

UKLJUČIVANJE ELEMENATA MATEMATIKE I LOGIKE U NASTAVU
PROGRAMIRANJAMarko Tot¹, Olivera Hrnjaković², Aleksandar Kupusinac³, Rade Doroslovački⁴^{1,2,3,4}Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, Srbija¹markotot@uns.ac.rs, ²olja.hrnjakovic@uns.ac.rs, ³sasak@uns.ac.rs, ⁴rade.doroslovacki@uns.ac.rs

Kratak sadržaj: Učenje programiranja, kao i učenje bilo koje nove veštine, iziskuje napor, posvećenost, strpljenje i vežbu. Poseban izazov u učenju programiranja predstavlja činjenica da je ono spoj više različitih disciplina. Programiranje upotrebljava koncepte apstraktnih nauka, poput matematike i logike i kombinuje ih sa eksperimentalnim metodama radi rešavanja problema iz realnog sveta. Elementi matematike se pojavljuju na svakom koraku u kursovima programiranja i uglavnom ostanu nedovoljno razrađeni ili objašnjeni samo na minimalnom nivou. Već na samom početku učenja programiranja, javljaju se pojmovi koji zahtevaju matematičko predznanje, ili su bazirani na logičkim konceptima. Zbog toga se često postavlja pitanje da li je znanje iz matematike i logike korelirano sa znanjem iz programiranja. Detekcija i analiza pomenutih korelacija može biti značajna jer može doprineti unapređenju nastave iz programiranja i potencijalno matematike. Cilj uvođenja matematičkih i logičkih elemenata u programiranje jeste da razjasni učenicima kako ove čvrsto povezane oblasti čine celinu i kako učenje jednog olakšava i doprinosi učenju drugog. Ovaj rad opisuje specifičnosti nastavnog procesa koje postoje prilikom podučavanja programiranja, analizira vezu između matematike i programiranja i ističe značaj uključivanja matematičkih koncepata u nastavu programiranja.

Ključne reči: Nastava, Matematika, Logika, Programiranje

INCLUDING MATHEMATICAL AND LOGICAL CONCEPTS IN PROGRAMMING
COURSES

Abstract: Learning programming, just like learning any new skill requires effort, dedication, patience and practice. A particular challenge in learning programming is the fact that it is a composition of several different disciplines. Programming encompasses the concepts of abstract sciences like mathematics and logic and combines them with experimental methods to solve real world problems. Elements of mathematics appear at almost every step in programming courses and are usually under-elaborated or are explained at a most basic level. Programming terms that require mathematical knowledge and logic concepts appear at the very beginning of programming courses. Therefore, the question "Is mathematical and logic knowledge correlated with programming knowledge?" often arises. The detection of such a correlation and its analysis could contribute to improving programming lessons as well as mathematics ones. The goal of introducing mathematical and logic elements in programming is to clarify to the students how these tightly knit areas of science are built on each other and to show them how learning one eases the learning of other. The paper describes the methods of teaching process while teaching programming, to analyse the link between mathematics and programming, as well as to highlight the importance of incorporating mathematical concepts into programming courses.

Key Words: Education, Mathematics, Logic, Programming

1. UVOD

Podučavanje programiranja donosi sa sobom veliki broj specifičnosti iz razloga što je ono apstraktna oblast te često može biti teško za razumevanje početnicima. Zbog toga je važno da se stalno radi na unapređenju metoda izvođenja nastave iz programiranja kada je reč o srednjoškolskom obrazovanju, ali i visokoškolskim ustanovama. Poseban izazov jeste objasniti programiranje na pravi način učenicima u srednjoj školi, jer programiranje kao predmet treba da se prilagodi uzrastu učenicima i uklopi sa ostatkom nastavnog plana i programa. Rad sa savremenim obrazovnim tehnologijama u toku nastave jedan je od načina da se programiranje približi učenicima i poveća kvalitet nastave. Međutim i dalje ostaju problemi oko nerazumevanja osnovnih koncepata u potpunosti, što se prenosi dalje tokom učenja i kasnije dovodi do većih problema. Tu nastaje ideja da se iskoristi povezanost programiranja sa drugim predmetima u školi kako bi se proširilo i povezalo generalno znanje i razumevanje učenika.

