Stručni rad

PRIMJENA ALATA ZA IZRADU DIJAGRAMA TIJEKA I PSEUDO-KODA U POUČAVANJU PROGRAMIRANJA

Antonio Kovač, Ivan Dunđer
Sveučilište u Zagrebu, Filozofski fakultet, Odsjek za informacijske i komunikacijske znanosti

Sažetak
Programiranje predstavlja složen i poprilično apstraktan proces, ponajprije za one koji se prvi put susreću s tom vještinom. Samom programiranju prethodi mnoštvo predradnji, a neke od važnijih koje dotiču programere početnike jesu definiranje algoritama i pseudo-kodova. Algoritam se može shvatiti kao naputak, tj. smjer kretanja određenog programskog koda. Za razliku od programskih jezika koji imaju strogu sintaksu, algoritmi i pseudo-kodovi mogu se opisati prirodnim jezikom. Algoritmi se u novije vrijeme pišu u računalnim aplikacijama koje su namijenjene upravo za to, a jedna od najkorištenijih je „Flowgorithm“ koja će biti prikazana u ovom radu. Temeljne odrednice ove aplikacije jesu jednostavnost korištenja i višejezična podrška uključujući i podršku za hrvatski jezik, a upravo su te dvije odrednice važne za učenje i poučavanje programiranja među početnicima. Alat omogućuje i konverziju dijagrama u neke od najkorištenijih programskih jezika današnjice. S druge strane, kao zamjena za algoritam može poslužiti pisanje pseudo-kodova. Ono što je svakako njihova prednost jest što ih može pisati i osoba koja nije nužno programer, tj. ona koja dovoljno dobro poznaje problematiku određenog zadatka kojeg programer mora ispisati u obliku koda. Cilj ovoga rada jest ukazati na važnost pisanja algoritma i pseudo-kodova u učenju i poučavanju programiranja te prikazati funkcionalnosti navedenog alata.

Ključne riječi: alati u poučavanju, programiranje, generiranje programskih jezika, Flowgorithm, grafičko sučelje

1. Uvod

Pojam „programiranje“ (eng. *programming*) zapravo implicira kako se radi o obliku razmišljanja. Programiranje je uistinu oblik razmišljanja jer osoba koja programira zapravo prethodno treba promisliti o razvojnom procesu programa koji želi izraditi. Naime, nije moguće „isprogramirati“ rješenje nekog programskog problema prije no što se razmisli i konceptualizira što se zapravo želi programirati. Stoga je programiranje sveobuhvatan i složen proces. Računalno programiranje je sve više prisutno na tržištu rada i smatra se jednom od najtraženijih vještina današnjice [6]. Kako bi se uspješno stvarali novi naraštaji programera, nužno je bilo napraviti preinake unutar školskog sustava te revidirati nastavni predmet *Informatika*, koji kao takav ima najveću ulogu u obrazovanju budućih programera. Računalno programiranje u nastavi informatike predstavlja značajan aspekt obrazovanja kako na psihičkoj tako i na psihomotornoj razvojnoj osnovi. Psihički, jer razvija kognitivne mogućnosti učenika koje učenik nadalje primjenjuje i u drugim predmetima osim informatike, a psihomotornoj jer pospješuje razvoj fine motorike kod mlađih učenika prilikom tipkanja programskog koda na tipkovnici. Unutar nastavnog predmeta Informatika u osnovnim i srednjim školama u Republici Hrvatskoj, protežu se ukupno četiri domene propisane kurikulom [7], a to su: *e-društvo*, *Digitalna pismenost i komunikacija*, zatim *Informacije i digitalna tehnologija* te *Računalno razmišljanje i programiranje* koja je od ključnog interesa u ovom radu. Unutar domene „Računalno razmišljanje i programiranje“ odabran je programski jezik Python kao temeljni jezik za poučavanje u nastavi informatike [6].Ovaj rad pojašnjava značaj i ulogu algoritma u učenju i poučavanju programiranja kao važnog elementa unutar domene „Računalno razmišljanje i programiranje“, naglašava važnost primjene dijagrama tijeka u izradi programa i učenju, prikazuje rad s alatom „Flowgorithm“ koji se može upotrijebiti za vizualizaciju tijeka programa te ukazuje na važnost razvoja pseudo-koda kao važne predradnje prije same implementacije programskog koda u odabranom programskom jeziku.

