Pišek: Spletna zbirka nalog s samodejnim preverjanjem

Gregor Jerše

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko

gregor.jerse@fri.uni-lj.si

Katja K. Ošljak

Code Week Slovenija in
Zavod Vsak

katja@vsak.si

Matija Lokar
Univerza v Ljubljani
Fakulteta za matematiko
in fiziko

matija.lokar@fmf.uni-lj.si

**POVZETEK**

Nov slovenski portal za poučevanje osnov programiranja s pomočjo besedilnih nalog z miselnimi izzivi je namenjen osnovno- in srednješolskim učiteljem, ki želijo pri pouku ali krožkih učence spodbujati k razumevanju osnovnih programskih konceptov. Med reševanjem raznolikih nalog, razporejenih v štiri sklope, osnovno- in srednješolci spoznavajo spremenljivke, zanke, pogojne stavke, tabele in druge koncepte, s čimer Pišek predstavlja tudi most med nalogami v Mednarodnem tekmovanju iz računalniškega mišljenja Bober in Srednješolskem tekmovanju ACM iz računalništva in informatike RTK. Osnutek portala je nastal leta 2018, večino nalog pa so 2019 ustvarili študentje v sklopu projekta Naloge za poučevanje in učenje računalniškega mišljenja – Portal Pišek (NPUR), pri katerem so sodelovali Fakulteta za matematiko in fiziko Univerze v Ljubljani, Kreativni center Poligon ter Code Week Slovenija.

**Ključne besede**

poučevanje, osnovna šola, programiranje, programski koncepti, slikovni programski jezik, Blockly

**ABSTRACT**

The new website 'Pišek' for teaching the basics of computer programming uses text assignments with different challenges for elementary and high school children. The Pišek website is aiming for teachers who want to encourage students to learn the basic concepts of computer programming. While students are solving various tasks from the four available sections, they learn about variables, loops, conditional statements, tables, and other computing concepts. That also makes Pišek a bridge between assignments in the International Algorithmic Thinking Competition Bober and the ACM High School Computer Science and Informatics Competition RTK. The website was first developed in 2018. However, most of the learning assignments created students within the 2019 project 'Tasks for Teaching and Learning of Computational Thinking - Portal Pišek (NPUR)'. The project partners were the Faculty of Mathematics and Physics from the University of Ljubljana, Poligon Creative Center and Code Week Slovenia.

**Keywords**

teaching, elementary school, computer programming, programming concepts, visual programming languages, Blockly

# UVOD

Pomembna kompetenca v informacijski družbi je algoritmični način razmišljanja. Tega uspešno razvijamo s poučevanjem različnih računalniških in predvsem programskih konceptov. Osnovno znanje katerega od programskih jezikov namreč pomembno pripomore pri opismenjevanju sodobnika. Prav zato je koristno učence seznanjati tudi z osnovami programiranja. Drugi cilj, vendar mogoče še pomembnejši [17], je mlajše učence navdušiti za računalništvo in informatiko in jim pokazati, da je učenje na tem področju lahko tudi ustvarjalno ter zabavno. Žal je učenje programiranje za določen del populacije težko dostopno: v šoli se poučuje v manjšem obsegu, izvenšolske iniciative pa prav tako dosegajo le manjši del populacije [13]. Samoučenje na drugi strani pa je zahtevno, saj pri njem učencu običajno manjka povratna informacija. Ta je pri začetnih korakih v programiranju neprecenljiva, vendar pa to zahteva bodisi pravočasno udeležbo učitelja, ki jo je pri prenapolnjenih učilnicah težko zagotoviti, bodisi uporabo sistemov za samodejno preverjanje pravilnosti programov. Poleg tega tudi kritično primanjkuje kakovostnih nalog v slovenskem jeziku.

Ker je programiranje veščina, se je učenci lahko naučijo le z veliko vaje. O tem pričajo številne raziskave, med drugim [18]. A pri tem moramo upoštevati [6], da ni pomembno le samo pisanje kode, ampak tudi zgoraj omenjena povratna informacija o tem, ali napisani programi izpolnjujejo zahteve zastavljenega problema.

