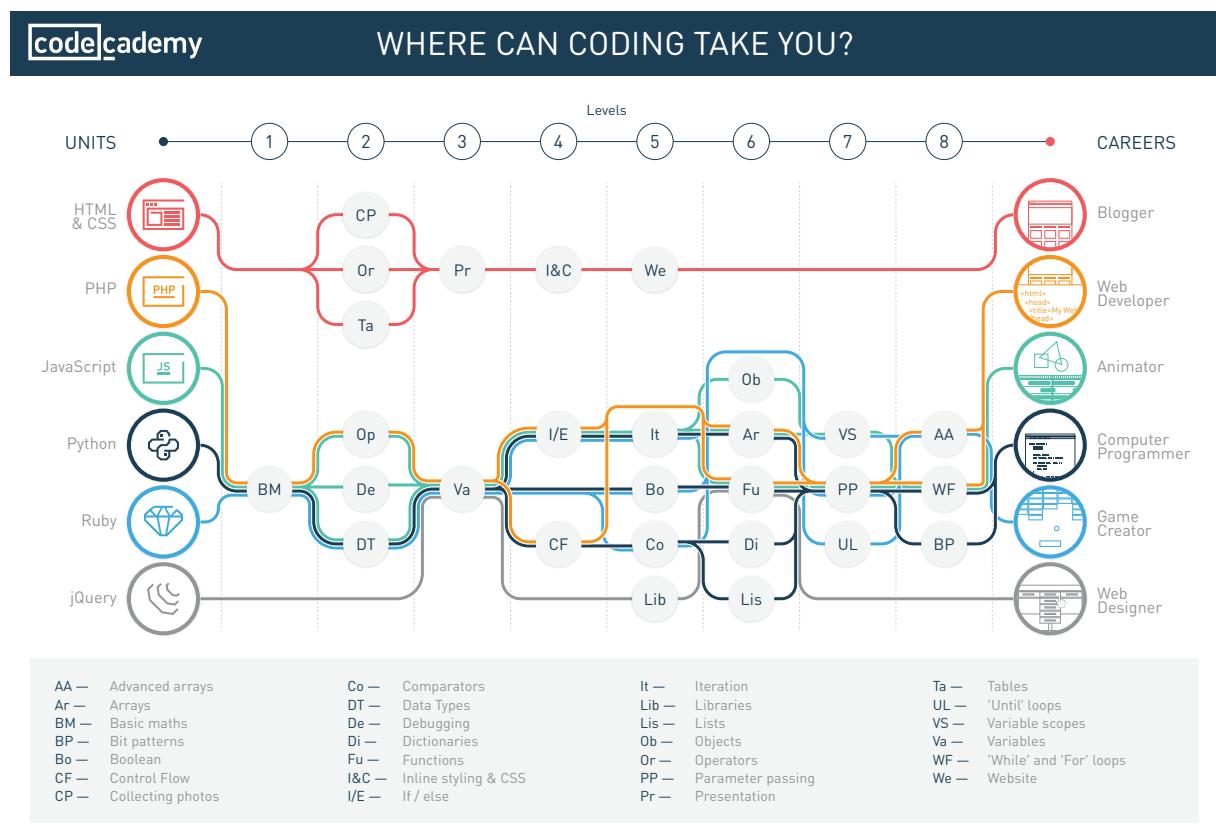
Spletni portal nudi vsebine, ki so prilagojene za šole in uporabo v šolah. V pregledu lahko ugotovimo, da za srednješolske profesorje ponujajo naslednje:

- **trening za učitelje**, ki je možen le v ZDA,
- **gradiva**,

- sledenje napredku učencem,
- ura kode (ang. *Hour of code*).

Podstran z **gradivi** profesorju ponuja razdelane posamezne teme na enote, podobno kot so razdelane na glavni strani. Pod vsako enoto lahko profesor preizkusi, rešuje vaje. Večina enot ima pripravljene kvize, ki jih profesor lahko prav tako preizkusi in se pripravi na učno uro.

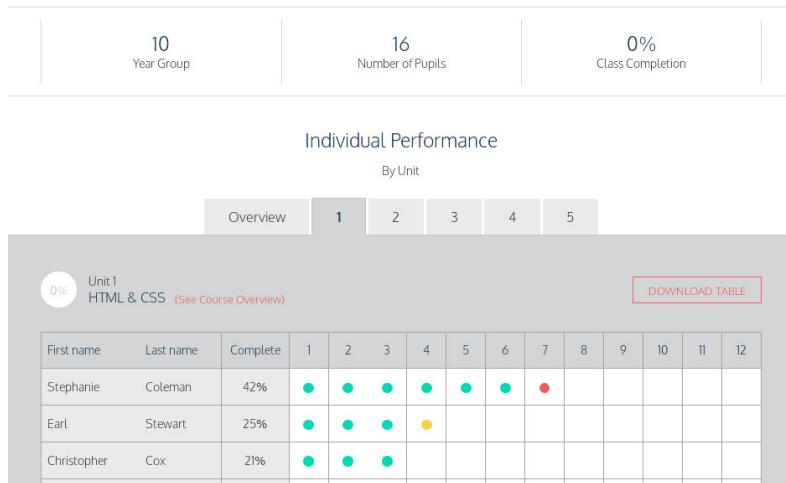
Kot so si zamislili avtorji spletnega portala, so tematski sklopi in posamezne enote med seboj prepletene in vodijo do posameznega cilja. Zemljevid (slika 19) prikazuje povezavo enot na posamezni stopnji in različne cilje. Za vsak tematski sklop lahko učitelj prenese **pregled posamezne teme**, v kateri so podrobnejše zapisani učni cilji in so označene stopnje, ki sovpadajo z zemljevidom.



Slika 19: Plakat s povezavami enot različnih tematskih sklopov po stopnjah, ki vodijo do posameznih ciljev (ang. *Level prograsion mapa*) [38].

Ena izmed naprednih zmožnosti spletnega portala je **sledenje napredku učencem** (ang. *Students tracking*). Ta omogoča, da profesor dijakom ustvari račune in jih povabi na spletnem portalu v razred. Določi tematske sklope, katerim bo sledil. V pregledu (slika 20) lahko profesor opazuje napredek posameznega dijaka po enotah ali v povzetku za celotni napredek. Napredek je prikazan s pikami različnih barv, ki predstavljajo odstotke napredka pri posameznem tematskem sklopu. V tem primeru **ne moremo** govoriti o **upravljanju razreda**, saj je omogočeno le sledenje napredku in ne tudi komunikacija med profesorjem in dijakom, prav

tako ni mogoče dodajati lastnih enot oz. nalog. Pri samem ustvarjanju razreda je profesorju delo zelo olajšano, saj portal omogoča, da informacije o dijakih ustvarjalec razreda kopira neposredno s programa, podobnega kot je **Excel**. V tabeli so podani podatki **ime, priimek, skupina, uporabniško ime**. Dijaki za dostop do spletnega portala uporabijo uporabniško ime, ki si ga izmisli učitelj ali oni sami, geslo je skupno za celotno učilnico. S prejetim uporabniškim imenom in gesлом se dijaki na spletni strani registrirajo s svojim elektronskim naslovom. Če so na portalu že registrirani, profesorju posredujejo uporabniško ime in jih ta doda v ustvarjen razred. Dijak mora povabilo potrditi.



Slika 20: Zaslonska slika *Codeacademy* [38]. Prikazuje tabelo za sledneje napredku učencem.

7.2.2 Povzetek

Codeacademy [38] ima dobro razdelano vsebino, ki je na nekaterih delih dokaj poglobljena. Sistematičnost, postopnost in problemski pristop so prav tako dobro zastavljeni. Portal ponuja številne projektne vsebine in učenje programskih jezikov. Izbor teh je tak, da ustreza današnjim spletnim tehnologijam in zahtevam. Znanje se omejuje predvsem na učenje programiranja ter da se to znanje zna uporabiti v praktične namene. Spletni portal ima nekatere vsebine in zmožnosti plačljive, ampak ima zadostno število brezplačnih vsebin, ki omogočajo normalno učenje.

Zanemarjeno je znanje **računalniške znanosti**, saj se med spoznavanjem programskih jezikov in podatkovnih struktur ne uči različnih algoritmov. Uči se bolj uporabo posameznih funkcij, ki so vgrajene v programske jezike. Slaba stran spletnega portala je ta, da je v celoti v *angleškem jeziku*. Poleg tujega jezika so nekatera navodila napisana dokaj kompleksno in zahteva že dobro poznavanje razumevanja sporočil tolmača in semantičnih napak, ki se zgodijo v programu.

Spletni portal je v primeru učenja spletnih tehnologij, kot je **HTML/CSS**, je primeren za

učence zadnje triade osnovne šole, saj se ti omenjeno snov učijo pri izbirnem predmetu **računalniška omrežja**. V primeru učenja programskega jezika **Python** spletni portal ponuja zahtevna znanja in je primeren predvsem za srednje in višje šole.

Da bi mentor lahko spletni portal uporabljal pri pouku, bi moral imeti prevode navodil za posamezno temo in enoto. Vsekakor ga je možno izvesti kot *praktično vodeno delo*. Spletni portal se lahko uporablja za domače delo in smo uporabo kot takega opisali v poglavju 8.2. Mentor lahko spletni portal priporoča v uporabo kot neobvezno dopolnilno dejavnost tistim dijakom, ki želijo razširiti znanje programiranja. Opozorimo, da jim lahko priporoča le brezplačne vsebine. Čeprav je možno, pa vseeno morda ni primerno uporabljati spletni portal tako, da bi z njim predelali celoten vsebinski sklop in bi ga pri pouku uporabljali kot edino orodje, saj je poučevanje računalniške znanosti na njem omejeno. Lahko pa ga s pridom uporabimo kot dodatek pri učenju programskega jezika, kot je na primer **Python**, ki v nadaljevanju služi kot orodje za poučevanje računalniške znanosti. Pri pouku smo omejeni tudi s spreminjanjem vsebine, ki je ni mogoče prilagoditi učnemu načrtu tako, kot bi to žeeli, zato moramo učno pripravo prilagoditi uporabi spletnega portala. Poleg prednosti in slabosti lahko zaključimo, da ima spletni portal še dodatno motivacijsko vrednost, saj njegova sistematičnost in postopnost ter nagrajevanje z dosežki motivira uporabnike, da imajo željo po dokončanju vsebinskega sklopa, ki ga obravnavajo.

Codeacademy | www.codeacademy.com

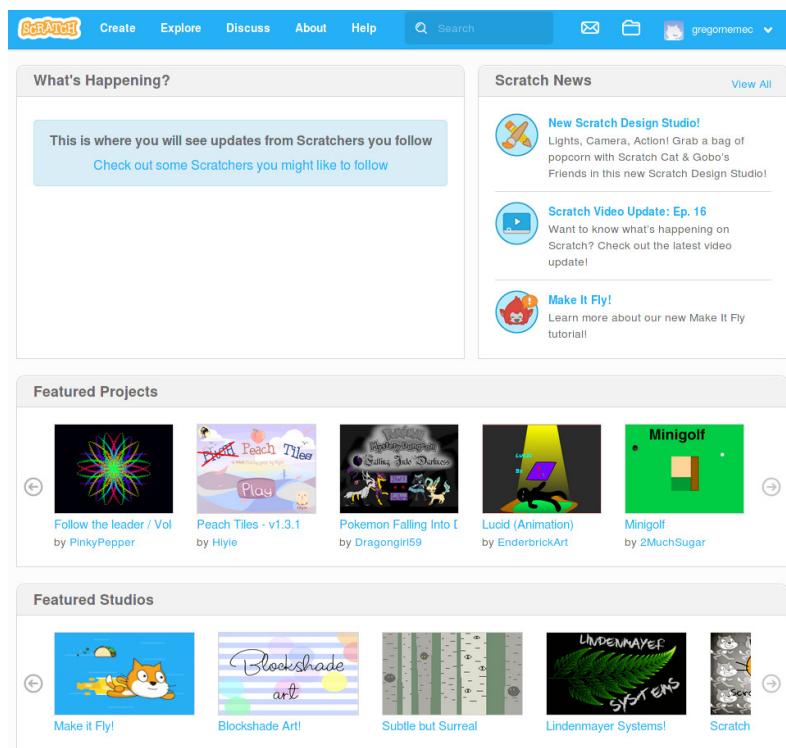
Vrsta vsebine	Napredna kombinirana vsebina: vadnica (primer + navodilo (vodič) + spletna aplikacija za programiranje).
Jezik spletnne strani	Angleščina: da, slovenščina: ne, drugi: ne.
Ponujena znanja	Znanje prog. jezikov, druge vsebine.
Programski jeziki	HTML+ CSS, Java JavaScript, Jquery, PHP, Python, Ruby.
Težavnostna stopnja	3/3 osnovna šola, srednja šola.
Upoštevanje načel	Upoštevanje načela sistematičnosti: da, postopnosti: da, problemski pristop: da.
Dosežki/Gamification	Da (značke).
Dodajanje lastnih vsebin	Ne.
Upravljanje razreda	Da, ustvarjanje razreda in sledenje napredku učencem.
Dostop vsebin	Polplačljiv (plačljivi so projekti, kvizi, podpora v živo).