2. Algoritam kao temelj u procesu programiranja

Ispijanje omiljenog toplog napitka ili kušanje omiljene hrane koju je čovjek pripremao nije ništa drugo već krajnji rezultat procesa „programiranja“, tj. planiranja i odrađivanja koraka koji su doveli do željenog cilja, tj. konkretnog ishoda. Kao primjer može se uzeti postupak kuhanja čaja. Svima je poznato da su za to potrebni sljedeći elementi: voda, posuda za vodu, šalica, čaj, a postupak bi mogao biti sljedeći: u posudu za kuhanje potrebno je uliti vodu, čekati dok voda zavri, isključiti štednjak, u šalicu staviti čaj, uliti uzavrelu vodu u šalicu i pričekati određeno vrijeme [6]. Postupak rješavanja problema ili postizanje određenog cilja u programiranju naziva se algoritam, pa je ujedno gore navedeni postupak, pravi primjer algoritma. U računalnom programiranju princip je identičan. Elementi koje je potrebno uvesti u programski jezik nazivaju se ulaznim elementima (eng. *input*), a ono što je rezultat tih ulaznih elemenata jest izlaz (eng. *output*) [6]. Kao i u svakom drugom poslu i pristupu rješavanja problema tako i u izradi računalnog programa, planiranje može uštedjeti mnogo truda, novca ali i vremena, što je u programiranju od velikog značaja. Ono što se planiranjem definira jest tko, kada i što radi. Na ta pitanja odgovori su često složeniji, pa se pri izradi programa treba konzultirati više stručnjaka s različitih područja [6]. U slučaju da program izrađuje samo jedna osoba, planiranje također dolazi do izražaja i uvelike se mogu predvidjeti i rasporediti pojedine faze izrade programa. Planiranje je dakle dio rješavanja problema, a algoritam se može definirati kao naputak u vezi rješavanja

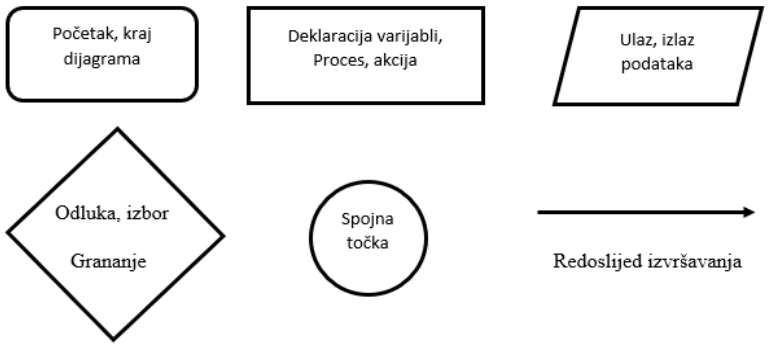
nekog zadatka [4]. Algoritam se može definirati i kao postupak sastavljen od konačnog broja koraka koji ukazuju na slijed operacija koje treba obaviti nad početnim objektima kako bi se dobili završni objekti ili rezultati [3]. Brojni su primjeri pomoću kojih se može prikazati rad algoritma, poput primjera kuhanja čaja. Rad algoritama učenicima je lako predočiti primjerima recepata, slaganjem kockica, rješavanjem puzzli i sl., jer se njima najjednostavnije mogu prikazati funkcionalnost, svrha i tijek algoritma. Dakle, može se zaključiti da algoritam predstavlja niz koraka, čijom se primjenom može postići neki željeni ishod. Za algoritam se kaže da je dobar naputak, jer se njime može prikazati i uspješno izvršiti neka uputa koju pojedinac možda i nije fizički izvodio ranije [6].

2.1. Uloga algoritma u računalnom programiranju

Kada se govori o računalnoj primjeni algoritama, algoritam se stvara na temelju specifikacije, tj. konkretizacije nekog problema. Zapisuje se uz pomoć niza jednostavnih operacija koje se mogu pretvoriti u naredbe konkretnog programskog jezika. Postupnim izvršavanjem tih naredbi, tj. na temelju ulaznih podatka može se dobiti željeni rezultat [4]. Algoritmi se prikazuju dijagramima tijeka ili pseudo-kodom. Dijagrami tijeka prilikom opisivanja složenijih algoritama znaju biti izrazito nepregledni, a kao zamjena za njih koristi se pseudo-kod [5]. No, prije no što se pobliže pojasni uloga pseudo-koda valja definirati dijagram tijeka.

