Sveučilište Jurja Dobrile u Puli
Odjel za informacijsko-komunikacijske znanosti

PATRIK LEKAJ

PROGRAMSKI JEZICI NAMIJENJENI UČENJU PROGRAMIRANJA

Završni rad

Pula, rujan 2017. godine

Sveučilište Jurja Dobrile u Puli
Odjel za informacijsko-komunikacijske znanosti

PATRIK LEKAJ

PROGRAMSKI JEZICI NAMIJENJENI UČENJU PROGRAMIRANJA

Završni rad

JMBAG: 0303038017 redoviti student

Studijski smjer: Informatika

Predmet: Programiranje

Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijsko-komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujanj 2017. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Patrik Lekaj, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA
o korištenju autorskog djela

Ja, Patrik Lekaj dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom "Programski jezici namjenjeni učenju programiranja" koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SADRŽAJ

UVOD

Pojavom osobnog računala u sedamdesetima, došlo je do veće popularizacije programiranja. Rani počeci programiranja bili su pretežito vezani uz matematiku i njegova se edukacija obično vršila samo na sveučilištima. U današnje vrijeme, djeca uče o računalima i informacijskoj tehnologiji od malih nogu i shodno tome, povećana je edukacija programiranja. Savladati programiranje nije lak posao. Mnogi početnici koji uopće nemaju iskustva s programiranjem, ne znaju da programiranje nije samo pisanje koda, već konstruiranje rješenja problema, otklanjanje grešaka u kodu, razvijanje i mijenjanje logičkog razmišljanja.

Seymour Papert u svom radu *Mindstorms,* govori o idejama kako djecu treba naučiti programirati na način da djecu treba naučiti kako učiti i da ih se ohrabruje kod traženja novih iskustva s učenjem. Kada bi promatrali programiranje kao takvog, vidjeli bi da je ono apstraktno, shodno tome i teško za razumijevanje. U isto vrijeme programiranje je dobro za razvijanje apstraktnoga razmišljanja. Većina mladih učenika koji pokušaju učiti programiranje u osnovnim školama, nemaju dovoljno razvijeno apstraktno mišljenje i učenje programiranja im se čini teškim. Pomoć učenicima početnicima u osnovnim školama pruža se kroz vizualne programske jezike, koji otklanjaju probleme ne shvatljive sintakse kroz vizualne reprezentacije. Pored velikog broja programskih jezika, odabir jezika namijenjenih učenju nije lak, i potrebno je razmotriti višestruka rješenja. (Bubica i sur., 2013)

Cilj ovog završnog rada je upoznavanje s programskim jezicima namijenjenim učenju programiranja. Za odabir prvog programskog jezika, potrebno je razmotriti određene kriterije. Jezici koji nisu pogodni za učenje programiranja, obično imaju kompliciranu sintaksu i zbunjujuću semantiku. Simboli i nepotrebne kratice koje u jeziku stvaraju konfuziju, početnicima ostavlja loš dojam o jeziku, a sukladno i smanjenu motivaciju učenja. Problem teške sintakse dovodi do velike praznine između učenja sintakse i učenja programiranja kroz algoritme. Radošević i suradnici navode da je manjak znanja programerske sintakse, zajedno sa slabijim sposobnostima rješavanja problema, uzrok velikog iskoraka poteškoća u programiranju. Iako je manjak prijašnjih znanja u programiranju, vjerojatno najveći problem početnicima programerima, veliki problem je kada početnici zaključuju da je programiranje za njih previše teško. U tim situacijama početnici ponekad odustaju od učenja programskih jezika i od sudjelovanja u aktivnostima opisanim u nastavnom programu, to jest silabusu.

U ovom završnom radu pisat će se o programskim jezicima koji su namijenjeni učenju. Ovakva vrsta programskih jezika imaju cilj olakšati učenje programiranja i svaki od njih ima drugačiji pristup. U prvom poglavlju objašnjavaju se programski jezici općenito zajedno sa njihovom podjelom. Uz to opisuje se povijesti programskih jezika i povijest jezika namijenjenih učenju. U sljedećim poglavljima opisani su programski jezici namijenjeni učenju. To uključuje programske jezike Logo, BASIC, Pascal, Scratch, C++, Java i Python. Obrađeni jezici opisani su u kratkim osnovama zajedno sa ciljevima, primjerima sintakse i načinima korištenja. Sedmo poglavlje s naslovom "Usporedba programskih jezika namijenjenih učenju", uspoređuje opisane programske jezike po kriterijima navedenim u drugom poglavlju. Zadnje poglavlje sadrži zaključak i procjena za budućnost programskih jezika namijenjenih učenju.

1. PROGRAMSKI JEZICI OPĆENITO

Potreba za automatizmom rješavanja problema dovela je do pojave programskih jezika. Programiranje je sastavljanje uputa namijenjenih računalu o tome što i kako treba napraviti. Upute se slažu kroz pisanje koda. Kako postoji mnogo načina i razloga za pisanje programa tako i postoje različiti programski jezici.

1.1. Razvoj programskih jezika kroz povijest

Početak razvitka programskih jezika uzrokovalo je sinteza numeričke kalkulacije, pred određene operacije i izlaz rezultata, skupa sa načinom kako organizirati i napisati ulazne instrukcije tako da je sve to relativno lako, razumljivo i uporabivo za čovjeka.

