Uporaba Sistema Pišek pri pouku neobveznega izbirnega predmeta računalništvo

Teaching elective subject Computing using system Pišek

Matija Lokar
Univerza v Ljubljani
Fakulteta za matematiko in fiziko
matija.lokar@fmf.uni-lj.si

Maja Mujkić
OŠ Koseze, Ljubljana
maja.mujkic@gmail.com

POVZETEK

V članku je v prvem delu predstavljen Sistem Pišek, spletna storitev, ki vsebuje bogato zbirko nalog s samodejnim preverjanjem pravilnosti rešitev. Naloge rešujemo s pomočjo programskega jezika Blockly, ki omogoča programiranja z delčki. V drugem delu prispevka je opisan način uporabe Piška pri poučevanju osnovnih konceptov programiranja pri neobveznem izbirnem predmetu računalništvo, ki se izvaja v drugi triadi OŠ. Pri tem je predstavljeno tako delo v razredu, kot tudi uporaba Piška v drugem delu šolskega leta 2019/2020, ko je bil pouk izvajan na daljavo.

Ključne besede

Poučevanje, osnovna šola, programiranje, programski koncepti, programiranje z delčki, delo na daljavo, Blockly

ABSTRACT

The first part of the article presents Pišek, a system with a rich collection of automatic verifiable problems. Problems are being solved in visual programming language. The second part of the article describes the usage of the system in teaching basic programming concepts in the elective subject Computing attended by pupils in the second triad of the primary school. The article presents work in classroom as well as the usage of Pišek in the second part of the school year 2019/2020 when distance learning has been used.

Keywords

Teaching, elementary school, computer programming, programming concepts, visual programming languages, distance learning and teaching, Blockly

1. UVOD

Programiranje je veščina. Od posameznika zahteva natančnost in doslednost. Lastnosti, ki nam pogosto prideta prav v življenju, vsekakor pa tudi učencem pri poljubnem predmetu v šoli. In pri tem navajanju na natančnost in doslednost lahko izrabimo lastnosti računalnika. Ta ne upošteva čustev in obrazne mimike, ne bere misli, ne predvideva, kaj smo želeli povedati, pa smo se samo nerodno izrazili in podobno. Računalnik naredi le tisto, samo tisto in točno tisto, kar smo mu »ukazali«.

V osnovni šoli je eden od ciljev pri neobveznem izbirnem predmetu računalništvo, ki se izvaja v drugi triadi, tudi ta, da znajo učenci algoritem zapisati s programom v nekem programskem jeziku. Programski jeziki, kot so Java, C, Python in podobni, so za osemletnike, ki se prvič srečujejo s programiranjem, izredno zahtevni. Poleg tega, da se morajo naučiti »programerskega« razmišljanja, morajo usvojiti še sintakso in semantiko, ki sta pri vsakem jeziku drugačni. Učenja programiranja pri najmlajših (in tudi pri starejših začetnikih) se je veliko lažje lotiti s programskimi jeziki, kjer programa ne tipkamo, ampak ga zložimo iz predpripravljenih delčkov. Na ta način se izognemo tipkarskim napakam, napačni uporabi presledkov, uporabi velikih in malih črk, manjkajočim podpičjem na koncu vrstic in drugim sintaktičnim napakam. Učenci se lahko osredotočijo le na pomembnost vrstnega reda ukazov, uporabe zank, pogojnih stavkov in podobno.

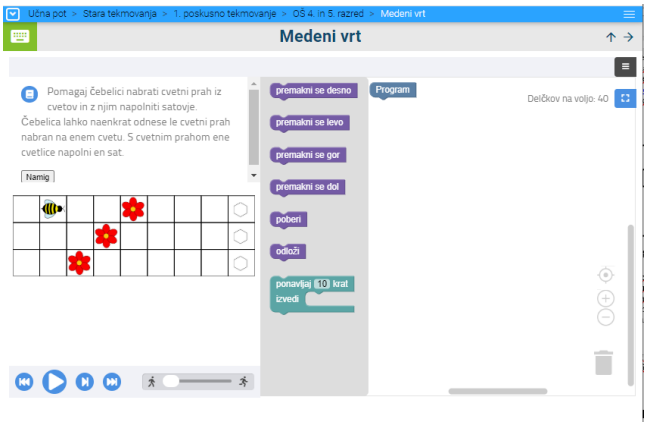
Pri učenju je eden od najpomembnejših elementov povratna informacija. Hitrejša in natančnejša je, bolj je smiselna in uporabna. A pri reševanju problemov s programiranjem je pogosto več poti do rešitve. Zato večinoma splošna in enovita povratna informacija ni možna. To pomeni, da mora učitelj pregledati vsako oddano nalogo posebej. Če ima v razredu 25 učencev, je precej nemogoče hitro pregledati vse, kar so učenci v eni uri ustvarili in jim hitro dati kvalitetno povratno informacijo. Zato so orodja, ki učitelju pomagajo pri pripravi povratnih informacij tako pomembna.