2.2. Dijagram tijeka

Dijagram tijeka (eng. *flowchart*) je grafički prikaz algoritma koji olakšava prikaz programa. On dakako nije nužan, ali može biti vrlo koristan i najčešće je od pomoći početnicima u svijetu programiranja. Neovisan je o programskom jeziku i računalu. Njegova glavna svrha jest vizualizacija i preglednost programskog problema, a označava se jednostavnim geometrijskim likovima spojenim crtama [6]. Crte ukazuju na tijek zadataka pa odatle i naziv dijagrama [4]. Na slici 1 prikazani su najčešći elementi, tj. dijelovi dijagrama tijeka i dana su objašnjenja svakog od likova [5].



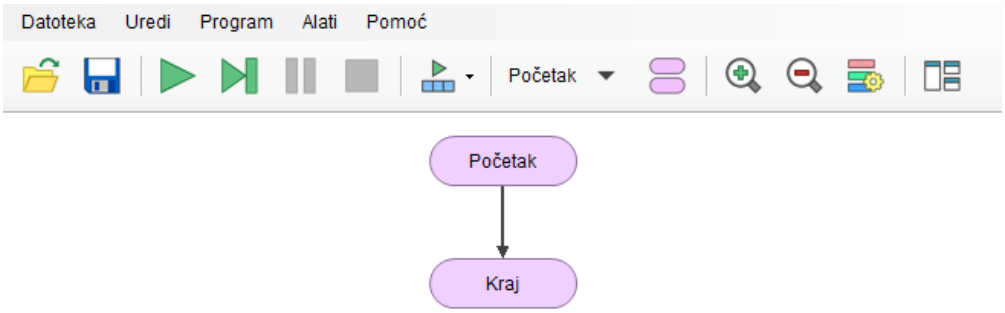
Slika 1. Geometrijski likovi za označavanje dijelova dijagrama tijeka
Izvor: rad autora

Dijagrami se mogu izrađivati pomoću pisala na nekom sredstvu ili uz pomoć nekog od specijaliziranog programa na računalu. Alati koji se pak koriste u svrhu izrade dijagrama tijeka jesu alati koji imaju mogućnost dodavanja automatskih (gotovih) geometrijskih likova i oblika [6]. Neki od alata koji se mogu koristiti za izradu dijagrama tijeka su LibreOffice, FreeOffice ili Google Workspace. Među najkorištenijima svakako je i Microsoft Word koji sadrži funkciju „Oblici“ (eng. *Shapes*) te Bojanje (eng. *Paint*).

No, uz ove poznate postoje i posebno specijalizirani alati za izradu dijagrama, a jedan od najpoznatijih je alat pod nazivom „Flowgorithm“ [6].

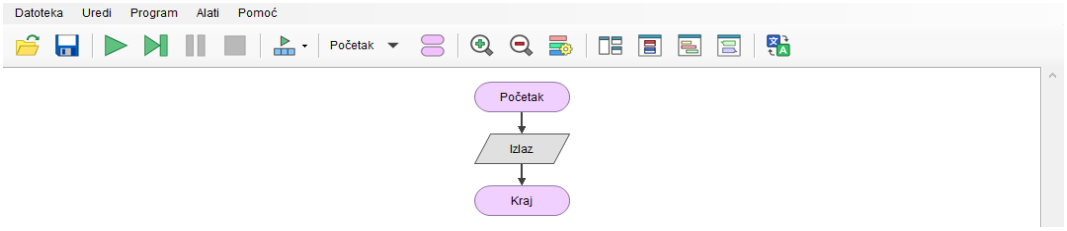
3. Alat Flowgorithm u izradi dijagrama tijeka