7.3 Scratch

Spletni portal *Scratch* [39] je spletna različica zelo popularnega programskega jezika **Scratch**. Scratch je pri nas popularen predvsem v **osnovnih šolah** in je zamenjal dolgo uporabljen **Logo**.

Razvoj samostojne namizne različice Scratcha se je končal pri različici 1.4, od tu naprej je razvoj Scratcha potekal za spletno različico. Spletna različica je narejena na osnovi zaprtega **Adobe Flasha**. Za poganganje Scratcha v spletnem brskalniku potrebujemo vtičnik **Flash**. Ko prvič naložimo spletni portal *Scratch* (slika 21), lahko ugotovimo, da ponuja naslednje funkcionalnosti:

- ustvarjanje programov (orodje Scratch),
- deljenje ustvarjenih programov,
- raziskovanje narejenih programov, drugih uporabnikov,
- forum za diskusijo,
- pomoč pri uporabi.



Slika 21: Zaslonski posnetek glavne strani *Scratch* [39].

Scratch smo po vrsti vsebine umestili med **spletne aplikacije za programiranje** oz. smo ga predstavili kot samostojno **orodje**. Lastne vsebine, ki bi v širšem smislu poučevala računalniško znanje, ne najdemo. Vse vsebine, ki so na strani, so namenjene učenju uporabe orodij in spoznavanju zmožnosti programskega jezika. Vseeno lahko govorimo o spletnem portalu, saj

ima ta vse za uspešno uporabo orodja in omogoča vso funkcionalnost, ki jo potrebuje neka spletna skupnost.

7.3.1 Uporaba Scratcha

Scratch omogoča ustvarjanje animacij, predstavitev in iger. Namenjen je učencem, starim od 8 do 16 let, vendar ne predstavlja nobene omejitve na zgornji meji starosti. Preveden je v številne jezike, med njimi je tudi **slovenščina** [40].

Če smo v preteklosti že uporabljali namizno različico Scratcha, nam uporaba spletne različice (slika 22) ne bo predstavljal nobenih težav, saj je postavitev uporabniškega vmesnika zelo podobna, kot je bilo to v namizni različici. Osnovni princip delovanja je tak, da na *oder* (slika 22) postavljamo različne *like*. Vsak lik, ki ga dodamo iz knjižnice ali ga naložimo sami, je predstavljen kot svoj objekt in vsakemu posebej dodajamo programsko kodo, ki jo sestavljamo iz različnih *gradnikov*. V samem orodju lahko dorisujemo k že obstoječim likom, rišemo nove ali jih naložimo neposredno iz računalnika. Dodajamo lahko tudi zvok, ki ga posnamemo sami ali ga izberemo iz knjižnice zvokov. Programsko kodo lepimo skupaj oz. sestavljamo podobno, kot bi sestavliali kocke. Kot smo že spoznali, takemu načinu sestavljanja programske kode pravimo tudi *“Blocky”*. Gradniki so oblikovani tako, da se sklopijo samo tisti, ki se med sabo lahko povežejo.



Slika 22: Zaslonska slika Scratch [39].

Vodiči v Scratchu najdemo na desnem robu (slika 22). Vodiči so sestavljeni tako, da uporab-

nika postopoma vodijo skozi gradnjo programa. S tem je zagotovljena **postopnost**. Posamezen korak v vodiču je sestavljen iz besedila in animiranega poteka dela.

7.3.2 Deljenje in raziskovanje projektov

Vsak uporabnik, ki se registrira na spletnem portalu, ima dostop do svojega profila. Na svoji strani se samodejno shranjujejo projekti, ki smo jih izdelovali in jih od tu lahko ponovno naložimo za urejanje. Vse projekte lahko delimo z drugimi. Vsak projekt ima svojo podstran, na kateri določamo nastavitev za *deljenje* (slika 23). Na strani lahko dodamo *navodila za program* in *zapiske in zasluge*, prav tako določamo, če želimo projekt deliti ali ne.



Slika 23: Podstran za deljenje projekta na *Scratchu* [39].

Če je omogočeno, da lahko delimo vsebine, je na spletnem portalu tudi dobro poskrbljeno za **raziskovanje projektov** drugih uporabnikov. Pod zavihkom *razišči* (ang. *Explor*) najdemo številne projekte, ki so razporejeni v kategorije. Tu lahko črpamo številne ideje in si najljubše projekte shranimo tudi na svojem profilu.

7.3.3 Povzetek

Učni načrt v slovenski osnovni šoli neobveznega izbirnega predmeta **računalništvo** je prilagojen prav za uporabo Scratcha, kot je razvidno iz ciljev. Čeprav z dobro pripravljenimi vodiči spoznavamo tudi na primer veitve, si mora učitelj pripraviti in prilagoditi vsebino za uresničevanja učnega načrta sam. Kot smo že lahko ugotovili, na spletu in na sploh ne zmanjka literature in idej, ki učitelju pomagajo pri uresničevanju učnega načrta s Scratchem. Tudi na spletni strani Scratcha lahko črpamo številne ideje iz projektov drugih uporabnikov.

Za slabost spletnih različic štejemo, da je narejen z zaprto tehnologijo **Adobe Flash**, kar pomeni, da si mora uporabnik naložiti vtičnik Flash na spletni brskalnik. Vtičnik Flash uradno podpira le nekatere komercialne operacijske sisteme in njegovo vlogo zamenjuje vedno boljše zmožnosti samih spletnih brskalnikov s tehnologijo **HTML5 + CSS + JS**.

Scratch | <https://scratch.mit.edu>

Vrsta vsebine	Spletna aplikacija za prog.
Jezik spletne strani	Scratch: angleščina: da, slovenščina: da, drugi: da. Spletni portal: angleščina: da, drugi: ne.
Ponujena znanja	Vodiči za spoznavanje uporabe Scratch in pomoč.
Programski jeziki	Scratch.
Težavnostna stopnja	Osnovna šola.
Upoštevanje načel	Problemski pristop: ne, sistematičnost: ne, postopnost: da (vodič).
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da, vendar v smislu ustvarjanja lastnih projektov in ne celovitih vadnic.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačen.

7.4 Repl.it

Repl.it [41] je še ena **spletna aplikacija za programiranje**. Podjetje, ki spletno stran ustvarja, je tržno osredotočeno na ponujanje **aplikacijskega programskega vmesnika - APV** ali (*ang. application programming interface - API*). Njihov APV uporabljajo številni spletni portali, kot je na primer <https://www.freecodecamp.com> [42] in nekateri drugi plačljivi spletni portali. Na svoji strani ponujajo prost dostop do spletne aplikacije za programiranje (slika 24).

```

repl.it
share save run
Python 3.5.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
[4, 59, 6, 90, 41, 2, 79, 90, 69, 32]
Največje število v seznamu je: 90
>> None

```

python ukazna vrstica

urejevalnik besedil

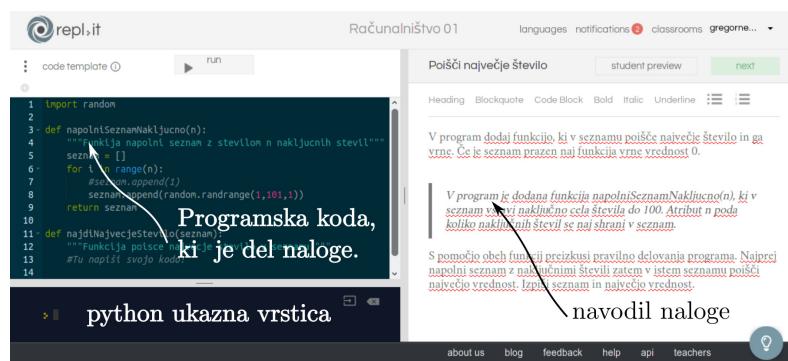
Slika 24: Zaslonska slika spletne aplikacije za programiranje *repl.it* [41].

Aplikacija ponuja številne programske jezike, kot je **Python3**, **Ruby**, **JavaScript**, **HTML**, **CSS**, **C#**, **JAVA** in še mnoge druge. Vsaka nova pojava spletnje aplikacije predstavlja novo sejo. Po registraciji na spletnem portalu lahko shranjujemo posamezne seje. Posamezno sejo lahko **tudi delimo** preko spletnne povezave. **Urejevalnik besedil** omogoča barvanje rezerviranih besed programske kode in ima napredno **možnost ponujanja predlogov** za samodokončanje izpisa rezerviranih besed in funkcij programskega jezika.

7.4.1 Ustvarjanje razredov in nalog

Spletni portal skoraj da ni vreden omembe z **vsebinskega vidika**, tudi kot spletna aplikacija ne predstavlja nekih posebnih funkcij, ki jih ne bi imele tudi nekatere druge spletnje aplikacije. Ima pa spletni portal veliko zmožnost, saj omogoča **ustvarjanje razredov**. Mentor lahko ustvari razred in v njega povabi dijake ter samodejno dodaja oz. **ustvarja lastne naloge** oz. **vadnice** (slika 25). Dijake v razred povabi z dodajanjem elektronskih naslovov v seznam. Dijakom ni potrebna registracija. S povezavo, ki so jo dobili na elektronski naslov, se prijavijo v razred. Mentor v načinu ustvarjanja naloge doda lastna navodila in začetno programsko kodo. V naslednjem koraku se mentor lahko odloči, ali bo pravilnost naloge preverjal sam ali bo dodal samodejni preizkus programske kode. V samodejnem načinu mora podati primer vhodnih podatkov in rezultat izhoda.

V razredu imajo dijaki vpogled v seznam nalog. Naloge so dijakom prikazane z navodili. Dijaki rešujejo nalogo, in ko so zadovoljni s svojo rešitvijo, nalogo oddajo. Mentorju se status naloge pri dijaku spremeni na *oddano* in sedaj lahko mentor nalogo pregleda, poda komentar in jo označi kot *opravljeno* ali jih s sporočilom tudi *zavrne*.



Slika 25: Zaslonska slika pogleda mentorja v načinu priprave naloge [41].

7.4.2 Povzetek

Spletna aplikacija za programiranje ponuja mentorjem računalniških vsebin osnovno orodje za ustvarjanje razredov in nalog ter komunikacijo z dijaki. Pri tem uporablja osnovne zmožnosti

brez pretirane kompleksnosti. Sam spletni portal ne ponuja nobene vsebine, kar je svojevrstna prednost, saj lahko mentor prilagodi naloge učnemu načrtu v svojem jeziku.

Repl.it | <https://repl.it/>

Vrsta vsebine	Spletna aplikacija za prog.
Jezik spletne strani	Angleščina: da, slovenščina: ne, drugi: ne.
Ponujena znanja	Ne ponuja nobene vsebine.
Programski jeziki	Python3, Ruby, JavaScript, HTML, CSS, C#, JAVA in drugi.
Težavnostna stopnja	Srednja šola.
Upoštevanje načel	Problemski pristop: ne, sistematičnost: ne, postopnost: ne (vodič).
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da. Ustvarjanje razreda in nalog ter komunikacija z dijaki.
Upravljanje razreda	Da.
Dostop vsebin	Brezplačen.

7.5 Tutorials point

Tutorials point [43] je eden izmed velikih portalov, ki ponujajo obsežne in zahtevne vodiče tehničnih in netehničnih vsebin. Na spletu je vodičev veliko, tega smo izpostavili, ker ponuja ogromno vsebin **programiranja in računalniške znanosti**, vsi primeri v vodičih imajo **možnost preizkusa** in imajo razvito lastno **spletno aplikacijo za programiranje**, ki ima veliko zmožnosti, nekatere značilne za namizne **IDE**.

Spletni portal ponuja knjižnico vodičev (26) različnih vsebinskih sklopov. S slike je razvidno, da je zares obsežna.