2. O SISTEMU PIŠEK

Sistem Pišek je portal, namenjen uporabi v slovenskem šolskem prostoru kot pomoč pri poučevanju programiranja. Je javno dostopen in njegova uporaba je povsem prosta.

Osnutek portala je nastal leta 2018 na osnovi francoskega sistema Algorea, ko so v okviru ŠIPK projekta ProNAL študenti pod vodstvom M. Lokarja in G. Jeršeta sestavili prvih nekaj nalog za Piška. Sam sistem pa je postal uporaben za splošno rabo, ko se je leta 2019 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, močno razširila baza nalog, ki jih lahko rešujemo na Pišku.

Nadaljnji razvoj Sistema Pišek je v letu 2019/2020 potekal v sklopu priprav na izvedbo ACM Tekmovanja v programiranju z delčki. Sama priprava na tekmovanje in izvedba dveh poskusnih tekmovanj so močno razširili obstoječo zbirko nalog. Prav tako je sistem dobil novo podobo, nov grafični vmesnik.



Slika 1. Naloga v posodobljenem grafičnem vmesniku

Podrobneje si o samem konceptu Piška in njegovem razvoju lahko preberete v prispevku [2], tukaj pa si oglejmo nekaj poudarkov:

- Na levi strani posamezne naloge je besedilo, ki opisuje problem. Desni del je namenjen sestavljanju programa. Najprej je stolpec, kjer so navedeni delčki, ki so na voljo. Od sestavljavca naloge je odvisno, ali bo navedel le delčke, ki so potrebni za rešitev, ali bo dodal še kakšne nepotrebne. Prav tako lahko te delčke razporedi v kategorije (Slika 2) ali ne (Slika 1).
- Večina nalog je tipa "naloge na mreži" (glej npr. primer na Slika 1). Vse se dogaja na praviloma pravokotni mreži, kjer se liki gibljejo po mreži in opravljajo določene akcije. Program je napisan pravilno, če je končno stanje na mreži tako, kot je zahtevano
- Zaradi uporabe jezika Blockly so delčki, ki so sicer konceptualno enaki (na primer, naredi nekaj s predmetom, ki je na polju, kjer je trenutno lik), v različnih nalogah lahko poimenovani različno (poberi plastenko, pojej deteljico). Prav tako lahko sestavljavec naloge tvori nove delčke in tako v posamezni ukaz "skrijem" določeno kompleksnost (npr. v nalogi je delček "nariši kvadratek"). Ustvarjanja funkcij, ki so za to potrebne, v jezikih, kot je npr. Scratch, ne moremo "skriti".
- Sistem omogoča zelo različne tipe nalog. Pri tem bi posebej izpostavili naloge tipa "zloži delčke v pravi program" (primer je prikazan na Slika 10). Gre za tako imenovan Parsonsov tip problemov ([4]) za katerega so raziskovalci pokazali (glej na primer [5]), da precej pripomore k lažjemu in hitrejšemu osvajanju osnovnih programskih konceptov.

Slika 2. Delčki, razporejeni v kategorije

3. UPORABA SISTEMA PIŠEK PRI POUKU

Za začetnike v učenju programiranja je ključno dvoje: sistem, ki omogoča ukvarjanje z vsebino problema in sistem, ki omogoča takojšnjo povratno informacijo. Sistem Pišek zajema oboje.