1954. godine izumljen je programski jezik FORTRAN. Bio je to prvi *high-level*[1] programski jezik u širokoj uporabi s funkcionalnom implementacijom. Dopuštao je programerima da specificiraju naredbe kalkulacije direktno sa pisanjem formule (npr. rezultat = prviBroj*4 + drugiBroj*2). Programski tekst to jest izvorni kod je preveden u strojni jezik pomoću posebnog programa nazvan *compiler*[2]. Prvi kompajler izdan je 1957. i jezik je još danas u uporabi.

Samo godinu nakon prvog FORTRAN kompajlera dolazi programski jezik Lisp (prvobitno LISP). Lako prepoznatljiv zbog velikog broja zagrada koji se pojavljuju u kodu. Napravljen kao implementacija matematičke notacije za računala. Lisp, kao jedan od prvih programskih jezika, pridonijeto je raznim idejama računalne znanosti uključujući strukture podataka (stablo), automatski menadžment pohranjivanja (*garbage collection*[3]), dinamičko provjeravanje koda, uvjeti, funkcije višeg reda, rekurzija, itd. Favoriziran kod istraživanja umjetne inteligencije.

Ne dugo nakon Lisp-a 1959. pojavljuje se programski jezik COBOL. Namijenjen i dizajniran za poslovno korištenje. Primjenjivan je u poslovnim, financijskim i administrativnim sustavima za kompanije i vlade. Još je u uporabi kod *legacy*[4] softvera, ali zbog sve većeg pada popularnost i

1 Programski jezik koji omogućava programeru da piše program u terminima koji su više apstraktni od assembly programskog jezika (visoki nivo apstrakcije)
2 Kompjuterski program koji za ulaz prima izvorni kod nekog programskog jezika, a na izlazu izbacuje isti program u drugom računalnom jeziku (specifično ciljani jezik, najčešće je to u strojni jezik)
3 Program koji je zadužen za čišćenje memorije koja se više ne koristi
4 Softver koji je zastario

sve manjeg broja iskusnih COBOL programera, programi se ponovo pišu pomoću modernijih programskih jezika.

1970. godine izdan je programski jezik Pascal (po Blaise Pascalu). Namjena mu je bila da bude mali i efikasan programski jezik koji će poticati dobre programerske prakse koristeći strukturirano programiranje i strukturirane podatke. Bio je u širokoj komercijalnoj uporabi u osamdesetima.

Programski jezik C originalno je razvio Dennis Ritchie 1972. Prvobitna uporaba je bila za ponovnu implementaciju Unix operativnog sustava. C je *low-level*[5] programski jezik opće namjene. Iako je C jezik sa *low-level* mogućnostima, njegov dizajn omogućava korištenje *cross-platform*[6] programiranja, što doprinosi pristupačnost na raznim uređajima, od ugrađenih sustava (*embedded*[7]) do super računala. Ekstremno je brz kada se izvodi i zbog toga je široko korišten kod razvijanja operativnih sustava, *drivera*[8], računalnih igara, itd.

1980. pojavljuje se programski jezik Ada (po Ada Lovelace). Ada je *high-level* programski jezik, nastao većinom po uzoru na Pascal. Napravljen po ugovoru za Ministarstvo obrane Sjedinjenih Američkih Država.

C++ je low-level programski jezik opće namjene nastao 1983. Tvorac Bjarne Stroustrup i ostali iz Bell Labs-a proširili su programski jezik C s klasama (prvobitni naziv je bio C s klasama). Ono što je falilo C-u je mogućnost objektno orijentiranog programiranja, što je C++ dobio s implementacijom klasa. C++ ima sličnu namjenu kao i programski jezik C i upotrebljava se kod embedded sustava, sistemskih programa, i kod općenito programa koji zahtijevaju velike performanse i imaju kompleksan sustav. Mnogi moderni programski jezici poput C# i Java preuzeli su neke ideje i dizajn iz C++-a. Iste godine nastao je i Objective C koji za razliku od C++-a dinamičan po stilu, nema mogućnost višestrukog nasljeđivanja, itd.

1987. nastao je *high-level* programski jezik Perl, a razvio ga je Larry Wall. Perl je opće namjene i napravljen je za procesiranje izvješća za UNIX sisteme. Poznat je po svojoj fleksibilnosti, snazi i raznolikoj primjeni, poput: grafičkog programiranja, administracije sistema, mrežnog programiranja, programiranja u području financija, bio-informatike, itd.

5 Jezici bliži strojnom jeziku nego ljudskom
6 Mogućnost podržavanja više platformi
7 Računalni dio sustava koji je ugrađen u veći dio mehaničkoga ili električnog sustava
8 Program zadužen za upravljanje određene komponente spojene na računalo

Python je *high-level* programski jezik opće namjene nastao 1991. Python je prevođeni programski jezik kojeg je stvorio Guido van Rossum. Prati filozofiju dizajna koja omogućava veću čitljivost koda koristeći razmake u kodu za označavanje blokove koda (umjesto vitičastih zagrada ili riječi). Python je jezik koji omogućava programerima da pišu koncepte u manje linija koda nego npr. C++ ili Java. Podržava različite paradigme programiranja što ga čini prilagodljivim. Ne dugo nakon Pythona izlazi programski jezik Ruby koji je sličan Pythonu.