Vsebinsko smo si pregledali vodič za **Python3** (slika 27). Vodiči so oblikovani tako, da na desni strani najdemo kazalo vsebine, v sredinskem delu je razložena snov s primeri.

Nekatere od primerov lahko tudi preizkusimo, s klikom na gumb **preizkusi (ang. Try it)** se nam na isti strani odpre podokno (slika 28). Kodo v urejevalniku lahko spremojamo in ponovno zaženemo.

Tutorials Library			
Free Online Tutorials & Courses			
JAVA TECHNOLOGIES	PROGRAMMING	WEB DEVELOPMENT	SCRIPTS
<ul style="list-style-type: none"> • Learn Apache Ant • Learn Apache POI (Powerpoint) • Learn Apache POI (Word) • Learn Apache POI • Learn AWT • Learn Design Patterns • Learn EasyMock • Learn Eclipse • Learn EJB • Learn Guava • Learn Hibernate • Learn iBATIS • Learn Jackson • Learn JasperReports • Learn Java XML • Learn Java • Learn JBossM5 	<ul style="list-style-type: none"> • Learn Apex • Learn Assembly • Learn Awk • Learn COBOL • Learn C++ • Learn C • Learn Computer Programming • Learn C by Examples • Learn C# • Learn Clojure • Data Structure & Algorithms • Learn D • Learn Erlang • Learn Euphoria • Learn F# • Learn Fortran 	<ul style="list-style-type: none"> • Learn Ajax • Learn AngularJS • Learn Angular Material • Learn ASP.Net • Learn Aurelia • Learn BackboneJS • Learn Bootstrap • Learn CSS • Learn Codeigniter • Learn CoffeeScript • Learn CPanel • Learn Drupal • Learn Django • Learn EmberJS • Learn ExtJS • Learn Flask • Learn Fortran 	<ul style="list-style-type: none"> • Learn JavaScript • Learn jQuery • Learn jQueryUI • Learn Lua • Learn Perl • Learn PHP • Learn PHP-7 • Learn Python • Learn Python-3 • Learn PyQt • Learn WxPython • Learn Ruby • Learn RSpec • Learn Sed • Learn Tcl/Tk • Learn Unix • Learn VBScript

Slika 26: Del seznama oz. knjižnica vodičev, ki ga ponuja spletna stran *Tutorials point* [43].

The screenshot shows a section titled "Single Statement Suites" from a Python 3 tutorial. It includes a code snippet and its execution result. The code is:

```
#!/usr/bin/python3
var = 100
if ( var == 100 ) : print ("Value of expression is 100")
print ("Good bye!")
```

The execution result shows the output:

```
Value of expression is 100
Good bye!
```

Navigation links at the bottom include "Previous Page", "Print", "PDF", and "Next Page".

Slika 27: Zaslonski izrez vodiča za Python3. S slike je razvidno kazalo in gumb za preizkus [43].

The screenshot shows a "urejevalnik besedila" (code editor) with the following Python code:

```
#!/usr/bin/python3
var = 100
if ( var == 100 ) : print ("Value of expression is 100")
print ("Good bye!")
```

Below the editor is a "Result" panel showing the output of the executed program:

```
Executing the program....
$python3 main.py
Value of expression is 100
Good bye!
```

Navigation links at the bottom include "Previous Page", "Print", "PDF", and "Next Page".

Slika 28: Podokno za preizkus primera programske kode [43].

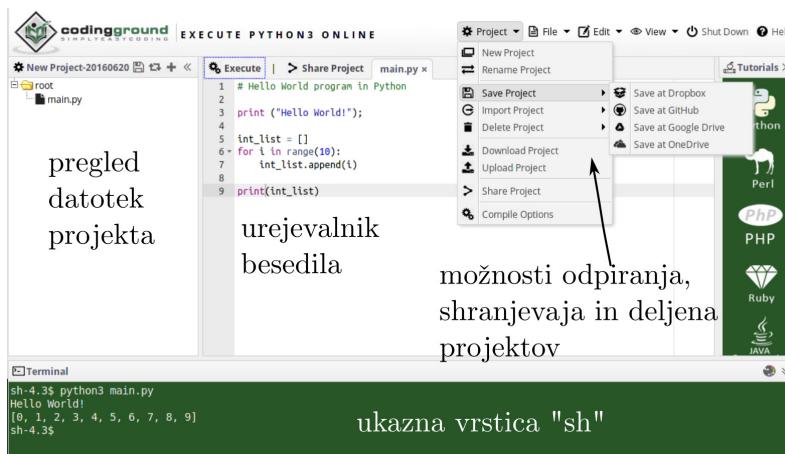
7.5.1 Coding ground

Kot smo že omenili, spletni portal kot orodje ponuja lastno spletno aplikacijo za programiranje. Spletna aplikacija se imenuje *Codingground* [44]. Na uvodni strani orodja (slika 29) smo odkrili številnost ponujenih **programskih jezikov**, ki sovpadajo s številnimi vodiči, ki jih ponuja spletni portal.

 Java 8	 Java MySQL	 JS javascript	 JSP JSP	 jQuery	 Julia
 Ksh Shell	 LATEX Latex	 Lisp	 LOLCODE	 Lua	 Matlab/Octave
 Malbolge	 Markdown	 MathML	 Mozart:OZ	 Nimrod	 NodeJS
 Objective-C	 OCaml	 Pascal	 PARi/GP	 Pawn	 Perl
 Perl MySQL	 PHP	 PHP MySQL	 Web View	 Pike	 Processing.js
 P5.js	 Prolog	 Python	 Python-3	 Python MySQL	 Rexx

Slika 29: Del seznama različnih programskih jezikov, ki jih lahko uporabljamo s spletno aplikacijo za programiranje *Codingground* [44].

Razporeditev spletnne aplikacije (slika 30) je tako, da imamo na levem robu seznam datotek v korenskem imeniku, v sredinskem delu je urejevalnik besedil, nad urejevalnikom najdemo menijsko vrstico in na dnu strani je **ukazna vrstica**, v kateri lahko zaganjamo napisano programsko kodo. V njej se izpisujejo tudi povratne informacije tolmača in izhod programske kode. **Urejevalnik besedil** omogoča barvanje kode rezerviranih besed in nastavljanje barvne sheme urejevalnika. Urejevalnik omogoča še samodejno zamikanje programske kode, ko je to potrebno. **Shranjevanje in uvažanje projektov v oblak** lahko smatramo kot eno izmed večjih zmožnosti te spletnne aplikacije. *Codingground* lahko nastavimo, da se poveže z oblačnimi shrambami, kot so **Dropbox**, **Google Drive**, **Onedrive**, in s sistemom za objavljanje, upravljanje različic in kolaboracijo **Git**. Seveda lahko projekt naložimo neposredno z računalnika in ga tja tudi shranimo. Prednost oblačnega shranjevanja je ta, da lahko programiramo od kjer koli in s katerimkoli orodjem želimo - spletno ali namizno aplikacijo. Spletna aplikacija omogoča tudi upravljanje z datotekami. Lahko ustvarimo, preimenujemo in brišemo datoteke ali imenike. To lahko počnemo iz **menija (file)** ali **ukazne vrstice**. Vsak projekt lahko delimo preko neposredne kratke **url povezave**, kot smo to že videli pri drugih spletnih portalih.



Slika 30: Spletna aplikacija za programiranje - *Codingground* [44].

7.5.2 Povzetek

Vsebina vodičev je zelo tehnična in deluje kot okrnjen povzetek uradne reference za določen programski jezik. Kot taka je predvsem primerna za programerje začetnike, ki se želijo poučiti o določenem programskem jeziku, vendar že poznajo osnovne koncepte programiranja. Velik plus je preizkus programske kode. Vodiče lahko priporočimo kot skrajšano različico reference programskemu jeziku.

S pravo nastavitevjo spletna aplikacija omogoča, da imajo dijaki programsko kodo in snov, ki jo v nekem trenutku predelujejo, povsod na voljo. S pomočjo shranjevanja in deljenja projektov lahko mentor uporabi spletno aplikacijo kot glavno orodje za učenje računalništva in programskega jezika. Mentor mora pripraviti sistem za izmenjavo navodil, programske kode in rešitev dijakov. To lahko stori z uporabo kratkih url povezav. Urejevalnik besedil bi lahko ponujal kakšno zmožnost več, kot jo, vendar zadosti osnovnim potrebam pisanja programske kode.

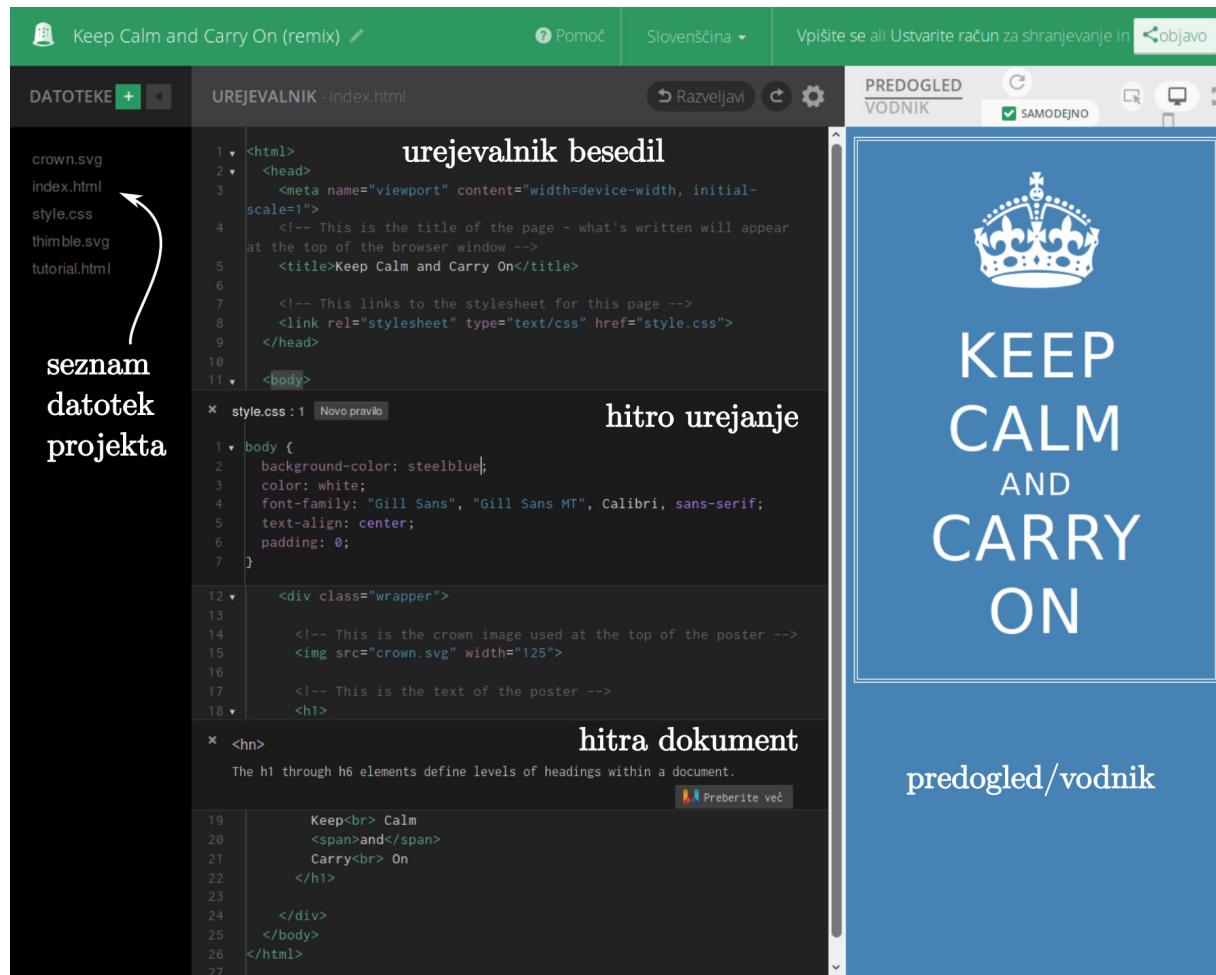
Tutorialspoint | <http://www.tutorialspoint.com/>

Vrsta vsebine	Osnovna kombinirana vsebina: vodič in preizkus programske kode. Posebej spletna aplikacija za učenje programiranja: Codingground.
Jezik spletnne strani	Angleščina: da, slovenščina: ne, drugi: ne.
Ponujena znanja	Znanja prog. jezikov + druge vsebine. Vodiči s številnih področij.
Programski jeziki	Velika knjižnica prog. jezikov.
Težavnostna stopnja	Srednja šola.
Upoštevanje načel	Problemski pristop: ne, sistematičnost: ne, postopnost: da (vodič).
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da. Ustvarjanje podporne programske kode v spletni aplikaciji za prog., vendar brez navodil in deljenja vsebine.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačen.