Sistem Pišek je lahko učitelju v veliko pomoč, saj v razredu omogoča diferenciacijo in individualizacijo. Vsebuje naloge čistih začetkov in naloge, ki zahtevajo kompleksnejša znanja. Ker so enake naloge pripravljene v različnih težavnostnih stopnjah, lahko učenci rešujejo le najlažje ali najtežje. Ker sistem sam javi, ali je naloga opravljena ali ne, se lahko učitelj posveti tistim učencem, ki imajo težave. Učenci lahko sami po uspešno opravljeni nalogi nadaljujejo z drugo nalogo in jim ni treba čakati, da učitelj nalogo pregleda. Ker sistem omogoča tudi prijavo z uporabniškim računom, ima učenec (in učitelj) pregled nad opravljenim delom in lahko rešuje vedno nove naloge.

3.1 Zasnova sistema

Slika 3. Učna pot

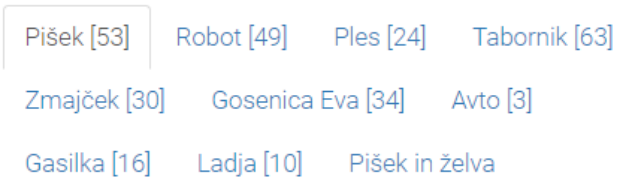
Sistem je zasnovan tako, da se lahko učitelj posveti le enemu programskemu konceptu ali pa naredi ponovitev različnih konceptov. V prvih urah učenja programiranja je pomembno, da učenci usvojijo zaporedje ukazov, kasneje to nadgradijo z zankami, pogojnimi stavki, spremenljivkami in drugimi programskimi koncepti.

Slika 4. Seznam programskih konceptov

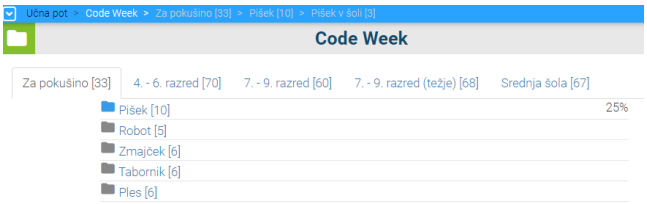
Ko imajo osnovno znanje, se lahko nalog lotijo »po zgodbah«. Te so zelo uporabne, ko želimo ponavljati različne koncepte in utrjevati znanje ali učence pripraviti na tekmovanje. Nekateri bodo raje reševali naloge z roboti, drugi naloge z gosenico, za dekleta bo morda najbolj zanimiva plesalka. V Zgodbah je deset tematsko

urejenih poglavij: Pišek, Robot, Ples, Tabornik, Zmajček, Gosenica Eva, Avto, Gasilka, Ladja ter Pišek in želva (Slika 5).

Ko učenec zaključi z eno zgodbo, nadaljuje z drugo in se pri tem kljub utrjevanju istih programskih konceptov ne dolgočasi ([2]).



Slika 5. Učenci si lahko sami izberejo junaka, s katerim se bodo učili



Slika 6. Sklop 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.

3.2 Usmerjanje učencev pri ustrezni izbiri nalog

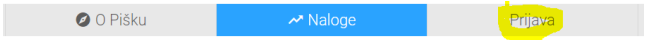
Predvsem ob ponavljanju konceptov lahko pustimo učencem proste roke. Sistem ima naloge zložene tudi po zgodbah, v katerih nastopajo različni junaki in vsak učenec gre lahko v svoj domišljijski svet. Pri tem bo občutek, da igra igrico, še večji, čeprav bo sestavljal program in utrjeval programersko razmišljanje. Junakov je na voljo dovolj, deset. V primeru, da učenci izrazijo željo po novem junaku, lahko stopimo v stik z ustvarjalci sistema in predlagamo še kakšnega.

3.3 Povratna informacija za učenca in učitelja

Zaradi takojšnje povratne informacije, bodisi pravilne (Slika 14), bodisi napačne (Slika 13), lahko učenec samostojno rešuje naloge v svojem lastnem tempu. Učitelj mu pomaga z usmeritvami. Če gre učencu težje, lahko rešuje samo naloge na prvi stopnji, če mu gre zelo dobro, jih lahko rešuje na vseh ali pa samo na zadnji.

Sistem poleg pravilnosti rešitev pregleduje tudi optimizacijo programa (z omejitvijo števila delčkov, ki jih sme učenec uporabiti, je prisiljen v uporabo zank, senzorjev, pogojnih stavkov). S tem učitelju ni treba pregledati vsakega programa posebej, ampak se lahko posveti tistim posameznikom, ki imajo težave oz. potrebujejo dodatno pomoč.