Hipoteza koja će biti obrađivana u ovom radu jeste da uvođenje dodatne matematike u nastavu programiranja poboljšava nivo razumevanja iz oba predmeta i samim tim uspeh kod srednjoškolskih učenika. Izložena hipoteza je formirana na osnovu ranijih istraživanja i veze između programiranja i matematike. U ovom radu biće analizirana navedene hipoteza i biće izloženi konkretni načini uvođenja matematike u nastavu programiranja.

2. POVEZANOST MATEMATIKE I PROGRAMIRANJA

Jedno od osnovnih pitanja učenika srednjih škola prilikom razmatranja odabira programiranja kao budućeg zanimanja i dalje edukacije glasi: "Da li moram da znam matematiku ako želim da budem programer?". Matematika je oblast koju mnogi srednjoškolci smatraju za težak i problematičan predmet [1], ali da li je, i u kojoj meri ona povezana sa programiranjem. Rešavanje kako matematičkih problema, tako i problema u programiranju zahtevaju od učenika kreiranje apstraktnih modela koji predstavljaju entitete i procese iz realnog sveta. Modelovanje se u obe discipline mora uraditi pomoću formalnog jezika - u matematici pomoću matematičke notacije, a u programiranju pomoću programskih jezika [2]. Ove tehnike zahtevaju od učenika visok nivo apstraktnog i lateralnog razmišljanja.

Svrha upotrebe formalnih jezika oslikava se u mogućnosti za preciznim definisanjem problema i opisa rešenja. Iako se danas najčešće ne koriste za te svrhe, računari i jesu nastali kako bi se rešili kompleksni matematički problemi. Sličan pristup koji se ogleda u načinu rešavanja problema ostavlja mnogo prostora za kombinovanje ove dve discipline. Iz ovog razloga mnoga istraživanja su sagledavala korelaciju između programiranja i matematike.

Programiranje zaista zahteva predznanje iz matematike, većina fakulteta svoje prijemne testove za smerove koji su fokusirani na programiranje i razvoj softvera baziraju na matematici. Jaka matematička osnova pokazala se kao dobar prediktor za uspeh u proceduralnom programiranju [3,4,5]. Uspeh ostvaren na uvodnim matematičkim predmetima ima pozitivnu korelaciju sa vizuelnim programerskih predmetima na narednim godinama, kao rezultat prikazano je da kursevi matematike zahtevaju od studenta razvoj kognitivnih sposobnosti koje su potrebne i u programiranju vizuelnih aplikacija [6, 7]. Moguće je pokazati sličnost u rezultatima studenata ostvarenih iz uvodnih kurseva matematike i uspeha u programerskim predmetima. Studenti koji su se isticali u oblasti programiranja takođe su demonstrirali i visok nivo matematičkih sposobnosti kao i viši IQ u poređenju sa ostalim studentima [5]. Zanimljiva je i činjenica da na uspeh iz programiranja utiču rezultati iz prirodnih nauka ali i druge sfere korišćenja računara kao što su korišćenje programa na računaru, surfovanje internetom, igranje računarskih igrica i redovna upotreba email-a [8]. Iako matematika ima veliki udeo u programiranju istraživanja su pokazala da studenti računarskih smerova nisu pokazali poboljšanje u matematičkim sposobnostima kao rezultat kurseva iz programiranja. Iznenadjujuće je to da ocena iz matematike u srednjim školama nema veliki uticaj na uspeh u računarskim naukama. Neke od pretpostavki za ovakav rezultat su razlike u gradivu koje se predaje, različiti udžbenici i drugačiji standard ocenjivanja u različitim srednjim školama [7].

3. NAČINI IMPLEMENTACIJE U CILJU UNAPREĐENJA NASTAVE U SREDNJOJ ŠKOLI

Na osnovu ranijih istraživanja i iskustava, ne može se zaključiti da je savladavanje programiranja uzročno povezano sa savladavanjem matematike kao ni obrnuto. Međutim, jasno je da su ove dve discipline međusobno korelirane. Kao što je već rečeno, obe discipline se bave apstrakcijom i imaju zadatak da mapiraju probleme iz realnog sveta u svoj domen apstrakcije kako bi rešili te probleme. Ipak način rešavanja problema se dijametralno razlikuje.