7.6 Thimble

Thimble [45] je spletni portal, ki ga gosti podjetje **Mozilla**, ki izdaja spletni brskalnik **Firefox**. Spletni portal je namenjen učenju spletnih tehnologij **HTML**, **CSS** in **JavaScript**. Pripravljenih je šest spletnih vsebin, ki jih lahko odpremo v projektu in jih preurejamo. Največja prednost

spletnega portala je spletna aplikacija oz. orodje, v katerem urejamo projekte (slika 31).



Slika 31: Urejanje projekta na strani *Thimble* [45].

Na desni strani spletne aplikacije imamo **seznam dokumentov**, ki sestavlja projekt. V projekt lahko dodajamo lastne dokumente. **Urejevalnik besedil** barva značke html in css jezika, prav tako upošteva barvanje besedila v primeru programskega jezika JavaScript. Urejevalnik ima dve priročni zmožnosti, kot je *hitri dokument* in *hitro urejanje*. Hitri dokument je takojšnja pomoč, ki se sproži ob pritisku kombinacije Alt + K na mestu, kjer je značka, za katero želimo dodatno razlago. Hitro ureja na mestu, kjer smo postavljeni v besedilu, s pritiskom kombinacije tipk Alt + E odpre podurejevalnik s css razredom, ki oblikuje značko dela html dokumenta. Funkcija hitro urejanje v css dokumentu, ko smo postavljeni na barvo, pomeni barve z barvne palete.

Vse spremembe, ki jih naredimo v urejevalniku, se v živo in samodejno posodobijo v **oknu predogleda**, ki se nahaja na levem delu. Na tem mestu najdemo tudi vodnika. Uvodna spletna stran in uporabniški vmesnik sta preveden v **slovenščino**. Žal pa vodniki, ki so pripravljeni na spletni strani, niso prevedeni v slovenščino.

Če na spletni strani opravimo registracijo in se prijavimo, se nam spremembe na projektu

shranjujejo samodejno. Projekt lahko preimenujemo in ga delimo z drugimi preko spletne povezave. Tisti, ki naš projekt odpre, ga spreminja kot lastnega in se sprememb v našem ne pozna. Ustvarjamo lahko tudi nove projekte, katerim dodajamo `html`, `css`, `js` datoteke. Dodamo lahko tudi datoteko vodiča, ki jo lahko poljubno spreminjam.

7.6.1 Povzetek

Spletni portal lahko uporabljamo na vseh stopnjah. Primeren je še posebej za uporabo v osnovni šoli pri izbirnem predmetu **računalniška omrežja**, saj omogoča preprost in učinkovit urejevalnik besedil. Učitelj se registrira in pripravi oz. prilagodi obstoječi projekt za pouk. Učencem deli povezavo. Učencem se ni potrebno registrirati in kljub temu lahko urejajo dokument kot lasten. Učenci svoje dokončane projekte delijo kot končen izdelek s povezavo nazaj učitelju. Na podoben način se lahko spletni portal uporablja tudi v srednji šoli.

Thimble | <https://thimble.mozilla.org>

Vrsta vsebine	Napredna kombinirana vsebina: vadnica (vodnik + spletna aplikacija za programiranje).
Jezik spletne strani	Angleščina: da, slovenščina: aplikacij, da; vodnik, ne. drugi: da.
Ponujena znanja	Znanje programskih jezikov, uporaba spletna in spletnih vsebin.
Programski jeziki	HTML, CSS, JavaScript.
Težavnostna stopnja	Osnovna šola (3. triada) in srednja šola.
Upoštevanje načel	Problemski pristop: da, sistematičnost: ne, postopnost: ne.
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da. Možno je ustvarjanje lastnih vadnic, ki jih lahko delimo naprej.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačen.

7.7 Code combat

Spletni portal *Code combat* [46] je mešanica med igranjem igre in pisanjem programske kode. V predstavitev spletne strani pravijo naslednje: “*Če se želiš naučiti programirati, moraš napisati veliko programske kode.*” In poudarjajo, da oni poskrbijo, da pri tem početju ostane zabava v ospredju [47]. Spletna stran ponuja tri načine registracije, ustvarite lahko **navaden**,

učiteljski ali **učencev** račun. V pregledu strani smo uporabili prijavo z navadnim računom, v nadaljevanju smo prav tako povzeli posebnosti ostalih dveh računov.

Po registraciji in prijavi v račun si izberemo **lik in programski jezik**, s katerim bomo igrali (slika 32). Spletna igra ponuja štiri programske jezike **Python**, **JavaScript**, **CoffeScript**, **LUA**.



Slika 32: Izbira junaka in programskega jezika [46].

Po izbiri junaka preidemo na izbor **zemljevidov** (slika 33). Izberemo lahko samo zemljevid, ki je odklenjen. Druge zemljevide odklenemo tako, da rešimo vse naloge v njem. Posamezen zemljevid predstavlja cilje posameznih programskih konceptov, ki se jih uporabnik nauči, ko predela vse naloge.



Slika 33: Izbor zemljevida, na katerem bomo reševali naloge [46].

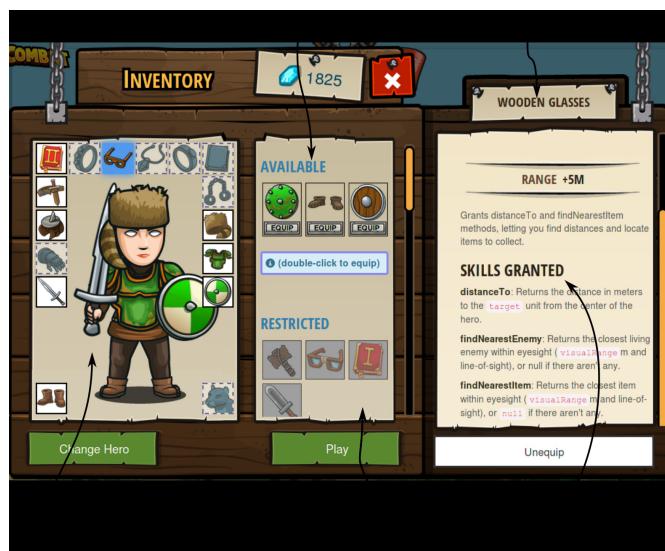
Igra se zgleduje po tipu iger igranja vloge (*ang. Role play game - RPG*). Z junakom napredujemo po zemljevidu z vsako opravljenou nalogou, ob koncu vsake naloge prejmemo **točke - izkušnje** in napredujemo v lastnih **stopnjah**. Junak nabira številne predmete, ki mu omogočajo nadgradnjo veščin in tako lažje napredovanje skozi misije. Za začetek naloge pritisnemo

na rdeče obarvan krog na zemljevidu (slika 34), prikaže se povzetek naloge in katere koncepte bomo uporabili pri reševanju naloge.



Slika 34: Podroben zemljevid za izbiro nalog [46].

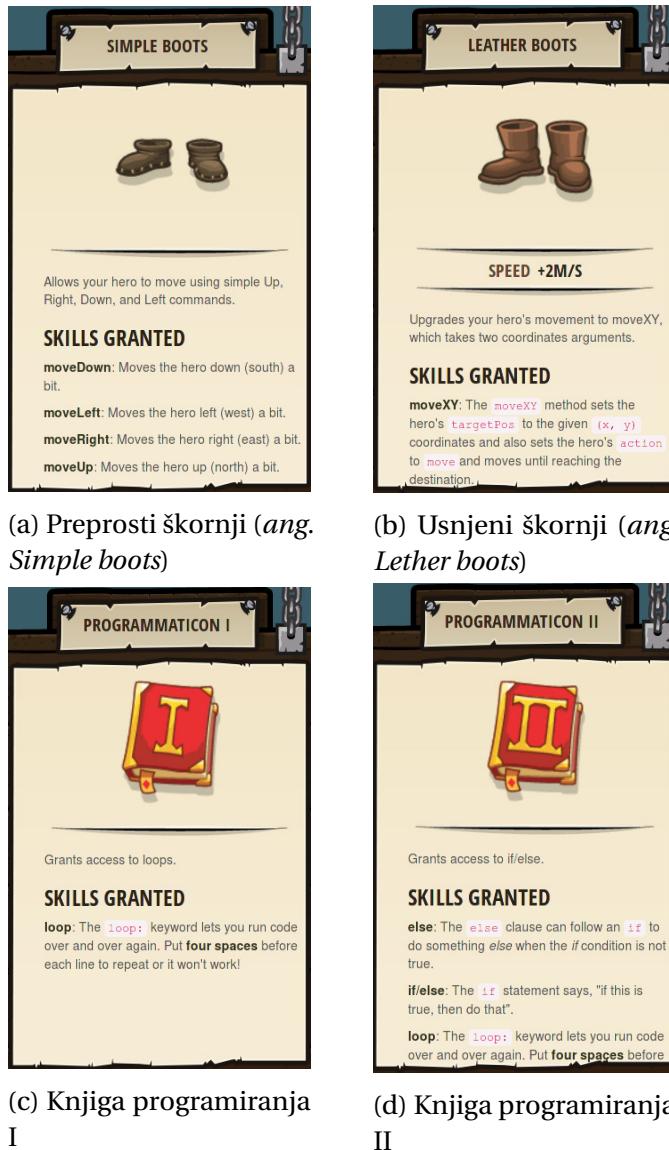
Sledi opremljanje junaka (slika 35). Igra tu pomaga v tolikšni meri, da omeji nekatere predmete, ki niso uporabni za trenutno naložbo. Sledimo logiki igre in izbiramo vedno najboljšo opremo, ki je na voljo.



Slika 35: Oprema junaka in njen opis [46].

Ko kažemo ta primer igre, smo z napredkom rešenih nalog skoraj na polovici drugega zemljevida in so oprema in veščine, ki jih uporablja naš junak, že napredne, zato naredimo primerjavo med prejšnjo in nadgrajeno opremo. S primerjavo škornjev (slika 36a in 36b) lahko povzamemo, katere metode je junak pridobil. Če je pri *preprostih škornjih* imel možnost gibanja le v smeri **levo, desno, gor in dol**, se lahko pri *usnjениh škornjih* giblje po najkrajši

poti na koordinate, ki jih podamo kot argument metode `hero.moveXY(x, z)`. S primerjavo *knjige za programiranje* med različico I in II (slika 36c in 36d), ki smo jo pridobili pozneje, je razlika med veščinami očitna. Če smo pri *knjigi za programiranje I* lahko uporabljali samo zanke loop: oz. `while True:`, pri *knjigi za programirane II* lahko zraven uporabljam še `if/else` stavek. Primerjali smo samo dva predmeta, junaku so na voljo številni predmeti z različnimi metodami za različne dele telesa, vse od **mečev, ščitov, kap, ur, pasa, obleke, očal** in tako dalje. Z napredovanjem veščin in naborom predmetov junak pridobiva na zmožnostih, prav tako se s tem postopno izboljšujejo veščine in se širi znanje uporabniku spletnne igre.



Slika 36: Primerjava med prejšnjo različico opreme in njeno nadgradnjo, ki jo lahko zamenjamo junaku [46].

Po vsakem zagonu igre sledi najprej prikaz cilja, ki ga moramo uresničiti. Postavitev igre (slika 37) je tako, da nalogo rešujemo v **urejevalniku besedil**, ki je na desni strani zaslona. Programská koda, ki jo izvajamo, se odvija v oknu na levi strani. Urejevalnik besedil omogoča nekatere

napredne funkcije, kot je **barvanje kode**, **samodejno zamikanje vrstic**, **prikaz zamika vrstic**, **sprotno opozarjanje na napačno sintakso** ter **predlogi za samodejno dokončanje** programske kode. Pri samem pisanju programske kode lahko zapišemo na primer samo del metode, kot je `find`, in se nam ob potrditvi samodejnega predloga izpiše celotna programska koda `enemy = hero.findNearestEnemy()`.



Slika 37: Postavitev igre [46].

V trenutni nalogi je cilj tak, da moramo napadati sovražnike, ko je ta bližje skrinji kot 10 m in ga moramo napasti z metodo `cleve`, če sovražnika ni v bližini, napadamo skrinjo. Ko smo zadovoljni s svojo rešitvijo, poženemo program in čakamo na končni izid (slika 38). Če je iztek programa uspešen in smo rešili nalogo, jo *posredujemo*.