Flowgorithm je besplatna aplikacija koja pomaže u izradi programa primjenom jednostavnih grafičkih dijagrama tijeka [1]. U pravilu se programi pripremaju uređivačima teksta pa ovisno o kojem se programskom jeziku radi, taj proces može biti jednostavan ili vrlo složen za programera početnika. Primjenom dijagrama tijeka početnici se mogu u potpunosti usmjeriti na koncepte i paradigme programiranja umjesto da se upoznavaju sa svim detaljima nekog programskog jezika. Velika prednost ovog alata je što omogućuje interaktivnu konverziju dijagrama tijeka u velik broj podržanih programskih jezika i okruženja: C#, C++, Java, JavaScript, Lua, Perl, Python, Ruby, Swift, Visual Basic .NET i VBA (Visual Basic for Applications koji se koristi u Microsoft Office paketu), Ada 95, AppleScript, Bash, Fortran 2003, MATLAB, Nim, Pascal, PHP, Powershell, QBasic, Scala, Smalltalk, Swift, Transact-SQL i TypeScript. Pored navedenog, rezultate je vrlo lako interpretirati, varijable se lako grafički pregledavaju, rekurzije se izvršavaju sigurno, moguće je primijeniti petlje, polja i razne izraze, a postoji i višejezična podrška unutar alata.Flowgorithm je izvrstan alat koji se koristi u nastavi informatike za izradu dijagrama tijeka te za njegovo izvršenje. Izvršavanje se odvija u konzoli te je vrlo jednostavan za korištenje. Alat je višejezičan i nudi podršku i za hrvatski jezik, stoga je koristan i za niže razrede osnovne škole, gdje učenici još uvijek nisu spremni za primjenu alata na stranome jeziku. Početni elementi koji se nalaze jesu „Početak“ i „Kraj“. Algoritam je moguće izvršavati jednostavnim klikom na tipku zelenog trokuta, nakon čega se otvara novi prozor, tj. konzola u kojoj se naredba algoritma izvršava i prikazuje. Na slici 2 prikazano je sučelje navedenog alata [6].



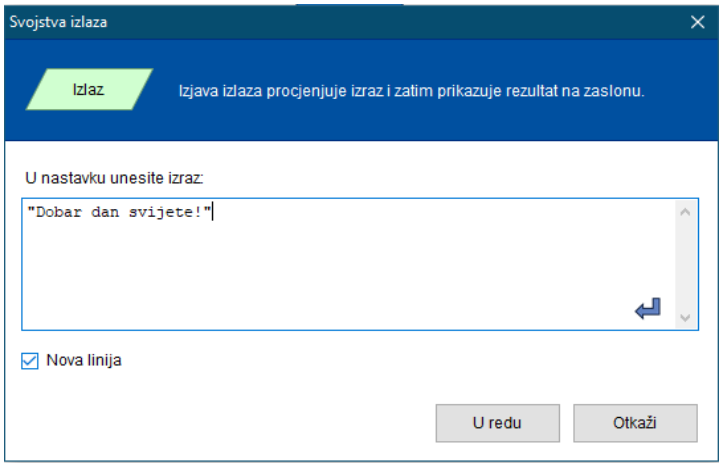
Slika 2. Sučelje alata „Flowgorithm“
Izvor: rad autora

Prilikom otvaranja programa automatski se dodjeljuju osnovne funkcije („Početak“ i „Kraj“), koje se mogu preimenovati. Dodavanje novih elemenata vrši se tako da se klikne na strelicu te se u izborniku odabere željena funkcija. Primjerice, ako se želi ispisati rečenica „Dobar dan svijete!“, onda je potrebno dodati funkciju „Izlaz“ (eng. *Output*), kao što je prikazano na slici 3 [6]. Po zadanim postavkama (eng. *default*) dodijeljena će funkcija dobiti naziv „Izlaz“ odnosno „Output“, ovisno o jezičnoj inačici alata.



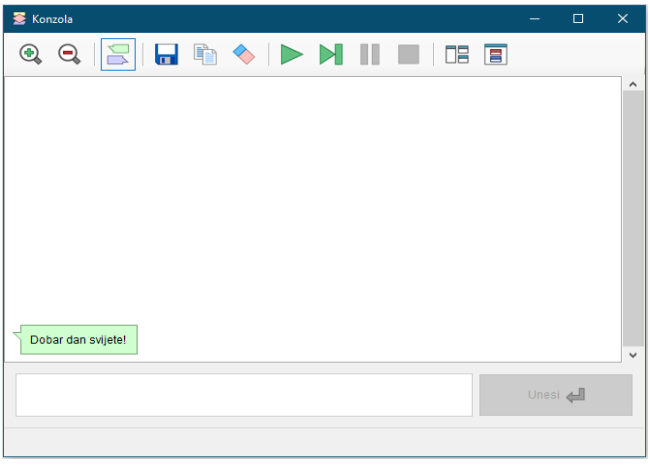
Slika 3. Prikaz funkcije za izlaz u alatu „Flowgorithm“
Izvor: rad autora

Dakako, u funkciju je potrebno unijeti što se želi odraditi ili ispisati, a unos u funkciju vrši se tako da se dva puta klikne polje funkcije te se u novootvorenom prozoru unese željeni izraz u za to predviđenu prazninu. Napomena je da se rečenice ili tekstualni podatci (tzv. stringovi) unose pod znacima navoda [6].