Sistem omogoča tudi prijavo in s tem shranjevanje dosežkov. Tako lahko vsak učenec nadaljuje tam, kjer je ostal, učitelj pa ima s tem pregled opravljenih nalog za vsakega posameznega učenca.



Slika 7. Sistem omogoča prijavo

Slika 8. Pregled uspešnosti reševanja

Ob seznamu nalog (**Error! Reference source not found.**) so različne oznake. Hitro je vidno, ali je naloga v celoti rešena ali delno, vidi se tudi, ali se je učenec naloge že lotil ali pa sploh še ne.

4. Uporaba Piška na OŠ Koseze

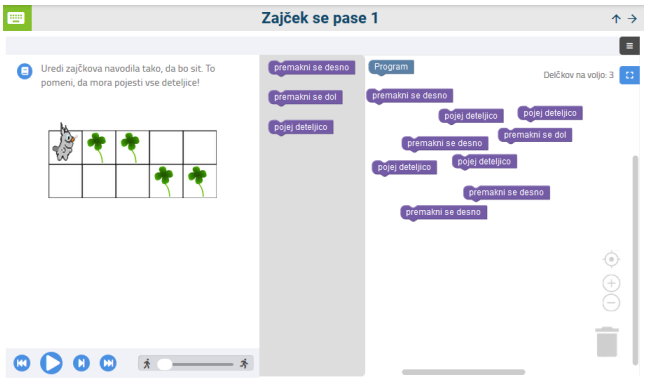
Na OŠ Koseze smo Sistem Pišek začeli uporabljati ob koncu šolskega leta 2018/2019, pri učencih 4. in 5. razreda predvsem kot utrjevanje že osvojenih konceptov, pri učencih od 7.-9. razreda pa kot uvod v programiranje.

V šolskem letu 2019/2020 so učenci 4. razreda s pomočjo Sistema Pišek usvojili koncepta zaporedje ukazov in zanke ponavljaj. Prvo uro, ko so se srečali s sistemom Pišek, smo si pogledali, kako je stran sestavljena, kako se delčki sestavljajo, da so nekje delčki »skriti« v kategorije ipd. Opozorjeni so bili, da morajo natančno prebrati navodilo naloge in da morajo biti pozorni, na kakšen način se junaki premikajo (Slika 9). Učenci so na spletni strani http://pisek.acm.si/ izbrali Programski koncepti (Slika 3) in Zaporedje ukazov (Slika 4) in reševali naloge. Najprej so morali rešiti prve tri (zato, da so začeli delati in da niso samo pregledovali nalog), potem pa so lahko izbrali katero koli nalogo na seznamu. Malo pred koncem ure so pokazali, kako uspešni so bili, torej, koliko nalog jim je uspelo rešiti (sistem tudi brez prijave beleži uspešnost reševanja kot je prikazano na sliki Slika 8 dokler ne zapremo okna brskalnika). Ker so bili učenci v sistem prijavljeni, so se njihovi rezultati shranili in so lahko naslednjo uro nadaljevali z nalogami, ki jih še niso rešili. Podobno smo naredili pri programskem konceptu Zanke. Učenci 5. in 6. razreda pa so ta dva koncepta ponovili in svoje znanje nadgradili s pogojnim stavkom in senzorji. Zato so izbrali Zgodbe (Slika 3) in si izbrali svojega junaka (Slika 5), naloge pa so morali reševati po vrsti, saj so na seznamu praviloma razporejene po težavnosti. Ko so prišli do nalog, ki jih še niso znali rešiti, so si izbrali drugega junaka. Ker so bili v sistem prijavljeni, so videli, katere naloge so že reševali v prejšnjem šolskem letu in so jih lahko spustili.

Ker se je na začetku tako pri četrtošolcih, ki so čisti začetniki, kot pri petošolcih in šestošolcih, ki so se s temi nalogami že srečali, izkazalo, da imajo težave z razumevanjem načina premikanja junakov, je smiselno, da pred začetkom uporabe sistema oziroma posamezno pri vsaki od prvih nekaj nalog, učence večkrat opozorimo na način premikanja po mreži, na kateri je zastavljena večina nalog. Na sliki Slika 9 sta prikazana dva sklopa ukazov za premikanje – uporabljamo pa lahko tudi druge, npr. premike glede na smeri neba. Z učenci lahko naredimo tudi vajo »v živo«, tako da se učitelj ali učenec premikata po navodilih kot so na delčkih. Tako si najlažje predstavljajo razliko med »obrni se desno« in »premakni se desno«.