Najbitnija stvar kod učenja programiranja na osnovnim, uvodnim kursovima jeste naučiti polaznika kako da kreira algoritme. Algoritamski način razmišljanja zahteva uključivanje logičkog razmišljanja i razvijanje kreativnosti. Da bi algoritam bio što ispravniji potrebno je da zadovoljava sledeće osobine: diskretnost, rezultatnost, determinisanost i masovnost [9]. Algoritam je pojam koji se u programiranju uvodi upravo iz matematike. Razlika između algoritma u matematici i algoritma u programiranju je to što algoritam u programiranju mora da bude primenljiv na računaru, što sa sobom nosi potpuno drugačiji način razmišljanja. Dakle, već na primeru algoritma se vidi da programiranje zaista koristi matematičke koncepte, ali ih menja tako da ih prilagodi realnom svetu. Programiranjem je moguće olakšati rešavanje određenih matematičkih problema. Postoje programski jezici koji su dizajnirani tako da budu korisni pre svega matematičarima, logičarima i statističarima. Neki od takvih programskih jezika su *Matlab*, *R*, *Python*, *Julia*. Sa druge strane, matematički koncepti su pristupni već na samom početku učenja programiranja. Sve ovo dovodi do zaključka da bi se već na osnovnom kursu iz programiranja mogla uvesti i dodatna matematika koja bi pratila kurs.

Brojni su koncepti iz osnova programiranja koji se lakše shvataju ukoliko se povežu sa matematikom. Već prilikom učenja tri osnovne kontrolne strukture: sekvence, selekcije i iteracije javlja se mogućnost uvođenja matematičkih koncepata kako bi se lekcija bolje objasnila.

Na primer prilikom objašnjavanja grananja neophodno je istaći razliku između različitih varijanti *if else* naredbe. *If else* naredba se u matematici može zameniti funkcijom koja u odnosu na različite vrednosti ulaza daje različit izlaz. Ukoliko učenici dok pišu *if else* naredbu u kodu, pišu i matematičku funkciju za dati slučaj, mogu od toga imati dvostruku dobit. Prva prednost ovakvog pristupa jeste da se smanjuje verovatnoća od pravljenja greške tokom pisanja koda. Važno je da učenici od samog početka nauče da razmišljaju o onome što

kucaju. Druga prednost je u tome što će bolje razumeti matematički termin funkciju na veoma jednostavnom, praktičnom primeru. Odnosno, uočiće upotrebu matematike u realnom svetu. Primer objašnjavanja *if else* naredbe nalazi se na slici 1.

$$\begin{array}{l} \text{if } (x > 0) \\ \quad \text{skup_naredbi1} \\ \text{else} \\ \quad \text{skup_naredbi2} \end{array} \quad f(x) = \begin{cases} \text{skup_naredbi1}, & x < 0 \\ \text{skup_naredbi2}, & x \geq 0 \end{cases}$$

Slika 1. Objašnjavanje *if else* naredbe u pseudokodu i u matematičkom zapisu prilagođenom programiranju

Objašnjavanje iteracije i nizova se takođe lako može povezati sa matematikom. Niz se u programiranju može slikovito objasniti i praktično "isprobati" prolaskom kroz njega uz pomoć petlje. Vrlo brzo se uči i izvođenje različitih operacija nad članovima niza poput na primer sumiranja elemenata niza. Sabiranje svih elemenata niza može da posluži kao zgodan trenutak za uvođenje matematičkog termina sume. Nakon ovakvog načina učenja nizova, kasnije može biti lakše uvođenja niza u matematici kao preslikavanja iz skupa prirodnih brojeva (indeks niza u programiranju) u neki skup brojeva (element niza u programiranju).

Funkcije u programiranju predstavljaju veoma značajnu nastavnu celinu i izuzetno je važno pravilno ih upotrebljavati i koristiti već prilikom samog modelovanja rešenja određenog problema. Iako upotreba funkcija u matematici i programiranju nije ista, programiranje može da približi učenicima pojam funkcije. Ukoliko se pak uči funkcionalno programiranje javlja se još jača veza između matematike i programiranja kada je reč o funkcijama. Međutim postoje neki programski jezici koji su posebno pogodni za objašnjavanje programiranja uz pomoć matematike. Programski jezik *Julia* je dinamički i programski jezik visokog nivoa dizajniran da odgovori na zahteve razvijenih numeričkih i naučnih izračunavanja [10]. Ovaj programski jezik specifičan je po svojoj sintaksi koja veoma podseća na matematičke zapise.