Večina začetniških napak je namreč preprosto rešljivih, če učence pravočasno usmerimo v pravo smer. Pri tem so dobre in sprotne povratne informacije bistvene za hiter napredek. Če pa učenci povratne informacije ne dobijo že med samim reševanjem nalog, lahko to lahko močno upočasni njihov napredek učencev.

Pri poučevanju programiranja je prav tako pomembna izbira okolja, v katerem se začetnik prvič sreča s programiranjem. V zadnjem času vrsta strokovnjakov proučuje, ali niso za začetnike morda najprimernejši slikovni programski jeziki, ki uporabniku omogočajo ustvarjanje programov s sestavljanjem vnaprej pripravljenih blokov oziroma kock programskega jezika. S tem se izognejo sintaktičnim napakam, kar učenje programiranja najbolj olajša začetnikom. Ti se lahko namesto s sintakso, tj. s pravilnostjo zapisa programa, ukvarjajo s postopkom, ki jih bo privedel do pravilne rešitve. Glavna prednost, ki jo torej prinaša uporaba slikovnih jezikov, je zmanjšana raven kognitivne obremenitve [15], ki ji je izpostavljen učenec med programiranjem. Zato je več lahko posveti reševanju problema.

Med učitelji poteka živahna debata o ustrezni izbiri prvega programskega jezika. Pogosto je glede uporabe slikovnih programskih jezikov slišati pomisleke, da je učenje programiranja na ta način preveč podobno igranju igric in ne nudi ustreznega uvoda v "pravo" programiranje, kjer se praviloma uporabljajo tekstovni programski jeziki. A obstaja več študij [5], ki kažejo, da je znanje slikovnih programskih jezikov prenosljivo na tekstovno usmerjene jezike. Nekateri raziskovalci opozarjajo na preprostost, s katero učenci prenesejo znanje (npr. glede spremenljivk, iteracijskih struktur in pogojev) iz slikovnih jezikov (kot so Blockly, Alice, Scratch in drugi) v tekstovne jezike (npr. Java ali Python). Weintrop v svoji disertaciji [18] poroča, da se učenci učijo več in hitreje, ko uporabljajo slikovne jezike, kot takrat, ko uporabljajo tekstovne.

Kot smo omenili, je večina začetniških napak dokaj preprosto rešljivih, če učence s pomočjo ustreznih informacij pravočasno usmerimo. Pri preverjanju pravilnosti rešitev učitelju lahko priskočijo na pomoč sistemi za samodejno preverjanje programskih rešitev. Ti učitelju pomagajo pri zagotavljanju hitrega odziva. Tako se učitelj lažje sooči z večjimi skupinami učencev in svojo pozornost usmeri k tistim učencem, ki jo najbolj potrebujejo.

Vendar je priprava tovrstnih sistemov za slikovne programske jezike precej težja naloga kot pri tekstovno usmerjenih jezikih. O tem priča že dejstvo, da obstajajo številni sistemi, ki podpirajo klasične jezike, kot so Java, C, Python in drugi (enega od pregledov obstoječih sistemov za samodejno preverjanje pravilnosti programov najdete npr. v [6]), za slikovne jezike pa je teh orodij bistveno manj.

Na Fakulteti za matematiko in fiziko Univerze v Ljubljani smo že pred časom razvili sistem Projekt Tomo (<https://www.projekt-tomo.si/>), ki podpira programske jezike Python, MATLAB in R [12].

Med tistimi redkimi sistemi za preverjanje pravilnosti programskih rešitev, ki podpirajo slikovne jezike, izstopa francoski sistem Algorea ([http://www.france-ioi.org](http://www.france-ioi.org/)). V sodelovanju z njimi smo ga priredili za uporabo tudi v našem jeziku, poimenovali Pišek in ga postavili na spletni naslov <http://pišek.acm.si/>.

S tem smo želeli učiteljem v slovenskih šolah dati na voljo učna gradiva in pripomočke, ki bi jim omogočala, da se bodo po lastni presoji odločali, ali bi pri poučevanju uporabljali slikovne ali tekstovne programske jezike. Menimo namreč, da učni položaj v razredu najbolje oceni vsak učitelj sam. V skladu s svojo strokovno presojo se odloči o primernem pristopu. Seveda pa mora v ta namen imeti ustrezna sredstva (predvsem učna gradiva in pripomočke).