Java je *high-level* programski jezik opće namjene, pojavljen 1995. Java je objektno orijentirani jezik i napravljen je s namjerom da omogući programerima pokretanje Java koda na svim platformama koji su Java kompatibilni bez potrebe ponovnog kompajliranja. Java kod je obično kompajliran u *bytecode*[9] koji se pokreće na Java virtualnoj mašini (*JVM*[10]) bez obzira na računalnu arhitekturu. Iz tog razloga i zbog bogate biblioteke, Javu prati velika popularnost. Programski jezik C# nastao je kao odgovor na Javu i imaju svoje sličnosti.

PHP je skriptni jezik namijenjen za pisanje serverskih skripti namijenjenih za razvijanje dinamičkih web aplikacija. Originalno ga je napravio Rasmus Lerdorf u 1995. godini. PHP kod se može ubaciti u HTML ili HTML5 kod ili se može koristiti u kombinaciji sa raznim sustavima predloška, *CMS*[11] sustavima, *web framework*[12].

Javascript je *high-level* programski jezik koji pored HTML i CSS-a sačini glavnu jezgru u proizvodnji sadržaja web-a. Većina web stranica koristi Javascript i svi ga moderni web browseri podržavaju. Podržava različite paradigme programiranja. Osim izvođenja koda na klijentu, Javascript ima mogućnost izvršavanja koda na poslužitelju, što omogućava pisanje *front-enda*[13] i *back-enda*[14] pomoću jednog programskog jezika. Isto tako Javascriptom je moguće pisati *desktop* i mobilne aplikacije.

9 Instrukcijski set koji je namijenjen za pokretanje na posebnom softverskom interpreteru
10 Omogućava pokretanje Java programa
11 *Content Managment System:* softver za potporu kreiranja i modificiranja digitalnog sadržaja
12 Softverski okvir napravljen za potporu razvijanja web aplikacija
13 Prezentacijski dio web-a
14 Dio web-a za pristup i manipulaciju podacima

1.2. Vrste i tipovi programski jezika

Svi programski jezici razlikuju se po konceptima koji definiraju kako će se pojedini dijelovi jezika ponašati (sintaksa i semantika), koja je njihova namjena i kako se izvode.

1.2.1. Programski jezici po načinu izvođenja

Kada bi podijelili programske jezike po načinu izvođenja, mogli bismo ih svrstati u tri kategorije: jezici koji su prevođeni u strojni jezik (kompajlirani), jezici koji su interpretirani i jezici koji se izvršavaju u obliku *bytecode*-a u virtualnoj mašini. U današnjici baš i nije sve tako crno i bijelo i uobičajena je kombinacija procesa kompajliranja i interpretiranja programskog jezika. Rijetkost je naći programski jezik koji je čisto interpretiran. Mnogo jezika koriste proces gdje se *high-level* kod prevodi u *bytecode* jezik koji je bliži za određenu platformu (gdje se interpretacija *bytecode*-a izvršava brže).

Kompajler i interpreter rade isti posao: prevode programski jezik u neki drugi jezik, koji je obično bliži hardwareu (npr. strojni jezik). Tradicionalno "kompajlirani jezik" znači da se prijevod izvršava odjednom, obično ga pokreće razvijač i rezultat kompajliranja (na primjer .exe datoteka) je distribuiran korisnicima. Primjer kompajliranih programskih jezika je C++. Proces kompajliranja traje duže jer se u tom procesu dešavaju velike optimizacije kako bi rezultat kompajliranja bio brži nakon pokretanja. Korisnici nemaju alate i znanje za izvršavanje kompilacije sami, pa se izvršna datoteka ne može optimizirati za svaki hardware specifično.

Tradicionalni interpretirani programski jezici su oni jezici u kojima se prevođenje odvija tijekom same egzekucije koda, to jest neposredno prije svake linije koda. Ovaj način prevođenja je spor i ne mogu se izvršavati skupe optimizacije kad se kod neposredno izvršava. Iz tog razloga postoje jezici koji su i interpretirani i kompajlirani, neki od primjera su Java i C#. Kod *bytecode*-a prvi korak je kompilacija trenutnog ljudskog čitljivog koda u *bytecode*. *Bytecode* je skup formiranih instrukcijskih setova koji su dizajnirani da budi pokretani preko interpretera. Razlika je u tome što je *bytecode* efikasniji od direktnog interpretiranja koda. Kod slučaja *bytecode*-a interpreter interpretira efikasan kod koji je baš namijenjen za njega (interpretera), kod slučaja normalnog

koda interpreter pretvara ljudski kod koji nije dobro optimiziran za direktno prevođenje.

1.2.2. Programski jezici po paradigmi

Programska paradigma je stil to jest "način" na koji programiramo pomoću određenog programskog jezika. Bilo bi krivo reći programska jezična paradigma, jer je paradigma samo način na koji nešto radimo (kao programiranje), a ne nešto konkretno (kao jezik). Uobičajeno je u govoru reći npr. "Haskell je funkcionalni programski jezik", ali to ne znači da postoji nešto takvo kao funkcionalna programska jezična paradigma. Neke od poznatih paradigmi programiranja su:

- Funkcionalna: Kod funkcionalnog programiranja funkcije se smatraju kao objekte prve klase. Drugim riječima, možete prosljeđivati funkcije kao argument drugim funkcijama, ili funkcije mogu vraćati drugu funkciju. Funkcionalna paradigma bazirana je na lambda kalkulusu. Jedan od primjera programskih jezika u kojima je lako koristiti funkcionalnu paradigmu su: LISP, Scheme, Haskell, itd.
- Imperativna: Imperativno programiranje označava davanje instrukcija računalu tako da objašnjavamo kako se nešto računa, korak po korak. Svaki korak ima utjecaj na globalno stanje računanja. Kada se kod piše na imperativni način, njegov prijevod na strojni jezik je optimiziraniji i efikasniji. Naravno jezici u kojima se koristi imperativna paradigma koriste i funkcionalne značajke kao npr. osnovna aritmetika (zbrajanje, oduzimanje, itd.).
- Logička: Logičko programiranje koristi relacije koje specificiraju činjenice i pravila nad kojima se vrše upiti to jest specificiraju ciljevi. Logika ima značajno mjesto u računarstvu i logika je glavna baza dizajna svih logičkih sklopova koje čine računalo.
- Objektno-orijentirana: Fokus je na objektima koje program predstavlja i na dozvoljavanju ponašanja tih objekta. Objekti pripadaju klasama. Tipično je da svi objekti u nekoj klasi imaju isti tip ponašanja. Klase se mogu organizirati hijerarhijski pomoću relacija.
- Deklarativna: U deklarativnom stilu programiranja definiramo to jest opisujemo kakav rezultat želimo, a ne kako doći do njega. Glavni primjer dekorativnog jezika je SQL.

1.2.3. Programski jezici po njihovoj namjeni

Većina programskih jezika su dizajnirani da budu jezici opće namjene. To generalno uključuje jezike poput Java, C, C++, Python i drugi. Neki jezici poput PHP, C# sa ASP.NET, JSP su najviše dizajnirani i namijenjeni za web. Imaju specifične biblioteke namijenjene za manipulaciju HTML-a, XML-a, primanje web formi, slanje web stranica, primanje zahtjeva za određene web stranice, itd. Postoje mnogo programskih jezika koju su znanstveno namijenjeni, kao na primjer R koji se koristi ekskluzivno u statističkim analizama. SQL kao deklarativni jezik smatra se jezikom namijenjenim za rad s bazom podataka i to uključuje kreiranje tablica, dohvaćanje i ubacivanje podataka, izvršavanja upita, itd.

Dolazimo da programskih jezika namijenjenih učenju, koji po svojoj namjeni imaju cilj olakšati učenje programiranja tako da budu sintaksno jednostavniji i lakši za shvaćanje. Neki od takvih jezika su LOGO, BASIC, Pascal, Scratch i drugi.

1.3. Povijest nastanka programskih jezika namijenjenih učenju

BASIC je programski jezik izumljen od strane John G. Kemeny i Thomas E. Kurtz. BASIC je svoje prve početke imao u 1964. godini. U to vrijeme dva profesora matematike čvrsto su vjerovala da će informatička pismenost biti esencijalna i napravili su BASIC koji je informatičku pismenost činio lakše ostvarivom. U 70-ima i ranim 80-ima dolazi do veće pojave osobnih računala i BASIC se pobrinuo da ta računala budu više korisnija. Za vrijeme eksplozivnog rasta korištenja, nastalo je mnogo BASIC varijacija, neki od njih su: Tiny BASIC, Altair BASIC, Microsoft BASIC, BBC BASIC, itd. U kasnijim godinama nastali su jedno od poznatijh verzija BASICA, a to su: QBasic, AmigaBASIC, PureBasic, True BASIC, Small Basic, Visual Basic koji je u kasnijim verzijama postao koristan za pisanje malih poslovnih aplikacija, a kasnije se razvio u Visual Basic .NET.

1967. godine, Seymour Papert, matematičar koji je radio na odjelu MIT-a, napravio je s ostatkom tima prvu verziju programskog jezika Logo. Logo programski jezik, koji je dijalekt od programskog jezika Lisp i dizajniran je kao alat za učenje. Njegova široka uporaba počela je s pojavom mikro-procesora, to jest pojavom osobnog računala 1977. pojavljuje se Terrapin Software, kompanija koja je započela distribuciju fizičkog robota u obliku kornjače, koja se još danas prodaje. Kornjača je popularno Logo okruženje i njom se upravlja Logo jezikom, u grafičkom ili fizičkom obliku. Tvorac jezika Logo 1980. objavljuje publikaciju pod nazivom Mindstorms, u kojoj se opisuje kako djeca (osnovna i srednjoškolska dob) mogu naučiti o informatičkoj pismenosti. Danas postoje više od 300 verzija Logo programskog jezika.

Niklaus Wirth 1971. predstavlja specifikaciju za novi visoko strukturirani programski jezik Pascal. Pascal je jezik orijentiran podacima i nudio je programerima mogućnost definiranja vlastitih tipova podataka. S tom slobodom dolazi to potrebe striktnog provjeravanja tipova, koji sprječava miješanje tipova podataka. Pascal je bio namijenjen kao jezik namijenjen učenju i kao takav bio je široko prihvaćen. Još veću popularnost u ranim 80-ima možemo objasniti i s odlukama SAD-a, koji su u svoje prijemne ispite za srednjoškolce odabrali Pascal kao dio ispita. Popularnost Pascala pridodaje inačica kompajlera Turbo Pascal, koji je u svoje vrijeme bio revolucionaran zbog svoje brzine kompajliranja, a kasnije objektnog programiranja.