Slika 4. Prozor za unos željenog izlaza
Izvor: rad autora

Na slikama 4 i 5 prikazani su postupci unosa rečenice „Dobar dan svijete!“ u funkciju „Izlaz“ te izvršenje algoritma u konzoli [6].



Slika 5. Konzola za ispis programa
Izvor: rad autora

Uobičajena interakcija s računalom je putem konzole. Korisnik može unositi podatke putem svoje tipkovnice, no Flowgorithm prikazuje podatke u obliku prozora kakve nalazimo u tipičnim *instant messengerima*. Zapravo se čini kao da se programer

„dopisuje“ s računalom. Dinamika „razgovora“ s računalom prikazana oblacima (eng. *chat bubbles*) kodirana je bojama kako bi se razlikovao unos od izlaza u dijagramima tijeka. U pravilu je korisnički unos prikazan plavom bojom, dok je odgovor računala označen zelenom bojom. Korisnik doduše ima mogućnost deaktivacije takvog prikaza i aktivacije klasičnog tekstualnog prikaza.Prozor za praćenje varijabli prati promjene vrijednosti varijabli kako se program izvodi. U bilo kojem trenutku moguće je ispisati vrijednost neke varijable. To se ne odnosi samo na jednostavne varijable, već i na polja [2]. Nadalje, svaka varijabla je kodirana bojom ovisno o tipu podatka, tako da je odmah na prvi pogled moguće odgonetnuti koji se tip podatka pohranjuje u programu: cijeli brojevi (tzv. integeri) prikazani su plavom bojom, decimalni brojevi ljubičastom, stringovi crvenom, a logičke (tzv. bool) varijable *teal* plavom bojom [2]. Kao što je već ranije navedeno, ono što je izrazita snaga alata Flowgorithm jest mogućnost interaktivnog generiranja stvarnog programskog koda. Prozor „Source Code Viewer“ (u prijevodu „preglednik izvornog koda“) može izvršiti konverziju dijagrama tijeka u nekoliko glavnih programskih jezika. To je posebno korisno programerima početnicima koji žele naučiti programirati u nekom konkretnom popularnom programskom jeziku [2]. Generirani kod označen je istom pozadinskom bojom kao i dio dijagrama tijeka, tj. elementi koji su zaslužni za nastanak odgovarajućeg odlomka programskog koda. Na taj način moguće je vizualno pratiti i odrediti korelaciju između programskog koda i dijagrama tijeka. Nadalje, ako se označi neki element na dijagramu tijeka, odgovarajući odlomak programskog koda bit će također označen pozadinskom bojom.Sljedeći pseudo-kodovi također su podržani: Auto Pseudocode, Gaddis Pseudocode i IBO Pseudocode [2]. Ukoliko željeni programski jezik i/ili pseudo-kod nije podržan za vrijeme instalacije alata „Flowgorithm“, moguće je koristiti prilagodljive programske predloške. To su zapravo jednostavne tekstualne datoteke koje se mogu izraditi pomoću bilo kojeg uređivača teksta. Nakon što se učitaju u program, dijagram tijeka se može automatski konvertirati u ciljnu sintaksu željenog programskog jezika [2].Alat podržava i šaroliku ponudu različitih shema boja koje dolaze predinstalirane s alatom. To ujedno pojednostavljuje rad i izvoz dijagrama tijeka u slikovnu datoteku koristeći razne stilove i prikaze. Flowgorithm može s interneta automatski preuzeti i učitati više od sto dodatnih shema boja, a ako ni to nije dovoljno, alat ima ugrađen uređivač shema boja koji omogućuje izradu vlastitih stilova [2]. Sam alat ima zaista impresivnu višejezičnu podršku za sučelje. Podržani su, između ostaloga [2]: afrikaans, arapski, katalonski, kineski, hrvatski, češki, nizozemski, engleski, farsi, francuski, galicijski, njemački, hebrejski, mađarski, indonežanski, talijanski, japanski, korejski, malajski, mongolski, poljski, portugalski, rumunjski, ruski, slovenski, španjolski, švedski, turski, ukrajinski itd.Alat podržava jednodimenzionalna polja, pre-test i post-test petlje, for petlje, eksplicitno deklariranje varijabli, sigurne rekurzije (održavajući interni stog kako ne bi došlo do rušenja programa ukoliko korisnik slučajno pokrene odbjegli rekurzivni poziv) koje korisniku prikazuju poruku o pogrešci, operatore iz različitih sintaksi više programskih jezika, korisničke funkcije koje mogu (ali ne moraju) vratiti neku vrijednost nazad u glavni program te preko 20 ugrađenih funkcija [2]. Nadalje alat podržava više različitih stilova dijagrama tijeka, poput klasičnog stila, IBM stila, SDL stila itd. Dijagrame je moguće eksportirati kao bitmap u PNG formatu ili kao vektor u SVG formatu, i to u svakom od podržanih shema boja. Prozore unutar alata Flowgorithm je također moguće namještati pomoću opcije „Layout Windows“, a izrađene programe je moguće sačuvati kao XML datoteke [2].