# NAMEN PROJEKTA

Namen projekta je bil ustvariti portal za poučevanje osnovnih programskih konceptov, ki bo dovolj preprost, da ga bodo učencem predstavili učitelji – tudi tisti, ki se s poučevanjem tovrstnih vsebin še niso ukvarjali. Hkrati mora biti zbirka nalog na portalu dovolj raznolika, da bo nagovarjala učence in učenke različnih starosti in z različnimi predznanji. Posebno pozornost smo namenili tudi sestavljanju nalog, ki nagovarjajo dekleta, torej tisto polovico šolajoče se populacije, ki jo programi poučevanja računalništva prevečkrat spregledajo [13].

## Vzpostavitev odprte zbirke nalog

Portal Pišek je nastal v okviru projekta Naloge za poučevanje in učenje računalniškega mišljenja. Financiranje prek Javnega razpisa Projektno delo z negospodarskim in neprofitnim sektorjem – Študentski inovativni projekti za družbeno korist nam je omogočilo, da smo k sodelovanju povabili študente, bodisi pedagoških znanosti, računalništva, matematike, medijskih študij idr. smeri, ki so ustvarili večino nalog na portalu. Z združevanjem različnih strokovnih področij in ob podpori mentorjev z Univerze v Ljubljani, Fakultete za matematiko in fiziko in Fakultete za računalništvo in informatiko, Kreativnega center Poligon ter iniciative Code Week Slovenija, so študentje objavili 250 avtorskih nalog, 69 pa so jih prilagodili ter prenesli iz učbenika Slikovno programiranje [1].

### Pomoč učiteljem

Vzpostavljen portal je namenjen uporabi v slovenskem šolskem prostoru kot pomoč pri poučevanju programiranja. Ker je javno dostopen in brezplačen, omogoča, da se bodo učenja programiranja na zanimiv način lotili tudi drugi – na primer organizatorji izvenšolskih aktivnosti na področu digitalne vzgoje.

Z odprto in brezplačno zbirko nalog za urjenje računalniškega mišljenja so slovenske šole dobile orodje za poučevanje računalniškega mišljenja, ki bo (tako si želimo avtorji portala Pišek) pomembno prispevalo k dvigu digitalne pismenosti bodočih generacij osnovno- in srednješolcev in njihovih učiteljev.

Z nalogami, ki od učitelja ne zahtevajo posebnih znanj s področja računalništva ali programiranja, prispevamo k preseganju digitalne ločnice, zaradi katere učenci kot *digitalni nomadi* in učitelji kot *digitalni imigranti* [3] v digitalnem svetu stojijo na ločenih bregovih. Otroci so v ta svet rojeni, zaradi česar se jih odrasli – učitelji in starši, ki vstopajo v novi svet kot priseljenci - nemalokrat bojijo oziroma se ogibajo interakcij z otroki, ki bi lahko razkrile, da to področje slabše obvladajo. Učitelji pa se morajo »jeziku domorodcev« prilagoditi, saj je to edini način, da bodo učencem pomagali pri pridobivanju digitalnih veščin ter kritični oziroma odgovorni rabi tehnologije. [16]

Portal Pišek s priročnostjo vsebin in nalog zasleduje prav to, da bo v prvi vrsti opogumil učitelje in dvignil njihov interes za poučevanje računalniškega mišljenja ter drugih digitalnih kompetenc. Te so še posebej pomembne za opolnomočenje današnjih otrok in prihajajočih generacij – za njihovo participacijo, raziskovanje in ustvarjanje v informacijski družbi.

### Spodbujanje deklet

Posebno pozornost pri pripravi nalog smo posvetili dekletom, saj je to skupina z manj interesa za učenje programiranja. Poleg tega raziskave kažejo, da se interes za učenje STEM pri dekletih z leti oziroma odraščanjem še zmanjšuje, kar se odraža tudi v manjšem deležu deklet, ki se odloča za študij na naravoslovnih ali tehniških smereh. Microsoftova študija iz leta 2018 v zvezi s tem ugotavlja, da so med razlogi za manjši interes deklet za naravoslovne in tehnične poklice tudi pomanjkanje podpore in spodbude s strani staršev ter učiteljev. Kot enega od ukrepov za dvig interesa deklet za to področje pa študija predlaga navduševanje deklet s pomočjo vključujočega učnega okolja [2]. To idejo o vključujočem okolju za dekleta smo prenesli tudi na Portal Pišek. Zato smo pripravili in objavili posebne sklope nalog za učenje programiranja, ki so zaradi izbire junakov/junakinj ali zgodbe, v katero je umeščen programski izziv, portal naredili zanimiv tudi za dekleta. Tako smo na primer v pilotski raziskavi med uporabniki portala v razredu učence in učenke spraševali, katere teme so jim blizu oziroma naloge s katerega področja bi si želeli reševati in tako prišli do ideje za sklop nalog Ples, v katerem se po mreži gibljeta plesalec in plesalka. Z mislijo na dekleta smo objavili tudi sklopa nalog Gasilka in Gosenica Eva.