Slika 38: Uspešno končan izid igre z napisano programsko kodo [46].

Ob koncu igre poberemo še **dosežke**, to so **točke izkušenj**, **diamante** in **značke** (slika 39).



Slika 39: Končni rezultat in pregled nad dobljenimi dosežki ob koncu igre[46].

Dostop do spletnega portala pa ni povsem brezplačen. Z **brezplačnim dostopom** lahko raziščemo 145 nalog v petih zemljevidih. Spletni portal ponuja **naročnino** 10 \$ na mesec, s katero lahko pridobimo dodatne **naloge, junake, diamante** in tako dalje.

7.7.1 Upravljanje razreda

Spletni portal omogoča **upravljanje razredov**. Razrede upravlja **učitelj**. Portal ima prilagojeno učno snov za tri stopnje po ameriškem **K-12** sistemu. Kot smo že primerjali šolske sisteme, lahko povemo, da so stopnje po starosti v slovenski šoli prilagojene na naslednje stopnje **osnovno šolo (2. triado in 3. triado) in srednjo šolo**. Upravljanje razredov (slika 40) je podobno, kot smo to videli pri **Code academy** (poglavlje 7.2.1). Učitelj ima nadzor nad dodajanjem učencev in ima pregled o napredku učencev. Z njimi preko portala ne more komunicirati.

The screenshot shows the 'Class Overview' section of the CodeCombat teacher dashboard. It displays basic class statistics: Language (Python), Students (1), Average level playtime (a minute), Total play time (3 minutes), Average levels completed (2.0), and Total levels completed (2). The class was created on 6/20/2016. On the right, there's a 'Adding students:' section with a 'DarkSpoonFoot' button (which copies a class code for others to join) and a 'Copy Class Code' button. Below it is another section with a URL (<https://codecomb>) and a 'Copy Class URL' button, along with a note that new students can visit this URL while logged in to join the class. At the bottom of this section is a 'Add Students Manually' button. Below these sections are tabs for 'Students', 'Course Progress', and 'Enrollment Status'. A dropdown menu 'Select course to view:' is open, showing options like 'Introduction to Computer Science' (which is selected and highlighted in blue), 'Computer Science 2', 'Computer Science 3', 'Computer Science 4', and 'Computer Science 5'. Below this is a 'Introduction to Computer Science: Course Overview' section showing student progress from 1 to 20. The third student, '3', is highlighted in yellow. At the bottom left, there's a 'Sort by: Name Progress' link, and the student 'jureneme' is listed with their email 'jureneme@gmail.com' and a progress bar from 1 to 20 where student 3 is also highlighted.

Slika 40: Učiteljev pogled na upravljanje razreda [46].

Dostop za šole ni brezplačen. Učitelj lahko zahteva demonstracijsko različico in v njo povabi neomejeno število učencev. V tej demo različici je na voljo samo prvi tečaj **Uvod v računalniško znanost**, vsi ostali so zaklenjeni. Za uporabo nadaljevalnih tečajev mora učitelj zaprositi za poizvedbo cene za nakup licence za posameznega učenca.

Učenec rešuje naloge podobno, kot smo to lahko videli pri navadnem računu, vendar ne more stopenj izbirati iz mape. Naloge oz. stopnje, ki jih rešuje, so prilagojene tečaju, v katerega ga

je vpisal učitelj. Po opravljeni nalogi učenec takoj nadaljuje na naslednjo stopnjo, kot je ta predvidena v tečaju. Učenec ima vpogled v naloge, ki ga čakajo v tečaju. V seznamu lahko izbere tiste naloge, ki jih je že opravil oz. tisto zadnjo, v kateri je ostal. Za nadaljevanje mora učenec rešiti prejšnjo nalogo. Če v navadnem računu lahko menjujemo različne dele oblačil in opreme, je to v načinu učenčevega načina onemogočeno. Učenčev lik ima na voljo stvari, ki jih potrebuje pri rešitvi naloge. S to omejitvijo je olajšano delo učitelja, saj se tako lahko razred osredotoči le na reševanje naloge.

Spletni portal ima prevode v več večjih svetovnih jezikov, žal trenutno slovenščina ni med njimi. V padajočem meniju, kjer učenci lahko spreminjače jezike, se najde tudi slovenščina, vendar se ob njenem izboru prikaže pojavno okno s pozivom za prevod v izbran jezik. Uporabniku se ponudi spremenjen račun, ki omogoča prevajanje v druge jezike. Torej obstaja upanje, da se bo mogoče nekoč nekdo lotil tega projekta in začel prevajati spletni portal in naloge v slovenski jezik.

7.7.2 Povzetek

Spletni portal oz. spletna igra ima vsekakor veliki motivacijski faktor. Naloge so narejene sistematično in postopno. Čeprav se v naših osnovnih šolah ne priporoča učenje programskih jezikov, kjer pišemo programsko kodo, razen morda **Logo**, naj bi se uporabljali vizualni programski jeziki, kot je **Scratch**. Za ta spletni portal verjamemo, da bi ga lahko uporabili pri osnovnošolcih, ki bi lahko poučevali **Python**. Težava je edino **slovenščina**, saj na strani še ni prevoda nalog, in **plačljivost** za uporabo razredov. To učitelj sicer lahko zaobide z uporabo **navadnih računov** in uporabo brezplačnih vsebin, vendar mu to uteži delo, ali pa zaprosi za demonstracijsko različico in v ta namen izkoristi vsebine, ki so na voljo brezplačno.

Codecombat | <https://codecombat.com/>

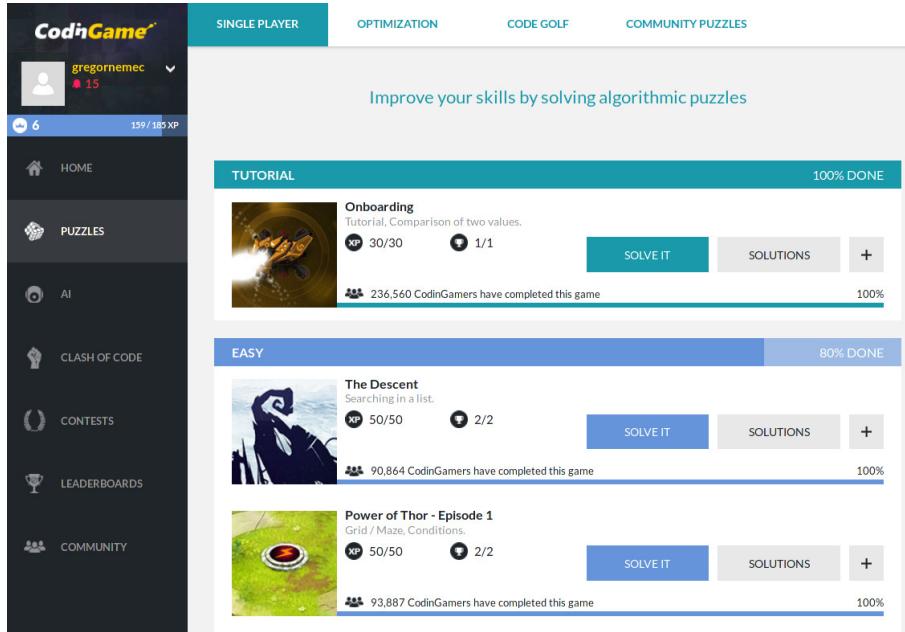
Vrsta vsebine	Spletna igra za programiranje. Kombinacija igre RPG + pisanje programske kode.
Jezik spletne strani	Angleščina: da, slovenščina: ne, drugi: da.
Ponujena znanja	Znanja programskih jezikov in algoritmov.
Programski jeziki	Python, JavaScript, CoffeScript, LUA.
Težavnostna stopnja	Osnovna šola (2/3 in 3/3) in srednja šola.
Upoštevanje načel	Problemski pristop: da, sistematičnost: da, postopnost: da.
Dosežki/Gamification	Da (značke, izkušnje, diamanti).
Dodajanje lastnih vsebin	Ne.
Upravljanje razreda	Da (za osnovni tečaj brezplačno, za ostale tečaje plačljiva licenca).
Dostop vsebin	Polplačljiv: brezplačnih je 145 nalog, plačljive so dodatne naloge - 95, dodatni diamanti ... za (9,99\$/mesec).

7.8 Codingame

Spletni portal *Codingame* [48] je spletna igra, katere bistvo je, da uporabnik rešuje probleme, ki so zahtevni in so predstavljeni kot izzivi v obliki igre. Uporabnik s podajanjem rešitev izboljšuje veščine programiranja in znanje algoritmov.

Za vsakega uporabnika se igranje oz. reševanje problemov začne pri **sestavljkah** (*ang. puzzles*) (slika 41). V enoigralskem načinu so igre razdeljene na več težavnosti, vse od **lahkih** do **zelo težkih**. Na tej strani lahko izbiramo med sestavljkami, kjer imamo nalogu, da programsko kodo čim bolj **optimiramo** (*ang. optimisation*), torej poiščemo rešitev, ki potrebuje najmanjšo časovno in prostorsko zahtevnost. Druga vrsta iger je taka, da s čim manj kode, torej znanjem trikov programskega jezika, rešimo problem na strani *ang. Code golf*. Imamo še možnost, da rešujemo naloge, ki jih je pripravila skupnost, na strani *ang. Community puzzles*. Na tej strani se lahko posredujejo lastne naloge, ki pa morajo biti odobrene s strani razvijalcev spletnih strani, preden se objavijo.

Poglejmo, kako rešujemo problem. V ta namen smo izbrali sestavljanko **Most - Episoda I** (*ang. The bridge - Episode I*) (slika 42). Za reševanje naloge lahko izbiramo med številnimi programskimi jeziki, kot so **C#, C++, Java, JavaScript, Python3, Bash, C, Clojure, Dart, F#**. V našem primeru bomo uporabili programski jezik **Python3**.



Slika 41: Podstran Codingame [48] - sestavljanke.

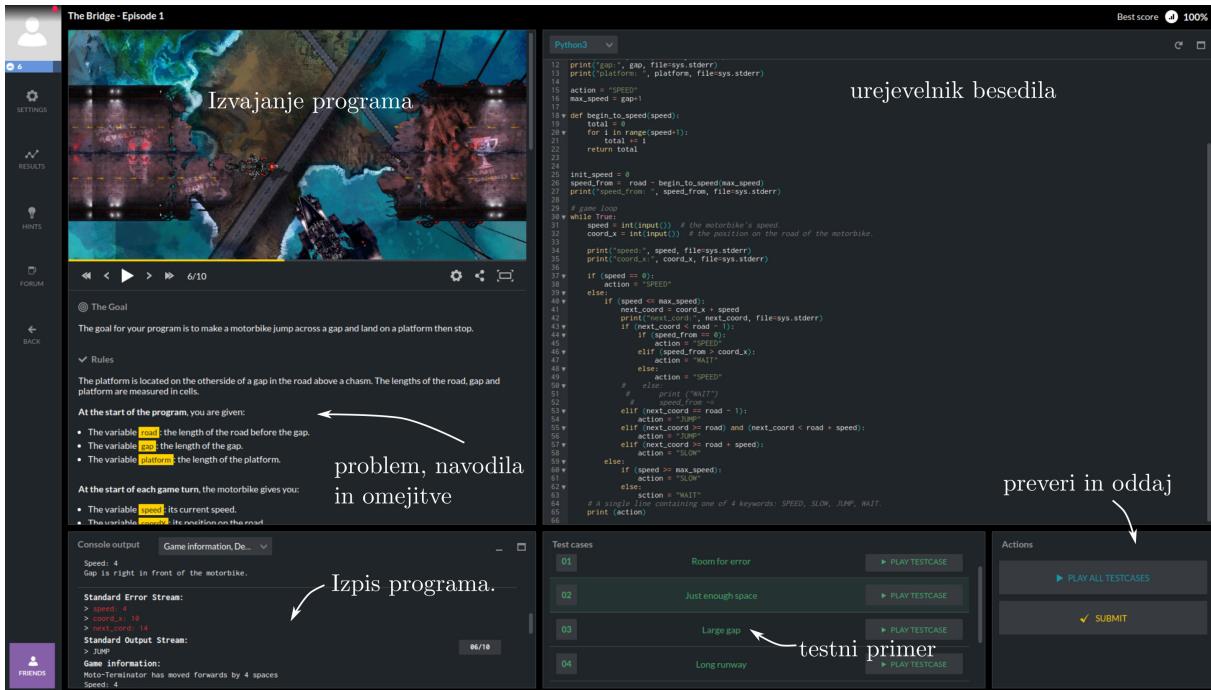
Primer izbrane naloge pravi, da moramo napisati program, ki upravlja hitrost motorja, ki se vozi po mostu in mora preskočiti prepad ter se na koncu pravočasno ustaviti. Celoten program teče v neskončni while True zanki. Na vsakem koraku moramo izpisati, kaj je naslednja akcija motorja ali naj POSPEŠI, ČAKA, ZAVIRA ali SKOČI. Na vsakem koraku se hitrost poveča oz. zmanjša za 1. Hitrost se na vsakem koraku odraža za razdaljo, ki je enaka hitrosti. Vhodni podatki programa so naslednji, cesta, ki je razdalja pred prepadom, naslednji podatek je velikost prepada in razdalja zaviralne poti.