Slika 9. Delčki z ukazi za premikanje

Predno se učenci lotijo samostojnega sestavljanja programov, je smiselno, da najprej rešijo naloge, kjer so vsi ukazi že podani, vendar »razmetani« in jih morajo pravilno zložiti (Slika 10). Ker gre za programiranje z delčki, sintaktične napake niso možne. Učenci pri teh nalogah pogosto zbrišejo že podane ukaze in se programa lotijo »od začetka«. Še posebej se to pojavlja pri učencih, ki se prej niso srečali s takim tipom nalog, zato v prvih urah te naloge zahtevajo malo več usmerjanja s strani učitelja.



Slika 10. Primer Parsonsovega tipa naloge

Podobno je pri nalogah tipa popravi napako. Učenec mora ugotoviti, kaj je narobe (npr. lahko manjka ukaz, lahko je kakšno število v zanki ali funkciji preveliko). Ker imamo delčke, se res ukvarjamo samo z vsebinskimi napakami.

Pri vsem tem je treba omeniti, da so učenci ob uri, ko smo uporabljali sistem Pišek, pogosto rekli, da igrajo igrico. Opažamo, da jeziki, ki podpirajo programiranje z delčki, pripomorejo k lažjemu razvijanju računalniškega mišljenja, sistemi s povratnimi informacijami pa učitelju olajšajo delo.

Kot smo že omenili, jezik Blockly omogoča avtorjem, da dodajajo delčke s čisto novimi ukazi kot npr. "poberi lešnik", "naberi med", "izvedi pirueto". S tem se ognemo uporabi funkcij, ki so v tem starostnem obdobju še preveč abstraktne. Po drugi strani pa smo opazili, da to pripomore tudi k navajanju na natančnejše branje navodil. Tako je v nekaterih nalogah dovolj, da junak pride na mesto s predmetom in ga s tem samodejno pobere, v drugih pa mora za to uporabiti ukaz.

Sistem se je izkazal za izredno koristnega tudi v času šolanja na daljavo, saj so učenci lahko reševali naloge, primerne svojemu predznanju, in pri tem takoj dobili povratne informacije. Učitelju so poslali zaslonsko sliko uspešnosti (**Error! Reference source not found.**), da je lahko spremljal njihov napredek. Če kakšne naloge niso znali rešiti, so poslali povezavo do naloge in zaslonsko sliko njihove rešitve, učitelj pa jih je usmeril k pravilni rešitvi.

Učenci od 4. do 9. razreda, ki obiskujejo neobvezne oz. obvezne izbirne predmete računalništva ali robotiko z elektroniko in elektrotehniko z robotiko, so se v šolskem letu 2019/2020 udeležili dveh poskusnih tekmovanj v programiranju z delčki Pišek. Tekmovanje je zastavljeno tako, da se lahko izvede v času pouka. Ima več kratkih nalog z različnimi programskimi koncepti in različnih težavnosti, tako da lahko učenci (in mentorji) dobijo celostno povratno informacijo o svojem znanju. Ker ima tekmovanje različne kategorije, je primerno za vse starostne stopnje ter za različna predznanja. Mentor ima tukaj pomembno vlogo, da tekmovalca pravilno usmeri v ustrezno izbiro kategorije. Ker je bilo drugo poskusno tekmovanje ravno v času šolanja na daljavo, je bilo vzpodbujanje in usmerjanje učencev toliko težje. Pojavilo se je tudi nekaj osnovnih težav, ker učenci niso dobro sledili navodilom (termin tekmovanja, prijavni podatki, dostop do nalog), kar pa nam je dalo dodaten zagon in nove ideje za načrtovanje pouka za naslednje šolsko leto.

Več o samem tekmovanju si lahko preberete v [3].

4.1 Primer poučevanja zanke

V ilustracijo konkretne uporabe sistema pri pouku, si poglejmo, kako smo razmišljali pri pripravi na poučevanje koncepta zanke.

Program je zaporedje ukazov. To učenci hitro usvojijo. Prav tako hitro spoznajo, da je za uspešen program pomemben vrstni red ukazov.