### Podpora izvenšolskim iniciativam

Portal Pišek je brezplačno dostopen brez registracije, saj si avtorji želimo, da ga bo uporabljalo čimveč učiteljev in mentorjev ter bo na ta način dostop do nalog imelo čimveč otrok. S tem namenom smo zasnovali poseben sklop Code Week, ki je namenjen podpori istoimenske iniciative, ki vsako leto v sklopu vseevropskega dogodka organizira tudi Slovenski teden programiranja. Pod naslovom Code Week so izbrane naloge združene glede na starost učencev in zahtevnost. Poleg tega s poglavjem Za pokušino učitelje nagovarjamo, naj v času Tedna programiranja v vsakem razredu vsaj del šolske ure namenijo programiranju, tako da učence usmerijo k reševanju teh izbranih nalog.

## Sodelovanje s študenti

Projekt je omogočal vključitev desetih študentov s petih fakultet[[1]](#footnote-1) iz različnih področij (matematika, računalništvo, družboslovje in pedagogika), ki so delovali v interdisciplinarni ekipi. Na ta način smo omogočili pripravo vsebinsko, strokovno, pedagoško in jezikovno ustreznih nalog.

# UPORABA PIŠKA

Na vstopni strani portala boste našli štiri sklope nalog – Zgodbe, Programski koncepti, Učbenik in Code Week. Ti so podrobneje predstavljeni v poglavju 3.3 Štirje sklopi nalog.

Po izbiri sklopa se učenec skozi sistem map premika do željene naloge.

## Reševanje naloge

Ob kliku na nalogo se prikaže stran z besedilom naloge, slika situacije ter delovni prostor, kjer gradimo svoj program. Kot je razvidno iz primera (Slika 1), je naloga lahko sestavljena iz več težavnostnih stopenj, ki jih označujejo zvezdice. Ko učenec uspešno reši nalogo, napreduje na naslednjo, zahtevnejšo stopnjo, na kateri razumevanje problema utrdi.



Slika 1. Primer naloge

### Samodejno preverjanje pravilnosti

Ko učenec na delovni površini bloke z ukazi poveže med seboj in zaključi program, ga odda v ocenjevanje. Samodejni ocenjevalnik preveri, ali je uporabnik nalogo uspešno rešil in uporabniku sporoči povratno informacijo.

### Pravilna rešitev

Kadar učenec nalogo reši uspešno, mu sistem to jasno sporoči (Slika 2) in mu predlaga napredovanje na zahtevnejšo stopnjo izziva ali naslednjo nalogo.



Slika 2. Sporočilo ob pravilni rešitvi

### Naloži/shrani rešitev

Pri določenih nalogah, predvsem tistih bolj zapletenih, ima učenec možnost, da rešitev shrani na svoj računalnik oziroma jo od tam naloži (glej rumeno pobarvana gumba Naloži/Shrani).

 

Slika 3. Možnost shranjevanja in nalaganja kode

### Iskanje hroščev

V nekaterih primerih je pod sliko tudi nekaj gumbov, ki učencu pomagajo pri iskanju hroščev v programu. Omogočajo izvajanje programa po korakih oz. z različnimi hitrostmi, kar močno olajša iskanje napak v programih.



Slika 4. Iskanje hroščev

### Napačna rešitev

Če je oddana naloga rešena napačno, bo portal to učencu sporočil. Možno je, da program uspešno prestane nekaj testov, vendar ne vseh, kar je pogoj za uspešno rešitev naloge (Slika 5. Napačna rešitev).