2. PROGRAMSKI JEZICI NAMJENJENI UČENJU

Osnovni cilj svakog programskog jezika koji je namijenjen učenju programiranja mora biti usvajanje osnovnih znanja, tj. koncepata programiranja koji se mogu primijeniti u ostalim programskim jezicima, a ne samo u jeziku namijenjenom učenju programiranja. Kod odabira ovakvih programski jezika bitan je naglasak na pedagoški aspekt učenja programiranja. Jedan od glavnih kriterija odabira programskog jezika predloženo je od strana kreatora programskih jezika: Seymoura Paperta (LOGO), Niklausa Wirtha (Pascal), Guida van Rossuma (Python) i Bertranda Meyera (Eiffel). Kriteriji se odnose na programske jezike kao cjelina i bez odnosa na njihovu paradigmu. Kriterije provjeravaju da li programski jezik (Bubica, 2015):

- Pogodan za nastavu,
- nudi opći okvir,
- promovira novi pristup učenju programiranja,
- promovira pisanje ispravnih programa,
- dopušta rješavanje problema koji se mogu riješiti u malim dijelovima koda (modularnost),
- nudi besprijekorna i jednostavna razvojna okruženja,
- podržan na različitim okruženjima,
- ima korisničku podršku kroz zajednice,
- otvorenog koda,
- ima prateći adekvatni nastavni materijal,
- ne koristi se samo u obrazovanju.

Ove kriterije olakšale su odabir programskog jezika za učenje. U radu "*An Objective Comparison of Languages for Teaching Introductory Programming*" uspoređivali su se programski jezici poput: C, C++, Eiffel, Haskell, Java, Javascript, Logo, Pascal, Python, Scheme i VB. Rezultat istraživanja bio je da su najprikladniji jezici za učenje programiranja Python i Eiffel. Valja napomenuti da je u ovoj analizi nedostatak to što se provjeravao je li je kriterij

ispunjen (da ili ne), a ne u kojoj mjeri. Prema ispunjenosti kriterija Java je zauzela treće mjesto koja je prvenstveno namijenjena za poduzeća i komercijalni *software* što je ujedno i glavna mana.

3. PROGRAMSKI JEZIK LOGO

Logo je originalno bio konstruiran kao programski jezik koji bi uveo djecu svijet programiranja. Cilj je bio razviti bolje vještine razmišljanja, koje se mogu koristiti u drugim kontekstima a ne samo u programiranju. Logo je obično smatran kao računarsko okruženje namjenjeno djeci s naznakom na računalnu grafiku. Ali on je puno više od toga. Logo je programski jezik širokog opsega i mogu ga koristiti učenici raznih godina na razne načine. Vodeći princip razvoja Loga bio je da ima *low treshold*[15] i *high ceiling*[16]. Što bi značilo da je ulazak u Logo svijet lagan i bez prepreka, ali jednom kad ste unutra, Logo vam pruža mogućnost kretanja od jednostavnih malih programa pa sve do kompleksnih projekta. Još jedan od principa razvoja bio je da Logo jezik ima *wide walls* što omogućava ljudima različitih interesa, ukusa, načina učenja da imaju ugodan razvitak u radu s Logom. Od kad je Logo projekt započet, njegov se svijet povećao. Prag se spustio, strop se povisio, a zidovi su postali širi.

U prošlosti najpopularnija verzija Loga bila je ona koja je u sebi imala komponentu geometrije kornjače. Djeca su koristila geometriju kornjače kako bi istraživali geometrijske oblike i pisali procedure koje su te oblike ispisivale. Logo je djeci omogućavao pisanje jednostavnih dizajna za crtanje oblika, koji su se mogli kombinirati za pisanje kompleksnih dizajna to jest crtanje kompleksnih oblika.

Geometrija kornjače je samo jedna domena Logo projekta. U ranim fazama razvitka, Logo jezik koristio se i u područjima glazbe, robotike i govornog jezika. U početku Logo nije ni imao kornjaču. Ime Logo što znači "riječ" na Grčkom, razlog odabira imena je bio naglasak na to da je Logo programski jezik namijenjen radu s riječima i rečenicama u odnosu na numerički fokus koji su imala većina programskih jezika tog vremena.

Animacija i igrice jedno je od grana Logo verzija. U ovim verzijama, Logo podržava korištenje višestrukih kornjača koje su mogle koristiti *sprites*[17], to jest različite kostime koji su mogli biti

15 Niski strop
16 Visoki strop
17 Kostimi

animirani. Ova funkcionalnost je standard u mnogim današnjim verzijama Loga, najpopularnija verzija ovakvog tipa je Scratch. Scratch omogućava unošenje različitih medija poput: slika, zvuka, videa i glazbe.

U 1980-ima dolazi do primjene Loga u robotici. Neke verzije Loga imaju mogućnost primanja ulaznih informacija iz senzora. Senzori omogućavaju programiranje fizičkog robota koji će primati podatke iz stvarnog svijeta i na njih reagirati ovisno o programu. Primjena Loga u robotici motivira učenika da nastavi učiti programiranje zato što je taj način programiranja zanimljiviji.

3.1. Osnove jezika LOGO

Za razliku od većinu drugih programskih jezika, Logo skoro u potpunosti ima uniformnu sintaksu. To znači da se sve različite naredbe koje Logo razumije, predstavljene na isti način, to jest istom notacijom: ime procedure je popraćeno s ulaznim podacima, koji mogu biti konstante ili neki rezultat dobiven nekom drugom procedurom. U Logo programskom jeziku procedura *print* uvijek prima točno jedan ulazni podatak. Ulazni podatak može biti riječ ili lista.