Na levi strani zgoraj **opazujemo potek rešitve**. Levo v sredini so napisana **navodila**, ki natanko opisajo problem, *vhodne podatke, omejitve* in pričakovani *izpis programa* na vsakem koraku. Levo spodaj je *ukazna vrstica*, na kateri spremljamo izpis programa in ki nam je v pomoč pri razhroščevanju.

Na desni strani je **urejevalnik besedil**, v katerega pišemo rešitev oz. program. Urejevalnik zna barvati programsko kodo, samodejno upošteva zamik programske kode in podaja samodejne predloge za metode, ki so vgrajene, ko napišemo “.”. V nastavitevah urejevalnika lahko izbiramo način, ki prilagodi vedenje in ga lahko nastavimo na *klasičen, emacs* in *vim* način. Izbiramo lahko med temno in svetlo barvno shemo. Nastavimo lahko samodejno zaključevanje oklepajev.

Spodaj desno poganjamo program s **testnimi primeri**, ki rešitev testirajo v skrajnih mejah. Testi, ki so podani kot vhodni podatki, so narejeni tako, da onemogočajo, da bi napisali program, ki bi imel statično vgrajene rešitve. Ko rešitev prestane vse teste, jo lahko posredujemo.

Pri reševanju problemov se uči tudi življenski krog razvijanja programske opreme in strategije



Slika 42: Reševanje sestavljanke *The Bridge* [48]

reševanja problemov, kot smo jih opisali v poglavju 4.2, saj **problem najprej analiziramo, izberemo pristop, ga razgradimo, razvijemo algoritem in testiramo njegovo pravilnost**. Ta postopek lahko ponavljamo v krajših korakih, dokler ne pridemo do prave rešitve in ponovimo faze razvoja in testiranja algoritma večkrat. Ko algoritem reši problem, se lahko ukvarjamo z njegovo **učinkovitostjo**, ki jo pridobimo z upoštevanjem lastnosti implementacije algoritmov, torej časovne in prostorske zahtevnosti ter značilnosti programskega jezika in njegovih zmožnosti. Na tak posreden način nas spletni portal uči programski jezik, saj rešitev od nas zahteva optimalno rešitev, ki jo lahko dosežemo z dobim poznavanjem programskega jezika in njegovih knjižnic.

Med reševanjem si lahko pomagamo z *namigi* in *pseudo kodo*. Ko rešitev oddamo, sledi izračun točk in nagrad v obliki **točk izkušenj** in **značk**. Uporabnik lahko brska med rešitvami drugih uporabnikov in tako nabira dodatno znanje.

Na spletni strani obstajajo še drugi načini igranja. Lahko se včlanimo v različne igre, kjer programiramo **bota**, ki tekmuje z **umetno inteligenco** na podstrani *ang. AI. Spopad kode* ali *ang. Clash of code* je način, v katerem do 8 igralcev tekmuje eden proti drugemu v bojih po 5 ali 10 minut. Vsak zase rešuje problem, na koncu zmaga najbolj točkovana rešitev. Spletni portal prireja tudi uradna **tekmovanja**, na katerih se brezplačno registriramo. Izbiramo lahko med dvema sistemoma nagrajevanja, ali so nagrade v fizični obliki, kot je npr. 3D-tiskalnik, majice in podobno, ali se odločimo za način, kjer se potegujemo za razgovor na določenem delovnem mestu, ki je takrat razpisano. To je dober koncept prepoznavanja talentov in iskanja službe.

7.8.1 Povzetek

Spletni portal je namenjen predvsem treniranju večin programiranja in učenju algoritmov. Uporabnik mora poznati več kot le osnove programiranja, zato je spletni portal namenjen predvsem za srednje šole, višje in univerzitetne programe študija. Spletna igra ni primerna za pouk, je pa lahko zelo dobro dopolnilo za dodatno vajo ter izpopolnjevanje lastnih izkušenj programiranja. Priporočamo ga lahko vsakemu, ki ga zanima programiranje. Sistem nalog in reševanja je dobro zasnovan. Problemsko je zasnovano tudi ozadje vsake naloge. Naloge si po težavnosti sledijo postopoma, sistematično so naloge razmetane po vseh težavnostih stopnjah, zato ne moremo potrditi načela sistematičnosti.

Codingame | <https://www.codingame.com>

Vrsta vsebine	Vadnica: spletna aplikacija za programiranje + reševanje problemov + testiranje).
Jezik spletne strani	Angleščina: da, slovenščina: ne, drugi: da.
Ponujena znanja	Znanje algoritmov in večine programiranja.
Programski jeziki	C#, C++; Java, JavaScript, Python3, Bash, C, Clojure, Dart, F# .
Težavnostna stopnja	Srednja šola, visoki, višji in univerzitetni.
Upoštevanje načel	Problemski pristop: da, sistematičnost: ne, postopnost: da.
Dosežki/Gamification	Da, značke, izkušnje, rangiranje.
Dodajanje lastnih vsebin	Da. Dodajanje lastnih nalog/problemov, ki jih rešuje skupnost.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačno

8 Možni načini uporabe spletnih portalov pri pouku

V spodnjem sestavku izpostavimo nekatere skupne prednosti, ki jih lahko prinese uporaba spletnih portalov za učenje programiranja pri pouku. S pregledom spletnih portalov, ki so nastali na univerzah (poglavlje 5), in tistih, ki smo jih podrobno pregledali (poglavlje 7), lahko ugotovimo naslednje prednosti.

Namestitev programske opreme ni potrebna, kar je zagotovo velika prednost. Mentorju praktično ni treba nameščati nobenega urejevalnika besedil, IDE, niti prevajalnika ali tolmača. Prav tako ni potrebe po nastavljanju sistemskih poti, ki jih mnogi prevajalniki zahtevajo. Uporaba SPUP je neodvisna od uporabe operacijskega sistema. Vsaka naprava, na kateri lahko poganjamo novodobni spletni brskalnik, omogoča uporabo SPUP. Učencem na domačih računalnikih ni treba namestiti nobene programske opreme. Na tem mestu bi poudarili, da mora biti učitelj previden pri dajanju domače naloge z uporabo računalnika. Zares se mora prepričati, da imajo to zmožnost vsi učenci. Najbolje je, da lahko vse dodatno delo, ki ga učitelj predvidi, učenci opravijo v šolski računalniški učilnici.

Seznanjanje s programsko opremo pri pouku ni potrebno. Znebimo se potrebe, da bi z novinci morali spoznavati urejevalnik besedil ali kompleksno IRO. Spoznavanje programskega jezika in reševanja problemov se lahko lotijo nemudoma.

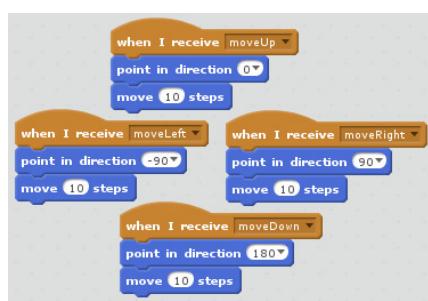
Pisanje programa od začetka do konca ni potrebno. Večina spletnih portalov, ki ponujajo vsebine v obliki vadnice, imajo programske naloge pripravljene tako, da mora uporabnik vnesti le del programske kode. Za novinca to pomeni, da se osredotoči le na nalogu in del sintakse, ki jo v danem trenutku potrebuje, da reši zadano nologo. To prednost smo že spoznali na SPUP avstralske univerze, kjer so uporabili tip naloge zapolni prazna mesta [7].

Pripravili smo tudi konkretna primera za OŠ in SŠ, kako bi s spletnim portalom lahko uresničili učne cilje, ki jih najdemo v učnem načrtu.

8.1 Primer uresničevanja ciljev učnega načrta v osnovni šoli

Učni načrt za računalništvo - neobvezni izbirni predmet [3] vsebuje vsebinski sklop **Algoritmi**, ki smo ga povzeli v poglavju 2.3.2. Spletni portal *Code combat* [46] lahko učitelj uporabi pri **ponovitvi sklopa**, ko je snov že predelal na primer s programskim jezikom in spletnim orodjem *Scratch* [39]. K obstoječim ciljem, ki so podani v učnem načrtu, lahko dodamo še naslednje: *učenci spoznajo osnove programskega jezika Python*. Ker se učenci s pisanim programskim jezikom srečujejo prvič, bi bilo dobro, da učitelj pripravi učne liste s primerjavo med *Scratchom*

in sintakso **Pythona** oz. metodami, ki se uporablja na portalu (slika 43b). V tem primeru niti ni pomembna podrobna obravnava programskega jezika Python. Pomembno je, da učenci razumejo, kaj posamezna vrstica programske kode izvrši, kateri del kode predstavlja junaka in kateri akcijo junaka. To lahko pokažemo nazorno z delom programske kode v **Scrachu** (slika 43a), ki ponazarja metode, ki jih uporabljam za vodenje junaka na spletni strani *Codecombat* (slika 43b).



(a) Programska koda, napisana v **Scrachu** [39] za premikanje *gor*, *levo*, *desno*, *dol*.



(b) Programska koda v **Pythonu**
klica predpripravljene metode, ki premikajo junaka *gor*, *levo*, *desno*, *dol*.
[46]

Slika 43: Primerjava programskih blokov, napisanih v **Scrachu**, in enačic klica metod, napisanih v **Pythonu** na spletni strani *Codecombat*.

Kot smo že omenili v predstavitvi spletnega portala *Codecombat*, ima učitelj dve možnosti:

- učenci uporabljajo svoje ustvarjene račune,
- učitelj ustvari **demo** razred in v njega vključi učence.

Glavna slabost v prvem primeru je ta, da tako lastno ustvarjeni računi učencev omogočajo izbiro junakov, predmetov, ki jih uporabljajo pri reševanju nalog, in se zato lahko pri učni uri zaplete pri nadzoru napredka pri posameznem učencu in je s tem delo učitelja oteženo. Zaradi teh razlogov se bolj nagibamo k drugi možnosti.

Učitelj lahko s prošnjo zaprosi za **demonstracijski** račun, v katerem ima možnost ustvariti razred s prvim vsebinskim sklopom **Uvoda v računalniško znanost** (ang. *Introduction to Computer Science*). To je edini sklop v tem načinu, ki je brezplačen in ga lahko uporabimo v šoli pri nas brez dodatnih stroškov. Učitelj učence povabi v razred z edinstveno ustvarjenim **URL** naslovom. Registracija po kliku na povezavo je potrebna. Učenci se od naloge do naloge premikajo brez povzetka prejema dosežkov, izbire predmetov in izbire opreme za svojega junaka. Za tako delo je primeren tip učen ure **praktično vodeno delo**, kjer učitelj delo vodi z vnaprej pripravljenimi učnimi listi primerjave programskih blokov (slika 43) in povzetki navodil v slovenščini. Ko učenci v začetnih nalogah osvojijo osnovne metode junaka, lahko učitelj stopnjuje samostojnost dela učencev. Delo na spletnem portalu je primerno za dvojno učno uro oz. **90 min**. Tiste naloge, ki jih učenci ne uspejo dokončati, jih lahko učitelj dodeli

za domače delo.