$$f(x) = x + 3$$

$$f(x::\text{Integer})=x+3$$

```
int f(int x) {
    return x + 3;
}
```

a) Matematički zapis funkcije

b) zapis iz programskog jezika *Julia*

c) zapis iz programskog jezika *C*

Slika 2. Objašnjavanje funkcija uz pomoć matematičkog zapisa (a), zapisa iz *Julia* programskog jezika (b) i zapisa iz *C* programskog jezika (c)

Ni nakon završetka osnovnog kursa programiranja matematički pojmovi ne manjkaju. I u naprednijim kursovima programiranja javlja se mogućnost da se ove dve discipline spoje i međusobno nadopunjuju. Da bi se napisali optimalni algoritmi korisno je dobro poznavati asimptotske notacije koje se koriste za opis složenosti algoritama. Rekurzija u programiranju se mapira na rekurentne funkcije u matematici, geometrijski problemi i formule se implementiraju različitim algoritmima, a bez grafova se ne mogu zamisliti mnogi značajni algoritmi u programiranju.

4. ZAKLJUČAK

Spoj više naučnih disciplina, pogotovo u srednjim školama predstavlja veliki izazov, koji ukoliko se prevaziđe može pružiti veliki doprinos znanju i sigurnošću studenata u naučenim oblastima. Matematika i programiranje su dva predmeta koja se ukrštaju od samog početka. Učenici koji pokazuju visok nivo znanja iz matematike takođe imaju uspeha u programiranju iako su tu uključeni i ostali faktori kao što su IQ i uspeh u ostalim prirodnim naukama. Međutim postoji korelacija između ovih predmeta. Logičko razmišljanje, postavljanje problema i koraka potrebnih za njihovo rešavanje kao i kreativnost su elementi koji se zahtevaju od učenika u obe oblasti. Celine kao što su kontrola toka programa, iteracija, nizova, sumiranje, funkcije, su sve osnovni koncepti programiranja koji se vrlo lako mogu objasniti pomoću matematičkih elemenata. I složeniji problemi se takođe mogu predstaviti kroz matematiku. Grafovi i rekurzija se mogu naći i u matematici i u programiranju, a složenost algoritama se najčešće predstavlja pomoću asimptotske notacije. Objašnjavanje konceptata iz programiranja se može olakšati upotrebom

matematike a samim tim i bolje razumevanje matematike i uvid u primenu matematike u realnom svetu. Pravilno povezivanje ovih disciplina olakšalo bi i unapredilo predavanja iz obe oblasti.

5. REFERENCE

- [1] Gafoor, K.A., & Kurukkan, A. (2015). *Why high school students feel mathematics difficult: An exploration of affective beliefs*. Paper presented at the UGC Sponsored National Seminar on Pedagogy of Teacher Education, Trends and Challenges, Kozhikode, Kerala, India, August 18-19, 2015. Online submission: ERIC document ED560266
- [2] D. M. Berry, "The essential similarity and differences between mathematical modeling and programming," Science of Computer Programming, 2010, <http://dx.doi.org/10.1016/j.scico.2010.05.002>
- [3] Alsbaugh, C. A. (1970). "A Study of the Relationships between Student Characteristics and Proficiency in Symbolic and Algebraic Computer Programming." Dissertation Abstracts International, 31(4627B);
- [4] Ricardo, C. M. (1983). "Identifying student entering characteristics desirable for a first course in computer programming." Dissertation Abstracts, A44(1), 96
- [5] Ignatuk, N. (1986). "An Analysis of the Effects of Computer Programming on Analytical and Mathematical skills of high school students." Dissertation Abstracts, A47(3), 854
- [6] White, G. & Sivitanides, M. (2003). *An Empirical Investigation of the Relationship Between Success in Mathematics and Visual Programming Courses*. Journal of Information Systems Education, 14(4), 409-416.)
- [7] Balmes, Irene. (2017). *Correlation of Mathematical Ability and Programming Ability of the Computer Science Students*. [10.13140/RG.2.2.22763.23849](https://doi.org/10.13140/RG.2.2.22763.23849).
- [8] Bergin, Susan & Reilly, Ronan. (2005). *Programming: Factors that Influence Success*. ACM Sigcse Bulletin. 37. 411-415. [10.1145/1047124.1047480](https://doi.org/10.1145/1047124.1047480).
- [9] Online: <https://sh.wikipedia.org/wiki/Algoritam>
- [10] Online: [https://en.wikipedia.org/wiki/Julia_\(programming_language\)](https://en.wikipedia.org/wiki/Julia_(programming_language))