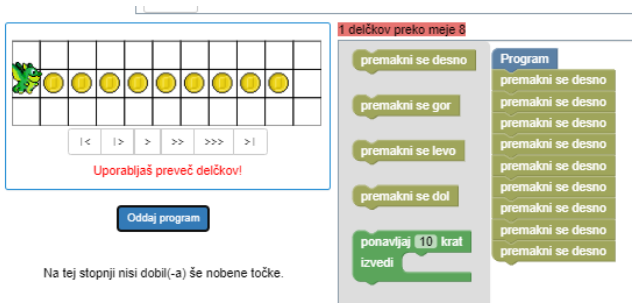
Prve težave se pojavijo, ko jih želimo naučiti program optimizirati, torej da namesto štirih enakih ukazov uporabijo zanko »ponovi štirikrat«. Program bo seveda deloval pravilno v obeh primerih, želimo pa si, da bi sistem sam preveril, ali zna učenec uporabljati zanke.

Slika 11. Primer osnovne optimizacije

Učenci sicer razumejo, da je učinek zgornjih dveh delov programa popolnoma enak. Kljub temu pa se večina, ko pišejo oz. sestavljajo program, ne spomni, da bi sami od sebe uporabili zanko. Pred Piškom smo delali v Scratchu in ker je program deloval ne glede na to, ali so uporabili zanko ali ne, jih veliko tega koncepta ni usvojilo. V sistemu Pišek jih lahko k temu "prisilimo" tako, da sestavimo nalogo, ki ima omejitev števila delčkov, ki jih lahko uporabijo pri sestavljanju programa.

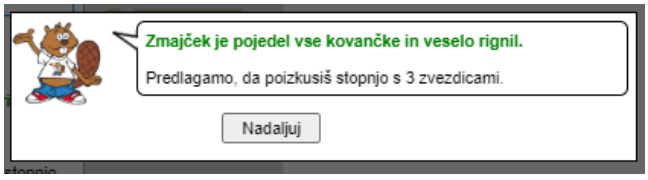
Slika 12. Primer naloge z omejitvijo števila delčkov

Učenec sicer lahko začne z zlaganjem več enakih delčkov, vendar bo kmalu ugotovil, da se mora naloge lotiti drugače (Slika 13). Sistem tako omogoča, da je edina prava rešitev, ko učenec uporabi zanko.



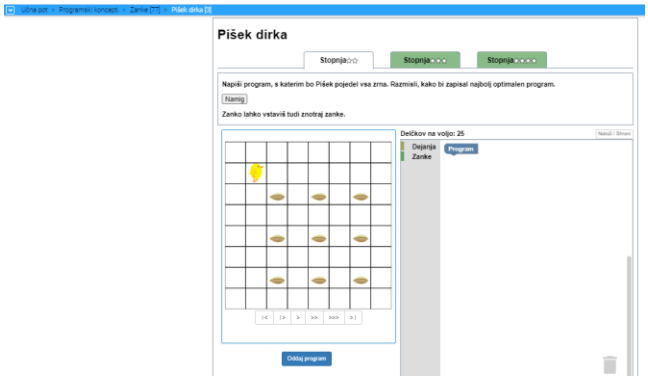
Slika 13. Primer reševanja brez uporabe zanke

Ko učenec odda program s pravilno rešitvijo in z upoštevanimi omejitvami, dobi povratno informacijo z usmeritvijo za reševanje naslednjih nalog.



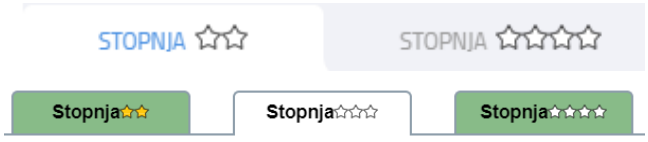
Slika 14. Povratna informacija ob pravilni rešitvi

Ko se učenci naučijo osnovne uporabe zanke ponavljaj, jo nadgradimo z »zanko v zanki«.