Slika 5. Napačna rešitev

### Preveč porabljenih blokov oziroma kock

Prav tako je možno, da bi program sicer pravilno rešil zastavljeno nalogo, a je učenec uporabil preveliko število blokov oziroma kock (Slika 6).



Slika 6. Preveč porabljenih blokov

### Ocenjevanje kakovosti rešitev

Ocenjevalni sistem lahko preverja ne le pravilnost ampak tudi kakovost uporabnikove rešitve. Takšna naloga je na primer »Pišek in škatle«, kjer kakovostnejša rešitev za dosego cilja porabi kar najmanj korakov, čeprav je končni rezultat enak. Na tak način uporabnik dobi hitro povratno informacijo, ki je pri začetnih korakih programiranja še kako pomembna. Tako lahko sistem Pišek uporabniki uporabljajo tudi samostojno in pri reševanju dobijo takojšen odziv glede kakovosti in pravilnosti svojih programov.

## Zgradba Piška

Kot že omenjeno, je portal Pišek nastal na osnovi francoskega sistema Algorea. Ta nam omogoča sestavljanje nalog v programskih jezikih Scratch, Blockly ali Python. Za nas je bil od teh najbolj zanimiv slikovni jezik Blockly. Namenoma smo se odločili, da pri portalu ne uporabimo jezika Scratch (ki je prav tako slikovni), ki je v naših šolah sicer bolj razširjen. Po več razgovorih s kolegi iz različnih ustanov, ki se ukvarjajo z začetnim poučevanjem programskih jezikov, Scratch ni tako primeren za reševanje nalog, ki jih želimo imeti na Pišku. Scratch je namreč primarno zasnovan kot jezik in okolje za izražanje [6].

### Navigacija

Ob obisku portala se odpre vstopna stran (Slika 7). Na voljo sta dva nivoja navigacije. Prvi nivo je predstavljen na zgornji horizontalni navigacijski vrstici ima dve možnosti: »O Pišku« in »Naloge«.

 

Slika 7. Vstopna stran in navigacija

V levem stolpcu uporabnik izbira med štirimi sklopi nalog: Zgodbe, Programski koncepti, Učbenik in Code Week, ki so razporejen v imenike z drevesno strukturo, podobno kot datoteke v datotečnem sistemu računalnika.

## Štirje sklopi nalog

Kot smo zapisali zgoraj, portal Pišek trenutno ponuja štiri sklope nalog: Zgodbe, Programski koncepti, Učbenik in Code Week, od katerih ima vsak svojo vlogo.

### Zgodbe

V Zgodbah se nahaja osem tematsko urejenih poglavij: Pišek, Robot, Ples, Tabornik, Zmajček, Gosenica Eva, Gasilka in Ladja, v katerih je zbranih prek 250 različnih nalog. Vsaka od zgodb deluje kot samostojna avantura, ki zbuja zanimanje učencev in učenk z različnimi interesi. Poleg tega učenci pri prehajanju med avanturami, torej ko zaključijo z reševanjem nalog v eni zgodbi in se nato lotijo naslednje, utrjujejo iste programske koncepte, ne da bi se pri tem dolgočasili.

### Programski koncepti

Drugi sklop je namenjen seznanjanju z oziroma utrjevanju že pridobljenega znanja o izbranih programskih konceptih (zaporedje ukazov, zanke, vhod/izhod in spremenljivke, pogojni stavki, podprogrami, tabele in nizi). S koncepti se učenci srečujejo tudi pri reševanju ostalih sklopov, vendar pa je zaradi sistematične ureditve in preglednosti ta učna pot kot nalašč za izboljšanje učnega procesa, kadar želimo z učenci poglabljati prav specifičen koncept.

### Učbenik

Te naloge temeljijo na e-učbeniku Slikovno programiranje, namenjeno uvodu v programiranje, do katerega lahko prosto dostopamo na spletu [1]. Naloge smo iz učbenika preselili na portal Pišek, ker je na Pišku reševanje nalog iz učbenika, ki sicer ponuja teoretski in praktični uvod v programiranje, uporabniku prijaznejše.

### Code Week

V četrtem sklopu so izbrane naloge razvrščene glede na starostne skupine otrok oziroma težavnost. Kot predlaga že ime sklopa, je ta v prvi vrsti namenjen izvajanju dejavnosti v okviru Slovenskega tedna programiranja – Code Week, ko učitelji ter mentorji prostovoljci iz vse Slovenije poskrbijo, da čim več otrok spozna programiranje.