Za početnike programere lako je pogriješiti s idejama ulaznih i izlaznih podataka. U Logu izlazni podaci nisu nešto što program uvijek ispisuje na ekran, kao što ulazni podaci nisu uvijek nešto što je upisano preko tipkovnice. Umjesto toga, ove ideje zamišljaju ulazne i izlazne podatke kao objekte (riječi i liste) koje su poslane kao parametar procedure (ulazni podatak) ili kao rezultat procedure (izlazni podatak). Svaka procedura ima određeni fiksni broj ulaznih podataka i izlaznih podataka. Broj ulaznih podataka može biti nula ili više, gdje broj izlaznih podataka može biti nula ili jedan. Procedura s izlaznim podacima obično zovemo operacije, a one bez, naredbama. Neke operacije imaju samo dva moguća izlaza: *true* ili *false*, tj. istinito (1) ili lažno (0). Ovakve procedure zovemo predikatima. Predikat se koristi kako bi program mogao izvršiti određeni dio koda samo ako je određeni uvjet zadovoljen. Po konvenciji predikati imaju zadnje slovo u nazivu procedure, slovo p. Na primjer, predikat *numberp* vraća 1, to jest istinu, samo ako je ulazni podatak broj. U slučaju da je ulazni parametar poslan kao riječ (npr. *sedam)*, a ne kao broj (7), predikat vraća 0 jer sama riječ *sedam* nije broj u Logo programskom jeziku. Kada bi poslali 7 kao parametar, odgovor bi bio 1.

U Logu brojevi su posebna vrsta riječi, u kojemu su svi znakovi brojevi. Nije potrebno stavljati navodni znak prije pisanja brojeva. Kako su brojevi vrsta riječi u Logu, moguće je primijeniti procedure koje vrijede za manipulaciju riječi. Ispis broja pomoću procedure *first* vraća prvu znamenku broja. Brojeve je moguće koristi u kombinaciji s aritmetikom.

Riječ u Logu možemo smatrati kao lista simbola to jest slova. Osim što je moguće manipulirati riječima kao skupom simbola i slova, moguće je manipulirati listom riječi, listom drugih podlista, itd. Ovo omogućava korištenje kompleksnih struktura podataka na lakši način. Korištenje liste u Logu označava se pomoću uglatih zagrada oko liste, a elementi u listi odvojeni su razmakom.

```
1  TO COUNTBGTHREE
2    MAKE "brojac 0
3    REPEAT 5 [
4      MAKE "broj READWORD
5      IF :broj > 3 [MAKE "brojac :brojac + 1]
6    ]
7
8    PRINT :brojac
9  END
```

Slika 1: Primjer programa u Logu

Za definiranje varijabli koristi se naredba *MAKE* koja za ulaz prima naziv same varijable i Logo objekt (na primjer riječ, broj ili procedura). Rezultat naredbe je da se drugi ulazni podatak pridodaje varijabli nazvanoj po prvom ulaznom podatku koja je riječ. U primjeru na slici 1 pridodajemo broj 0, varijabli pod nazivom *BROJAC*. Varijablu *BROJAC* pozivamo pomoću notacije *:imeVarijable* i predajemo vrijednost proceduri *PRINT*.

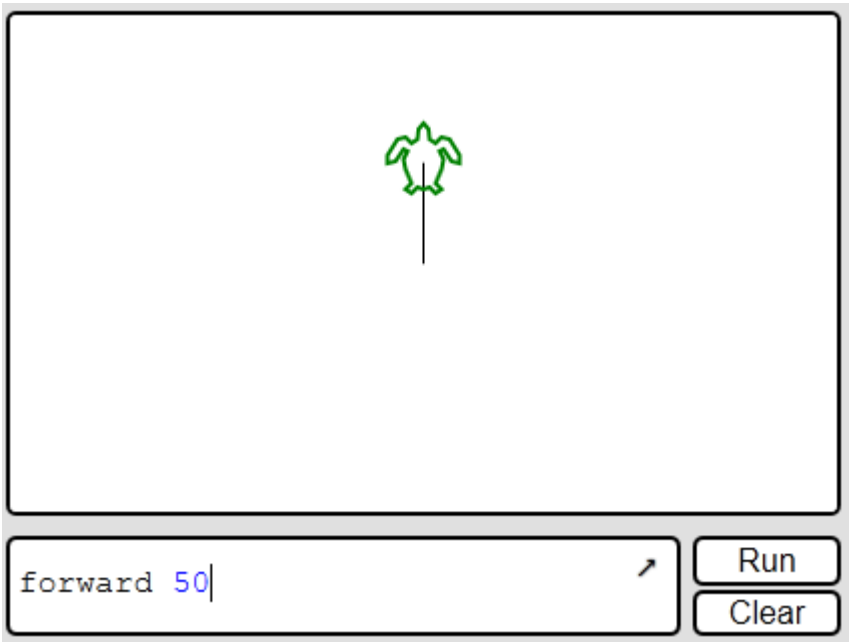
Za definiranje procedure koriste se ključne riječi *TO* i *END*. Kod definiranja procedura potrebno je navesti naziv procedure i ulazne podatke ako ih ima.