Sklop vsebinsko predstavlja naslednje koncepte programiranja: **osnove sintakse, argumente, spremenljivke, tip spremenljivke, string, while zanka**. Pri preverjanju sklopa nekaterih operativnih ciljev ne moremo preveriti, zapisali smo tiste, ki jih ni moč preveriti, in zakaj:

- **algoritem predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku.** Na spletni strani algoritmi niso predstavljeni z diagrami, lahko jih sicer pripravi učitelj, vendar v tej fazi to ni nujno potrebno;
- **znajo v algoritem vključiti vejitev.** Uporaba vejitev v tem prvem sklopu **CS1** na spletnem portalu ni omogočena;
- *znajo uporabiti nekatere ključne algoritme za sortiranje in iskanje;*
- *poznajo osnovne algoritme za iskanje podatkov.*

Menimo, da ostale cilje lahko uspešno povzamemo in preverimo v učni uri ponovitve vsebinskega sklopa. Čeprav uporaba spletnne igre predstavlja kar nekaj priprave učitelja, ima izjemno motivacijsko vrednost, ki lahko učence navduši za nadaljnje delo.

8.2 Primer uresničevanja ciljev učnega načrta v srednji šoli

Učni načrt **informatike** na programu splošne gimnazije predvideva učenje algoritmov in programiranja pa ravni **posebnih znanj** v tematskem sklopu obdelave podatkov. Predpostavimo, da mentor oz. profesor informatike uresničuje znanje algoritmov in programiranja, ta lahko uporabi spletni portal *Codeacademy* [38] v namene domačega dela. Profesor mora upoštevati medpredmetno povezovanje s predmetom angleščina in za dani sklop pripraviti prevode navodil, ki so lahko dijakom v pomoč. Prednost portala je še ta, da lahko vsebinske sklope in posamezne teme, ki so na voljo, poljubno izbiramo ne glede na vrstni red oz. na to, kaj smo že predelali. Kako lahko profesor izkoristi uporabo sledenju napredka dijakov, smo že podrobno opisali v poglavju 7.2.1.

Spletni portal lahko uporabi pred obravnavo neke snovi in se tako dijaki že seznanijo z določeno funkcionalnostjo in sintakso programskega jezika. Ker dijaki že del znanja in sintakse usvojijo, so naloge pri pouku lahko zahtevnejše in je lahko večji poudarek na strategiji reševanja (težjih) problemov. Konkretno, za domače delo lahko dijaki rešujejo nalogo "*Strings & Console Output*", kjer usvojijo osnovno znanje ravnjanja s **pisanjem izhodnih podatkov** in **podatkovnega tipa string** oz. niza. Naučijo se uporabljati metode, ki jih ponuja podatkovni tip **string**, lepijo dve besedi skupaj in tako dalje. Kot že rečeno, z osvojenim znanjem lahko pozneje pri pouku rešujejo zahtevnejše naloge.

V drugem primeru lahko ta portal uporabi po obravnavi določenega vsebinskega sklopa in

vadnico na portalu *Codeacademy* [38] izkoristi, dijaki z domačim delom utrdijo svoje znanje. Poglejmo konkreten primer. Dijaki na primer imajo po **obravnavi sklopa vhodni podatki**, **podatkovni tip**, **string** in **if else vejitve** zadostno znanje, da rešijo nalogo “*PygLatin*”. Napisati morajo program, ki sprejme besedo in jo pretvori po določenem pravilu slovarja ter jo izpiše na zaslon. Naloga je razdeljena na več vadnic, dijaki nalogo rešujejo korak po koraku. Napredek reševanja naloge lahko spremi profesor preko portala.

9 Zaključek

V diplomskem delu smo najprej spoznali, da učenje računalniške znanosti in programiranja spada v **primarno področje uporabe računalnika v izobraževanju**. Pri nas se je na splošnem izobraževalnem področju uveljavila usmeritev, ki zagovarja **splošno usposobljenost** za delo z računalnikom, kljub temu se z računalniško znanostjo in programiranjem večina učencev prvič sreča pri izbirnih predmetih *urejanje besedil, multimedija in računalniška omrežja* in pri neobveznem izbirnem predmetu *računalništva v OŠ*. Dijaki se srečajo s programiranjem v 1. letniku pri predmetu *informatika*. Posebnost so strokovni programi, katerim je osnova računalništvo in je poudarek na programiranju dosti večji, kar smo spoznali pri pregledu učnega načrta predmeta *računalništva* tehniške gimnazije. Ugotovimo lahko, da je programiranje pri splošnih predmetih zastopano do te mere, da se učenci oz. dijaki dotaknejo teh znanj, pri čemer je v bolj strokovnih predmetih, kot je pri *računalništvu* v OŠ in na programu tehnične gimnazije, zastopano v vseh podrobnostih.

Pri pregledu osnovnih pojmov smo spoznali, kaj predstavljajo programske paradigme, in ugotovili, da danes prevladuje paradigma **objektno orientiranega** programiranja. Podrobneje smo predstavili značilnosti programskih jezikov, ki so trenutno najbolj popularni, in ti so **Java, C++, JavaScript in Python**. Čeprav so v slovenskem izobraževalnem prostoru na srednješolskem nivoju prisotni vsi omenjeni, se za začetnike priporoča uporaba **Pythona**. V osnovni šoli bi se naj uporabljal programski jezik **Scratch**, ki uporablja grafičen način sestavljanja gradnikov. Pri spoznavanju osnovnih konceptov programiranja smo pripravili nekaj primerov programov, ki jih skušajo čim bolj nazorno predstaviti.

Pri proučevanju **aktivnega pristopa** smo spoznali, da mora biti pouk računalništva pozitivno naravnан in naj bo znanje grajeno z **aktivnim učenje**, torej učenci z lastnim delom odkrivajo in gradijo miselni model. Pri tem se naj uporablja **konstruktivne metode** poučevanja. Za aktivno učenje je ravno tako značilen problemko naravnан pouk. Reševanje nalog oz. raznovrstnih problemov je naravno prisotno pri poučevanju računalniške znanosti, zato smo izpostavili **strategije reševanja problemov** in smo osnovne korake podrobno razdelili. Strategije reševanja problemov niso pomembne le za področje računalništva, temveč za mnoga področja tehničnih in znanstvenih disciplin ter na sploh pri vsakdanjem življenju. Zanimalo nas je, ali so vse te značilnosti pouka računalništva zajete tudi z uporabo spletnih portalov.

Prvi spletni portali za učenje programiranja so nastali v akademskem okolju na različnih univerzah po svetu. Spoznali smo, da se študenti novinci vsepovsod srečujejo s podobnimi težavami, kot je namestitev in nastavitev programske opreme, uporaba IRO, uporaba sintakse programskega jezika, razumevanje prevajalnika in tako naprej. Pri vseh treh spletnih portalih, ki smo jih pregledali, smo lahko strnili naslednje značilnosti, glavni del vsakega spletnega portala je **spletna aplikacija za učenje programiranja (SAZP)**. Njene značilnosti

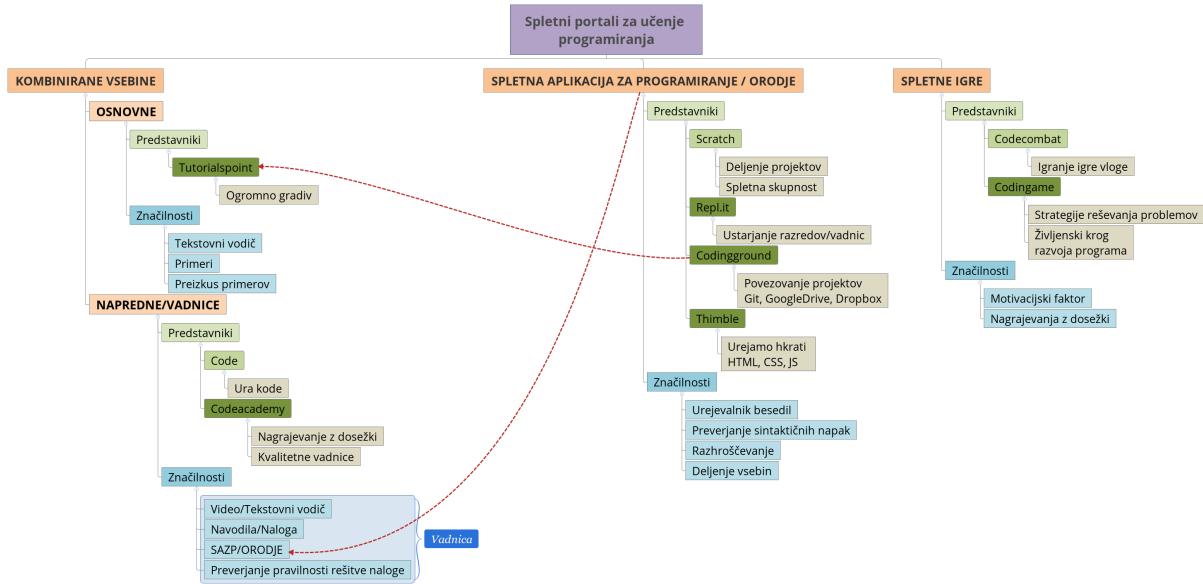
so, da vsebuje **urejevalnik besedil**, omogoča zagon napisanega programa, vrača povratne informacije o napakah ter izpis vhodno-izhodnih podatkov. Drugi elementi spletnega portala, poleg SPZP, so še **razdelane vsebine z nalogami**, omogočena je komunikacija med mentorjem in novincem, omogočen je pregled nad napredkom novinca.

Najzahtevnejši del tega diplomskega dela je bilo iskanje kriterijev, s katerimi smo lahko spletné portale klasificirali in jih ovrednotili. Najprej smo jih razdelili po vsebin in ugotovili, da so najzanimivejše tiste vsebine, ki so **kombinirane**. Kombinirane vrste vsebin smo razdelili na **osnovne in napredne**, v slednjih se prepleta **SPZP**, tekstovni ali/in video **vodiči** ter **navodila**. Vsi ti elementi naprednih kombiniranih vsebin tvorijo **vadnico**, ki je navadno osnovna enota neke učne teme. Za posamezni spletni portal želimo določiti, v katerem **jeziku** je predstavljen, katera **znanja ponuja**, ali so to veščine programiranja, znanja algoritmov, ali morda druga projektna znanja, kot je izdelava spletnih strani. Za vsak spletni portal smo zapisali, učenje katerih **programskih jezikov** omogoča, kakšni **težavnostni stopnji** je namenjen. Podrobnejše nas je zanimalo, ali spletni portali, tisti z vsebino, znajo upoštevati nekatera **učna načela**, kot je **problemski pristop**, **načelo sistematičnosti** in **načelo postopnosti**. Pri posameznem spletnem portalu smo izpostavili, ali ta uporablja **ocenjevanje**, **dosežkov**, ki je značilno za video igre. Zanimalo nas je še, ali omogočajo spletni portali, da **dodajamo lastne vsebine** ter ali je omogočeno **upravljanje razreda**. Nazadnje nas je zanimalo, ali so gradiva na spletnih portalih **brezplačna**, **polplačljiva**, torej so nekatera gradiva ali storitve brezplačne, druge plačljive, in **popolnoma plačljive**.

Spletih portalov, ki želijo poučevati računalniške znanosti in programiranja, je na spletu ogromno, zato smo morali določiti omejitve, s katerimi smo izbrali tiste, za katere smatramo, da nam bodo pri pouku najbolj v pomoč. Te omejitve so naslednje, spletni portal mora vsebovati **SAZP**, ki pa lahko nastopa tudi samostojno brez vsebine, vrsta vsebin naj bo **kombinirana** ter naj bodo vsebine na spletnih portalih dosegljive **brezplačno** ali **polplačljivo**. Veliko spletnih portalov, ki na prvi pogled vabijo s kakovostno pripravljeno vsebino in vadnico, zaradi popolne plačljivosti nismo mogli uvrstiti na seznam podrobne obravnave, saj zaradi tega razloga niso uporabne za pouk.

V diagramu (slika 44) smo zbrali povzetek pregleda spletnih portalov. Portale smo razvrstili po vrsti vsebine. Vsaki smo dodali predstavnike, ki smo jih pregledali, povzeli smo skupne značilnosti vsake vrste in za posamezni portal pripisali izstopajoče lastnosti.

Najprej povzemimo spletnne portale **kombiniranih vsebin**, ti se ponašajo s kakovostno pripravljenimi **vadnicami**. Posamezni vsebinski sklopi so razdeljeni na manjše enote, ki so med seboj sistematično povezane. Taki spletni portali nudijo tudi ideje in pripravljena gradiva za mentorje. Pregledali smo portal *Code*, ki je uporaben predvsem za uvodne učne ure računalništva v OŠ, tudi v razredih prve triade. Spletni portal *Codeacademy* je primeren predvsem v SŠ kot dopolnilno gradivo učenja programskih jezikov, kot je **Python** in osnovnih konceptov



Slika 44: Miselni vzorec s povzetkom pregledanih spletnih portalov s predstavniki in značilnostmi za posamezno vrsto vsebine.

programiranja. Glavna težava teh portalov je ta, da so gradiva napisana v **angleškem jeziku**. Mentor mora zato imeti pripravljene prevode navodil nalog. Učne ure z uporabo teh spletnih portalov lahko uporablja z medpredmetno povezavo z angleščino.