## Tipi nalog na Pišku

Strokovnjaki ugotavljajo, da ima pri začetnem uvajanju programskih konceptov [4], [8], [9] branje kode zelo pomembno vlogo. Zaradi tega je učence potrebno izpostaviti nalogam, kjer spreminjajo vnaprej podane programe, jih dopolnjujejo in popravljajo. Zelo primerne so naloge, ki omogočajo zlaganje koščkov kode v ustrezen vrstni red, o čemer pišemo v poglavju 3.4.5 Parsonsov tip nalog.

### Naloge na mreži

Pri večini nalog na Pišku reševanje poteka na mreži (npr. Slika 5 ali Slika 6). Junak se po mreži pomika in pri tem izvaja določena opravila (npr. Pišek pobira zrna).

### Želvja grafika

Vrsta nalog predvideva uporabo želvje grafike, pri katerih mora učenec z ustreznimi ukazi poustvariti zahtevano sliko/vzorec. Večina teh nalog je v sklopu Programski koncepti/Želva.



Slika 8. Naloga z želvjo grafiko

### Naloge tipa »sestavi program«

Večina nalog zahteva, da reševalec sestavi ustrezen program od začetka. Pri tem ima lahko na voljo le izbrane ukaze (Slika 9), kjer so na voljo le trije osnovni ukazi za premikanje (ni pa ukazov za npr. zanko).



Slika 9. Napiši program z izbranimi ukazi

### Naloge tipa »popravi program«

Določene naloge so klasične programerske, kjer program prebere določene podatke in izpiše rezultat (Slika 10). Spodaj je primer naloge, kjer je program že dan, vendar pa naloge ne reši pravilno. Učenec mora poiskati napake in jih odpraviti.



Slika 10. Popravi program

### Parsonsov tip nalog

Pri teh nalogah imamo delčke kode že na delovni površini. Naloga učenca je, da jih zloži v pravilni vrstni red (Slika 11). Obstaja več različnih podvrst Parsonsonovega tipa nalog [17], ko je npr. delčkov kode preveč oziroma so med ustreznimi skriti tudi napačni ali pa so zloženi na napačnih mestih ipd.



Slika 11. Naloga Parsonsonovega tipa

### Druge logične naloge

Poleg naštetih se v razdelku Programski koncepti / Razno nahaja nekaj nalog, ki se ne rešujejo s povezovanjem blokov v programe. Pri teh nalogah mora uporabnik predvsem razmisliti, kako sploh priti do rešitve, ter jo potem na določen način (ne nujno s povezovanjem blokov) vpisati v računalnik. Tako morajo na primer pri nalogi Pacman ubežati duhcu, pri nalogi Zlaganje paketov pa z žerjavom premikajo škatle zložene v stolpce.

Med drugim sestavljalci nalog ta razdelek uporabljamo kot peskovnik, v katerem preizkušamo nove tipe nalog.

# NAMESTO ZAKLJUČKA

Del Portala Pišek je nastal v okviru projekta *Naloge za poučevanje in učenje računalniškega mišljenja – Portal Pišek (NPUR)*, ki je potekal v okviru javnega razpisa *Projektno delo z negospodarskim in neprofitnim sektorjem – študentski inovativni projekti za družbeno korist 2016–2020*, sofinanciranega s strani Evropskega socialnega sklada, Ministrstva za izobraževanje, znanost in šport Republike Slovenije ter Javnega štipendijskega, razvojnega, invalidskega in preživninskega sklada Republike Slovenije.

Osnovnošolski učitelji so Piška že testirali v razredih in avtorjem posredovali večinoma pozitivne odzive. Podrobneje pa je portal in naloge na njem spoznalo 22 udeležencev Počitniške šole programiranja, umetnosti in etike v organizaciji Zavoda Vsak, ki je avgusta potekala v Poligon kreativnem centru. Mentorji so nam sporočili, da so se učenci in učenke, stari 11-13 let, med reševanjem nalog poglobili in za dve uri popolnoma umolknili ...