3.2. Geometrija kornjače

Logo je najviše poznat zbog toga što je uveo kornjaču kao alat za računalnu grafiku. Mnogima je Logo i geometrija kornjače sinonim. Većina računalnih kompanija koristila je Logo

kao proizvod koji je pružao samo kornjaču kao alat za manipulaciju računalne grafike. Ideja da je Logo u glavnom bio zamišljen kao grafika kornjače je pogreška. Kao što je navedeno prije, naziv Logo nastao je zbog grčke riječi *Logos* što znači riječ, i Logo je bio zamišljen kao jezik s naglaskom na radu s riječima. Ipak, kornjača je ostala kao jedan od velikih značaja Logo jezika.

Glavna ideja kornjače je da kornjača ima jedan smjer i da se kornjača može kretati u tom smjeru. Može se kretati naprijed i nazad, ali ne u stranu. Ako želimo da se kornjača kreće u nekom drugom smjeru moramo joj prvo promijeniti smjer. Primarni način kretanja kornjače je pomoću naredbe *forward,* ili skraćeno *fd*. *Forward* za ulaz prima jedan podatak, i on mora biti tipa broj. Efekt naredbe *forward* je da se kornjača kreće naprijed u smjeru kojem je postavljena, a duljina kretanja definirana je ulazom. Jedinica kretanja kornjače nazvana je "*turtle step*".



Slika 2: Primjer naredbe forward u Logu

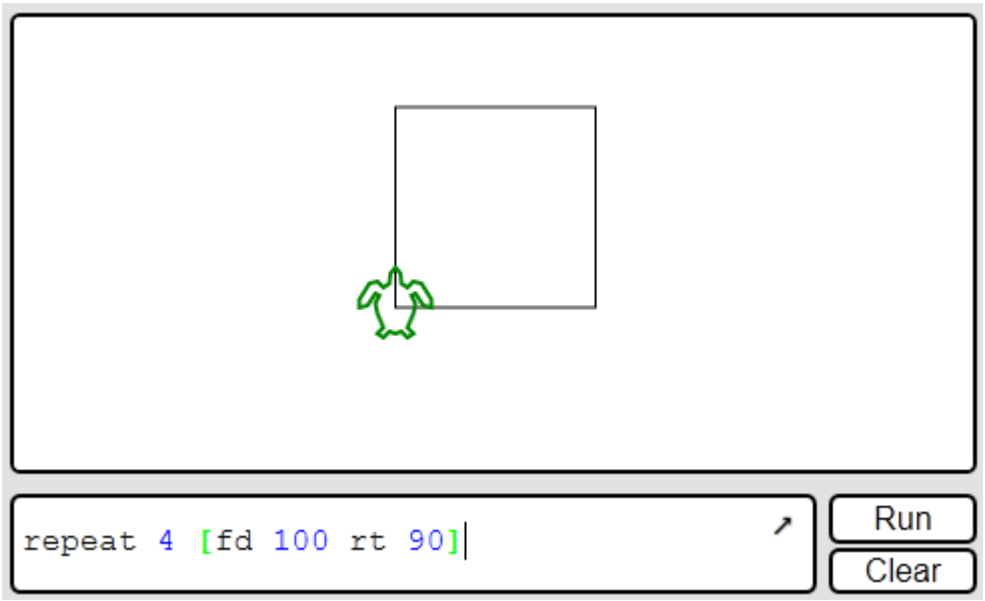
Na slici 8 prikazano je kretanje kornjače unaprijed za 50 koraka. Kornjača za sobom ostavlja trag kao da je prati olovka. Moguće je "podignuti olovku" naredbom *penup* to jest *pu.* Nakon *pu* naredbe kornjača više ne ostavlja trag. Suprotno od naredbe *pu* je naredba *pendown* to jest *pd.*

Neke od naredbi za upravljanje kretnjom kornjačom, a nisu navedene, su:

- Naredba *back:* pokreće kornjaču unazad za određeni broj koraka,

- *left:* rotira kornjaču u lijevo za određeni broj stupnja,
- *right:* rotira kornjaču u desno za određeni broj stupnja,
- *setpos:* postavlja koordinate kornjače opisane ulaznim podacima *x* i *y,*
- *home:* postavlja kornjaču u centar, usmjerena prema gore.

U Logu često se koristi naredba *repeat* koja prima dva ulazna podatka. Prvi podatak je pozitivan broj, a drugi je lista instrukcija. Rezultat je izvršavanje instrukcija iz liste instrukcija onoliko puta koliko je definirano u prvoj prvom ulaznom podatku.