V posebno kategorijo smo uvrstili spletne portale, ki ponujajo samostojno **spletno aplikacijo za programiranje**, ki jo uporabimo kot orodje. Ti portali navadno ne vsebujejo lastnih vsebin. Kot prvo smo pregledali spletno aplikacijo **Scrach**, ki je v OŠ že dobro poznana. Scrach ima številne zmožnosti in bi ga lahko uporabili takoj po uvodu, ki ga naredimo s spletnim portalom **Code**. Naslednja SAZP **Repl.it** omogoča ustvarjanje razredov in lastnih vadnic. V vadnicah lahko vključimo teoretični uvod, navodila ter ogrodje programa **Codingground**, ki je del spletnega portala **Tutorials point**, omogoča dober urejevalnik besedil z ukazno vrstico ter dobro povezljivost z oblačnimi shrambami, kot so **Dropbox**, **Google Drive**, **OneDrive** in nadzor različice **Git**. **Thimble** omogoča ustvarjanje projektov z več ločenimi datotekami, projekt delimo naprej, tako da jih uporabniki ali učenci lahko spreminjajo. Izkaže se za odlično orodje učenja spletnih tehnologij. Bistvena prednost uporabe SAZP je ta, da ima mentor svobodno izbiro, katero vsebino bo podajal, vendar mora v priprava vsebine vložiti več truda in časa. Vsebino prilagaja in jo priepla po potrebi učnega načrta ter lastni presoji zmožnosti učencev.

Posebna kategorija spletnih portalov za učenje programiranja so **spletne igre**. Izpostavili smo dve, prva **Codecomba** je namenjena višjim razredom OŠ ter SŠ. Čeprav smo govorili o tem, da pisani programske jeziki niso primerni za uporabo v OŠ, ta spletni portal predstavlja izjemo, kar smo povzeli s primerom možnih načinov uporabe spletnih portalov. Spletne igre temelji na **igranju igre vloge**, igralec prevzame vlogo junaka, ki ga vodi in se z njim bojuje

s pisanjem programske kode. Junak ima omogočene veščine, torej metode programskega jezika, ki jih uporablja in so vgrajene v različne predmete ali opremo. Drug primer spletnne igre je *Codingame*, ki sodi v višjo zahtevnost in je namenjen predvsem tistim, ki želijo svoje znanje in veščino programiranja, poznavanje programskega jezika ter algoritmov izpopolniti z dokaj zahtevnimi problemsko zastavljenimi nalogami. Z reševanjem nalog se uporabnik uri ne le v strategijah reševanja problemov in pisanja algoritmov, temveč tudi optimizaciji napisanega algoritma. Spletna aplikacija prav tako uči življenski krog pisanja programa. Skupna značilnost spletnim igram je ta, da imajo močan motivacijski faktor, ki je včasih zelo potreben, da novinci ostanejo pri pisanju programske kode in reševanju problemov.

V splošnem lahko trdimo, da je uporaba spletnih portalov za učenje programiranja problemsko naravnana in v sami osnovi vzpodbuja aktivnost učencev. Uporaba prinaša prednosti za mentorja in novince, kot je ta, da ni potrebe po namestitvi programske opreme, kar omogoča kompatibilnost na številnih platformah, saj lahko spletno aplikacijo naložimo v vsakem novodobnem brskalniku. Novince ni potrebno priučiti uporabe zahtevnih uporabniških vmesnikov IRO. Navadno je vsebina spletnih portalov nastavljena tako, da jim ni potrebno pisati celotne programske kode, saj je ogrodje že podano in morajo napisati samo zahtevno rešitev. Tako se lahko bolj osredotočimo na reševanje problemov. Kot smo v diplomskem delu spoznali, so spletni portali za učenje programiranja uporabni, vsak od njih ima svojo prednost. Na mentorju je naloga, da skuša čim bolj izkoristiti te prednosti in s tem poskušti navdušiti čim več novincev, da bodo postali dobri programerji.

Literatura in viri

- [1] Gerlič, Ivan, *Sodobna informacijska tehnologija v izobraževanju*, DZS, Ljubljana, 2000.
- [2] Vladimir Batagelj et al., *UČNI načrt, Izbirni predmet: Program osnovnošolskega izobraževanja, Računalništvo*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2002. Pridobljeno 2. 4. 2016 iz, http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/predmeti_izbirni/Racunalnistvo_izbirni.pdf
- [3] Radovan Kranjc et al., *UČNI načrt, Program osnovnošolskega izobraževanja, Računalništvo: neobvezni izbirni predmet*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2002. Pridobljeno 2. 4. 2016 iz, http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/program_raszirjeni/Racunalnistvo_izbirni_neobvezni.pdf.
- [4] Wechtersbach Rado, *UČNI načrt, Informatika [Elektronski vir]: gimnazija: splošna, klasična, strokovna gimnazija: obvezni predmet (70 ur), izbirni predmet (210 ur), matura (70 + 210 ur)*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2008. Pridobljeno 2. 4. 2016 iz, http://www.mss.gov.si/fileadmin/mss.gov.si/pageuploads/podrocje/ss/programi/2008/Gimnazije/UN__INFORMATIKA_gimn.pdf.
- [5] Predmetna komisija Tea Lončarič et al., *Računalništvo [Elektronski vir]: gimnazija, tehniška gimnazija: izbirni strokovni maturitetni predmet (280 ur)*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2010. Pridobljeno 2. 4. 2016 iz, http://eportal.mss.edus.si/msswww/programi2010/programi/media/pdf/un_gimnazija/tehniska-gimnazija/UN_Racunalnistvo.pdf.
- [6] O. Hazzan, T. Lapidot, N. Ragonis, *Guide to Teaching Computer Science*, Springer, 2011.b
- [7] Nghi Truong, *A web-based programming environment for novice programmers*, Queensland University of Technology, Australia, 2007.
- [8] Wikipedia contributors, *Computer program*, Wikipedia, The Free Encyclopedia. Pridobljeno 25. 4. 2016 iz, https://en.wikipedia.org/wiki/Computer_programming.
- [9] Wikipedia contributors, *Algorithem*, Wikipedia, The Free Encyclopedia. Pridobljeno 25. 4. 2016 iz, <https://en.wikipedia.org/wiki/Algorithm>.
- [10] Jonah Bitautas, *The Differences Between Programmers and Coders*, Workfunc. Pridobljeno 26.4.2016 iz, <http://workfunc.com/differences-between-programmers-and-coders/>.

- [11] Wikipedia contributors, *Programming paradigm*, Wikipedia, The Free Encyclopedia. Pridobljeno 26. 4. 2016 iz, https://en.wikipedia.org/wiki/Programming_paradigm.
- [12] Carl Reynolds, Paul Tymann, *Principles of Computer science*, McGraw-Hill, London, 2008.
- [13] Stoyan stefanov, Kumar Chetan Sharman, *Object-Oriented Java Script, Second edition*, Packt Publishing, Ltd, Birmingham, 2013.
- [14] Wikipedia contributors, *Java(programming language)*, Wikipedia, The Free Encyclopedia. Pridobljeno 27. 4. 2016 iz, https://simple.wikipedia.org/wiki/Java_%28programming_language%29.
- [15] Wikipedia contributors, *C++*, Wikipedia, The Free Encyclopedia. Pridobljeno 27. 4. 2016 iz, <https://en.wikipedia.org/wiki/C%2B%2B>.
- [16] Wikipedia contributors, *Python(programming language)*, Wikipedia, The Free Encyclopedia. Pridobljeno 30. 4. 2016 iz, https://en.wikipedia.org/wiki/Python_%28programming_language%29.
- [17] Zed A. Shaw, *Learn Python the Hard Way*. Pridobljeno 2. 5. 2016 iz, <http://learnpythonthehardway.org/book/>.
- [18] Norbert Jaušovec, *Kako uspešneje reševati probleme?*, Državna založba Slovenije, Ljubljana, 1991.
- [19] S.C. Ng, S.O Choy, R. Kwan, S.F. Chan, "A Web-Based Environment to Improve Teaching and Learning of Computer Programming in Distance Education", *ICWL'05 Proceedings of the 4th international conference on Advances in Web-Based Learning*, 2005
- [20] L. Ma, J. D. Ferguson, M. Roper, I.Ross, M. Wood, "A web-based learning model for improving programming students' mental models", v *Proceedings of the 9th annual conference of the subject centre for information and computer sciences*, HE Academy, 2008 pp. 88-94.
- [21] Wikipedia contributors, *Tutorial*, Wikipedia, The Free Encyclopedia. Pridobljeno 5. 6. 2016 iz, <https://en.wikipedia.org/wiki/Tutorial>.
- [22] The Python Software Foundation, *The Python tutorial*. Pridobljeno 6. 6. 2016 iz, <https://docs.python.org/3/tutorial/index.html>.
- [23] Richard E. Mayer, *Principles for multimedia learning*. Pridobljeno 6.6.2016 iz, <http://hilt.harvard.edu/blog/principles-multimedia-learning-richard-e-mayer>.

- [24] *Udemy*. Pridobljeno 6. 6. 2016 iz, <https://www.udemy.com>.
- [25] Python fiddle, *Python Cloud IDE*. Pridobljeno 6. 6. 2016 iz, <http://pythonfiddle.com/>.
- [26] *Codenvy*. Pridobljeno 6. 6. 2016 iz, <https://codenvy.com/>.
- [27] *Cloud9*. Pridobljeno 6. 6. 2016 iz, <https://c9.io/>.
- [28] *Fightcode*. Pridobljeno 6. 6. 2016 iz, <http://fightcodegame.com/>.
- [29] *w3school*. Pridobljeno 6. 6. 2016 iz, <http://www.w3schools.com/default.asp>.
- [30] *Codeschool*. Pridobljeno 6. 6. 2016 iz, <https://www.codeschool.com/>.
- [31] Wikipedia contributors, *Education in the United States*, Wikipedia, The Free Encyclopedia. Pridobljeno 5. 6. 2016 iz, https://en.wikipedia.org/wiki/Education_in_the_United_States#K.E2.80.93education.
- [32] Elliott Bristow, *Gaming in education: Gamification*. Pridobljeno 6. 6. 2016 iz, <https://www.theedublogger.com/2015/01/20/gaming-in-education-gamification/>.
- [33] *Moodle*. Pridobljeno 6. 6. 2016 iz, <https://moodle.org/>.
- [34] *Code.org Promote*, Pridobljeno 22. 6. 2016 iz, <https://code.org/promote>
- [35] *Code.org*, Pridobljeno 22. 6. 2016 iz, <https://code.org>
- [36] *Code.org About us*, Pridobljeno 22. 6. 2016 iz, <https://code.org/about>
- [37] *Code studio*, Pridobljeno 27. 6. 2016 iz, <https://studio.code.org>
- [38] *Codeacademy*, Pridobljeno 6. 9. 2016 iz, <https://www.codecademy.com>
- [39] *Scratch*, Pridobljeno 15. 9. 2016 iz, <https://scratch.mit.edu/>
- [40] *Scratch About*, Pridobljeno 15. 9. 2016 iz, <https://scratch.mit.edu/about/>
- [41] *Repl.it*, Pridobljeno 18. 9. 2016 iz, <https://repl.it/>
- [42] *Freecodecamp*, Pridobljeno 18. 9. 2016 iz, <https://www.freecodecamp.com>
- [43] *Tutorialspoint*, Pridobljeno 18. 9. 2016 iz, <http://www.tutorialspoint.com/>
- [44] *Tutorialspoint Codingground*, Pridobljeno 18. 9. 2016 iz, <http://www.tutorialspoint.com/codingground.htm>
- [45] *Thimble by mozilla*, Pridobljeno 30. 6. 2016 iz, <https://thimble.mozilla.org>

[46] *Code combat*, Pridobljeno 20. 6. 2016 iz, <https://codecombat.com>

[47] *Code combat, About*, Pridobljeno 20. 6. 2016 iz, <https://codecombat.com/about>

[48] *Codingame*, Pridobljeno 28. 6. 2016 iz, <https://www.codingame.com/home>