# LITERATURA IN VIRI

1. Anželj, G., J. Brank, A. Brodnik, L. Fürst in M. Lokar. 2018. Slikovno programiranje v1.00: E-učbenik za uvod v programiranje, Blockly. Univerza v Ljubljani.
2. Ashcraft, C., E. Eger, and M. Friend, “Girls in IT: The Facts. 2018. National Center for Women & Information Technology (NCWIT).
3. Barlow, J. P. 1996. Declaration of the Independence of Cyberspace. Dostopno prek: http://editions-hache.com/essais/pdf/barlow1.pdf.
4. Busjahn, T. and C. Schulte. 2013. The use of code reading in teaching programming. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research (Koli Calling '13*). ACM, New York, NY, USA, 3-11. DOI: <https://doi.org/10.1145/2526968.2526969>
5. Chandler P. in J. Sweller. 1996. Cognitive Load While Learning to Use a Computer Program. V *Cognitive Psychology*.
6. Corbett, A. T. in J. R. Anderson. 2001. Locus of feedback control in computer-based tutoring: impact on learning rate, achievement and attitudes. V *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*, New York.
7. Ericson, B. J., Margulieux, L. E., & Rick, J. 2017. Solving parsons problems versus fixing and writing code. V *Proceedings of the 17th Koli Calling International Conference on Computing Education Research (pp. 20-29)*. ACM.
8. Ericson, B., Moore, S., Morrison, B., & Guzdial, M. 2015. Usability and usage of interactive features in an online ebook for CS teachers. V *Proceedings of the Workshop in Primary and Secondary Computing Education (pp. 111-120).* ACM.
9. Griffin, Jean M. 2016. Learning by taking apart: deconstructing code by reading, tracing, and debugging. V *Proceedings of the 17th Annual Conference on Information Technology Education.* ACM.
10. Guzdial, M. 2018. Teaching Two Programming Languages in the First CS Course. Dostopno prek (22. maj 2018): [https://cacm.acm.org/blogs/blog-cacm/228006-teaching-two-programming-languages-in-the-first-cs-course/fulltext#](https://cacm.acm.org/blogs/blog-cacm/228006-teaching-two-programming-languages-in-the-first-cs-course/fulltext)
11. Ihantola, P., Ahoniemi, T., Karavirta, V., in Seppälä, O. 2010. Review of recent systems for automatic assessment of programming assignments. V *Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling '10)*. ACM.
12. Jerše G. in M. Lokar. 2018. Uporaba sistema za avtomatsko preverjanje nalog Projekt Tomo pri učenju programiranja. V *Vzgoja in izobraževanje v informacijski družbi - VIVID 2017: zbornik referatov*, Ljubljana.
13. Koren Ošljak, K. in T. Oblak Črnič. 2018. Digitalna kultura na obronkih: vloga izvenšolskih iniciativ pri kultivaciji digitalne pismenosti. V *Revija knjižnica, letnik 62, št. 1-2,* Ljubljana.
14. Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. 2010. The scratch programming language and environment. V *ACM Transactions on Computing Education (TOCE), 10(4)*.
15. Meerbaum-Salant, O., M. Armoni in M. Ben-Ari. 2010. Learning computer science concepts with Scratch. V *Proceedings of the Sixth international workshop on Computing education research (ICER '10)*, New York.
16. Prensky, M. 2010. Teaching digital natives – partnering for real learning. A Sage company.
17. RINOS, 2018. Snovalci digitalne prihodnosti ali le uporabniki? Dostopno prek: <https://fri.uni-lj.si/sl/novice/novica/uporabniki-ali-snovalci-digitalne-prihodnosti>, Ljubljana.
18. Weintrop, D. 2016. Modality Matters: Understanding the Effects of Programming Language Representation in High School Computer Science Classrooms (Dissertation). Northwestern University.
1. Sodelujoči študentje: Luka Čušin, UL PeF, Blaž Dobravec, UL FMF, Urška Erjavec, UL PeF, Zoran Fijavž, UL FF, Žiga Flajs, UL FMF, Karel Križnar, UL FMF, Ajda Lah, UL FDV, Anže Lokar, UL FS, Aleš Poklukar, UL FMF, Dimitar Stoilkov, UL FMF, Krištof Špenko, UL FMF in Maks Valenčič, UL FDV. [↑](#footnote-ref-1)