4. Značaj i primjena pseudo-koda

Riječ „pseudo“ dolazi od grčke riječi *pseudos* što znači laž. „Lažan“ je, jer premda nalikuje na računalni program, on to uistinu nije, jer nije napisan u programskom jeziku koji bi se mogao izravno primijeniti na nekom računalu [6]. On se sastoji od „kratkih izraza (riječi) na govornom jeziku koji opisuju i ukratko objašnjavaju pojedine zadatke algoritma“ [4]. Ti izrazi trebaju biti napisani na razumljiv način, tj. osobe koje govore taj jezik trebaju razumjeti izraze. Važno je napomenuti kako osoba koja piše pseudo-kod ne mora nužno biti i programer, čak niti pripadati toj domeni, već se zapravo usredotočuje na način i princip rješavanja problema ili zadatka [4]. Za razliku od dijagrama koji koristi geometrijske likove, pseudo-kod koristi prirodni jezik i on je puno prikladniji za opis složenijih algoritama. Proces kuhanja čaja spada u jednostavne primjere algoritma, kao i izračunavanje opsega stranica pravokutnika. Prilikom opisa nekog algoritma, pseudo-kod u svojem sadržaju koristi izraze kojima se ljudi svakodnevno koriste [6]. Na autoru pseudo-koda je da razmotri i na kraju odluči koliko će detaljno riječima opisati neki algoritam. Međutim, ne treba previše detaljizirati, jer se tako gubi smisao opisivanja algoritma pseudo-kodom. Time se zapravo nije dobila početna svrha pseudo-koda, a to je preglednost početnog programerskog problema. Također, nije dobro pisati kod previše šturo jer nedostatak detalja može dovesti do nejasnoće pri čitanju, osobito početnicima [5]. U konačnici, pseudo-kod bi trebao biti načinjen tako da ga bilo koji programer može pročitati, te na osnovu njega napisati program u odabranom programskom razvojnom okruženju i programskom jeziku [4]. Iz dosadašnjih primjera i opisa u ovom radu, može se zaključiti kako su algoritam i pseudo-kod oblik razmišljanja i način prezentiranja podataka u računalu. Važno je dakle planirati, razmotriti, opisati, prezentirati i sadržajno prikazati određeni problem. Pri procesu kreiranja novog pseudo-koda trebalo bi načiniti popis ključnih i temeljnih zadataka koje treba riješiti programom. Popis podataka najčešće se temelji na prethodno definiranom dijagramu tijeka ili algoritmu [6]. Zadatke je potrebno razlučiti na što manje zadatke (podzadatke), točnije riječ je o određenoj vrsti klasifikacijskog postupka, tj. svjesnog i namjernog sređivanja podataka. Svaki podzadatak bi se trebao moći opisati kratkim i jednostavnim, a opet razumljivim jezičnim izrazom. Preporučuje se koristiti uvlake (eng. *tab*), jer se tako stvara bolja preglednost i razumljivost koda. U programskim jezicima poput Pythona i drugih, uvlake se događaju automatski prilikom logičkih upita. U primjeru u nastavku prikazan je oblik pseudo-koda, tj. zadatak koji prikazuje prolazak automobila kroz križanje sa semaforom [6].