Slika 15. Primer naloge z ugnezdenimi zankami

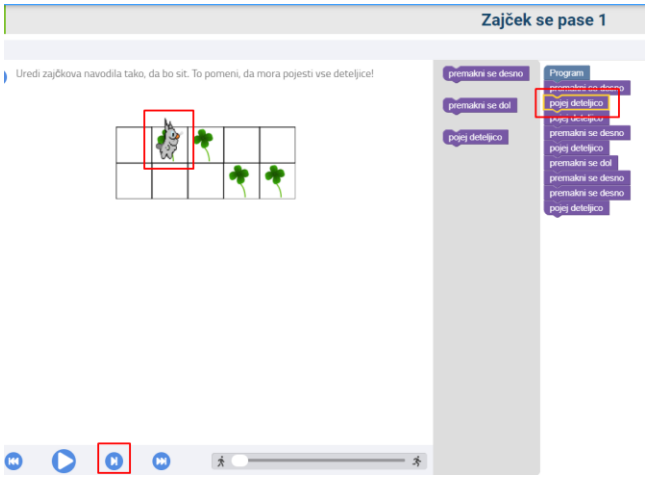
Večina nalog je sestavljena na treh različnih stopnjah. Različne stopnje preverjajo poznavanje istih programskih konceptov, razlika je največkrat v dolžini programa. Ko učenec reši stopnjo, se zvezdice obarvajo rumeno (Slika 16), kar omogoča učitelju, da hitro vidi, kako učenec napreduje (da ni kakšne naloge oz. stopnje preskočil).



Slika 16. Oznake stopenj

4.2 Preverjanje pravilnosti programa po korakih

Sistem omogoča tudi, da učenci svoj program preverjajo po korakih, vsak ukaz oz. delček posebej. Tako naj bi hitreje našli napako v svojem programu. Na naši šoli se je pokazalo, da je v splošnem za učence osnovne šole to precej težko in si s tem ne znajo pomagati. Pri iskanju napak po korakih potrebujejo veliko vodenja in usmerjanja učitelja. Je pa koristna možnost, da je pri izvajanju po korakih na mreži sproti pokazan učinek, v samem programu pa označen ukaz, ki se bo izvedel naslednji (glej označene dele na Slika 17).



Slika 17. Izvajanje po korakih

5. ZAKLJUČEK

Pri neobveznem izbirnem predmetu računalništvo, ki se izvaja v drugi triadi OŠ, se srečujemo z nemalo težavami. Že to, da imamo lahko skupino 28-tih učencev različne starosti in s popolnoma različnim predznanjem, saj so nekateri pri predmetu prvo leto, nekateri pa že tretje, zahteva od učitelja veliko inovativnosti pri načrtovanju pouka. Kako narediti predmet zanimiv in koristen in ne prezahteven, ko pa so si učenci tako različni? In to predmet, ki je po eni strani izbirni, po drugi strani pa uči veščine, ki so nujno potrebne v vsakdanjem življenju.

V takih situacijah nam je lahko IKT v veliko pomoč, saj nam res omogoča nujno potrebno individualizacijo in diferenciacijo. S sistemom za avtomatsko preverjanje pravilnosti rešitve pa učitelju omogoča, da lažje prepozna učence s težavami in mu zagotovi čas, ki ga potrebuje za pomoč in usmerjanje.

Uporaba jezikov, ki omogočajo programiranje z delčki, samemu programiranju seveda ni na neki čarobni način odvzelo zahtevnosti. Vendar opažamo, da je učencem blizu in jih spominja na igranje igric. In učenje skozi igro je tisto, ki je pri otrocih najpomembnejše in gradi osnove za druge, višje oblike učenja in razvoj mišljenja.

Zato so orodja, kot je Sistem Pišek, pomembna. Lahko jih uporabljamo zgolj kot vir idej za naloge in še vedno uporabljamo svoj izbrani programski jezik (na primer Scratch) in/ali svoje okolje (na primer MakeCode). Lahko pa Sistem Pišek uporabljamo kot celoto in izkoristimo možnost samodejnega preverjanja pravilnosti rešitev.

6. 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] Jerše, G., K. Koren Ošljak, M. Lokar. 2019. Poučevanje programskih konceptov: Spletna zbirka nalog s samodejnim preverjanjem. V Informacijska družba, Zbornik 22. mednarodne multikonference – IS 2019.

[3] M. Lokar. M. Mujkić. 2020. ACM Tekmovanja – Pišek: Tekmovanje v programiranju z delčki, V Informacijska družba, Zbornik 23. mednarodne multikonference – IS 2020

[4] Parsons, D., Haden, P.: Parson's programming puzzles: a fun and effective learn-ing tool for first programming courses. V Proceedings of the 8th AustralasianConference on Computing Education-Volume 52. str. 157–163 (2006)

[5] 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.