Vozi prema križanju
Uoči semafor
Donesi odluku na temelju boje svjetla
Zeleno svjetlo:
    Nastavi vožnju.

Žuto svjetlo:
    Procijeni ima li dovoljno vremena za zaustavljanje.
        Ako ima: uspori i zaustavi vozilo.
        Ako nema: prođi kroz križanje.

Žuto treptavo svjetlo:
    Procijeni opasnost na križanju jer semafor ne radi.
        Ako ima: uspori vozilo i pripazi na promet i pješake.

Ako nema: oprezno nastavi kretanje vozilom uz sporu vožnju.

Crveno svjetlo:

Zaustavi vozilo i čekaj promjenu svjetla.

U ovom primjeru vidljivo je da su zadaci (situacije) podijeljene na podzadatke, tj. na moguće slučajeve. Zadatak se mogao opisati i na drugačiji način. Stvar je procjene. Naime, kreator koda procjenjuje koliko detaljno je potrebno razlučiti pojedini zadatak [6]. Ono što je sigurno jest da pseudo-kod mora biti dovoljno jasan i dovoljno podroban, jer programeri moraju moći na temelju pseudo-koda i bez dodatnih pomagala i objašnjenja napisati program u odabranom programskom jeziku [4].

5. Zaključak

U radu su prikazani osnovni principi izrade dijagrama tijeka i pseudo-koda. Načelno gledano oba principa imaju sličnu svrhu, a to je olakšati razumijevanje i čitanje određenog programskog koda. U radu je prikazan alat pod nazivom „Flowgorithm“ koji je postao sinonim za softversku izradu dijagrama tijeka, što zbog svoje jednostavnosti, što zbog mogućnosti povezivanja sa stvarnim programskim jezicima. S druge strane, algoritmi i dijagrami tijeka su od velikog značaja programerima početnicima, jer omogućuju vizualan prikaz onoga što program upisuje ili ispisuje. Ipak, svojevrstan nedostatak dijagrama tijeka jest njegova nepreglednost pri pisanju složenijih programskih kodova, stoga se njegova uporaba ipak zadržava pri početničkom učenju i poučavanju programiranja. Pseudo-kod se može smatrati rudimentarnim oblikom programskog koda, ali ono što ga bitno razlikuje jest jezik kojim je pisan. Može se reći kako je pseudo-kod konkretizacija algoritma, tj. detaljima oblikovan algoritam. Algoritmi i pseudo-kodovi nisu nužan preduvjet za pisanje određenog programa u programskom jeziku, ali mogu biti i jesu od iznimne koristi pri početnom učenju programiranja.

6. Literatura

[1.] Flowgorithm. URL: http://www.flowgorithm.org (pristup 08.01.2023.)
[2.] Flowgorithm (Features). URL: http://www.flowgorithm.org/about/features.html (pristup 08.01.2023.)
[3.] Galešev, V., Dmitrović, N., Vlahović, V., Kager, D., Lučić, K. (2019). *Informatika 1: Udžbenik informatike iz 1. razred općih i prirodoslovno-matematičkih gimnazija te 2. razred klasičnih i jezičnih gimnazija*, SysPrint, Zagreb. Dostupno na: https://sysprint.hr/eudzbenici/infgim2019/ (20.05.2021.)
[4.] Grundler, D., Blagojević, L. (2008). *Informatika 1: udžbenik informatike za 1. razred općih, jezičnih i klasičnih gimnazija*, Školska knjiga, Zagreb.
[5.] Hruška, M. (2018). *Osnove programiranja Python*. Srce, Zagreb. Dostupno na: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d450_polaznik.pdf (26.05.2021.)
[6.] Kovač, A. (2021). *Analiza i primjena programskog jezika Python u nastavi informatike* // diplomski rad. Filozofski fakultet Sveučilišta u Zagrebu, Zagreb.
[7.] Ministarstvo znanosti i obrazovanja (2018). *Kurikulum nastavnog predmeta Informatika za osnovne škole i gimnazije*, MZO, Zagreb. Dostupno na: https://mzo.gov.hr/UserDocsImages/dokumenti/Publikacije/Predmetni/Kurikulum%20nastavnog%20predmeta%20Informatika%20za%20osnovne%20skole%20i%20gimnazije.pdf (10.01.2023.)