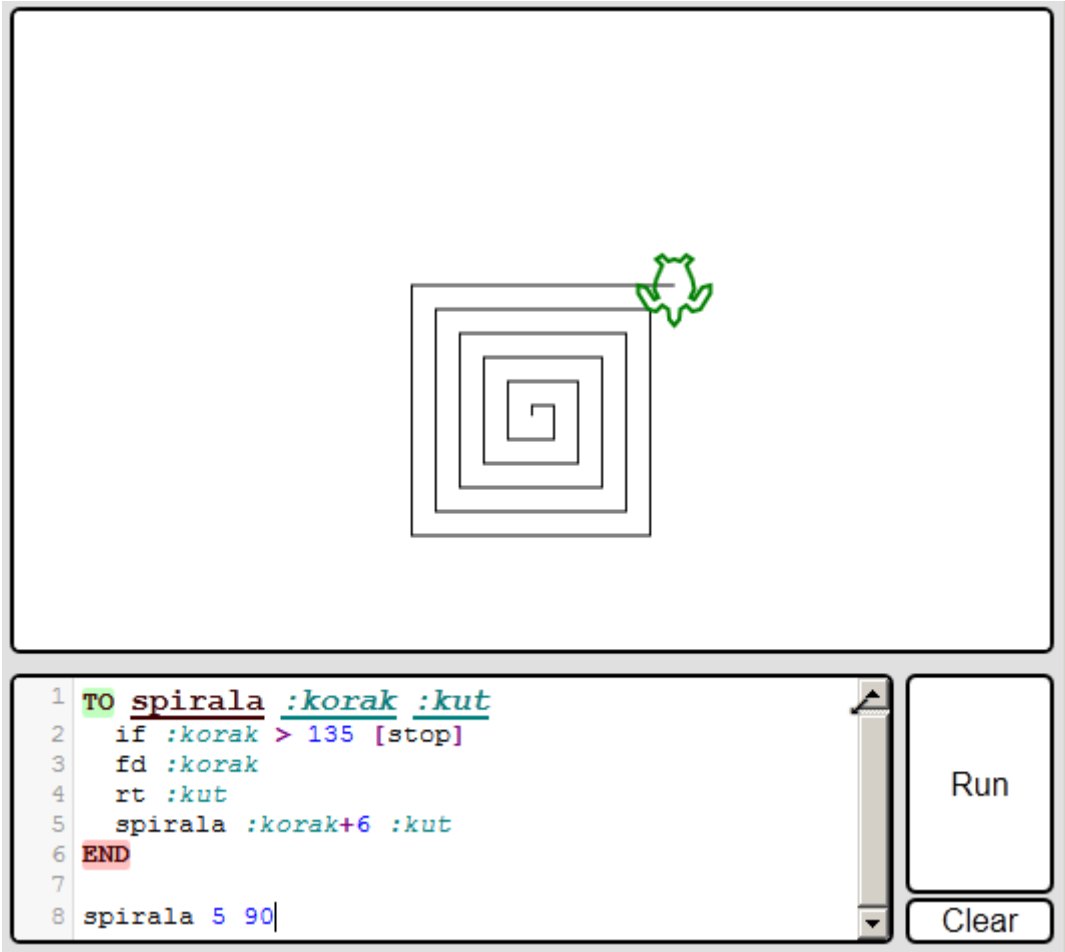


Slika 3: Primjer korištenja naredbe repeat u Logu

Na slici 9 koristi se naredba *repeat*, u kojoj je specificirano da se naredbe *fd 100 rt 90* ponavljaju 4 puta. Rezultat je ponavljanje kretanje kornjače naprijed za 100 koraka, zaokret kornjače u desno za 90 stupnjeva (pravi kut). Logo nam omogućava još mogućnosti s ovim naredbama:

- Naredba *fill:* ispunjava bojom lik u kojem se kornjača nalazi,
- *setpencolor:* mjenja boju olovke kornjače,
- *setbackground:* mjenja boju pozadinske površine,
- *clearscreen:* briše ekran i postavlja kornjaču u centar usmjerena prema gore.

Rekurzija nije strani pojam programskom jeziku Logo. U slici 10 napisana je procedura koja prima dva ulazna podatka. Prvi ulaz je količina koraka, a drugi je stupanj promjene smjera. Pozivanjem procedure *spirala* s parametrima 5 i 90 dobivamo pravokutnu spiralu. Na kraju procedure definirana je linija koda *spirala :korak+6 :kut* koja poziva istu proceduru ali s različitim prvim parametrom. Parametar *korak* je svaki put povećan za 6, što nakon nekog vremena rezultira da uvjet *if :korak > 135* bude istinit, a time se zaustavlja procedura.



Slika 4: Primjer korištenja rekurzije u Logu

Ako pozovemo proceduru spirala s parametrima 5 i 60, dobivamo drugačiju spiralu, to jest spiralu s manjim kutovima.

4. PROGRAMSKI JEZIK BASIC

BASIC je skraćeno: *Beginner's all-purpose symbolic instruction code*. Proceduralni jezik BASIC nastao je u 1964. godini na Dartmouth sveučilištu. Tvorci jezika imali su ideju napisati jezik koji bi se koristio kao alat za učenje programiranja. Dizajn je baziran na osam principa:

- Učenje jezika je lako za početnike,
- univerzalno prihvatljiv
- proširivost jezika,
- interaktivnost,
- Kratko vrijeme odaziva
- neovisnost od hardwarea,
- neovisnost od operativnih sustava,
- razumljive poruke o greškama.

BASIC je u prošlosti bio jedan od najčešće korištenih programskih jezika. U vrijeme kad je BASIC bio popularan, smatrao se jednostavnim korakom za studente koji će taj jezik naučiti prije učenja FORTRANA, koji je bio kompliciraniji za programere početnike. BASIC-ova prednost bile su njegove engleske naredbe, koje su bile lakše za shvatiti i zapamtit, za razliku od ostalih programskih jezika.

Programski jezik BASIC bio je prvi jezik dostupan osobnim računalima. Programi napisani BASIC-om imali su reputaciju da su bili spori, što je bilo istina u tom dobu osobnih računala. Glavni razlog sporosti bila je činjenica da je BASIC u to vrijeme bio interpretirani jezik. Pa se interpretiranje koda izvršavalo tijekom izvođenja programa, linija po liniju. Naravno današnje moderne verzije BASIC su kompajlirane, pa ujedno i brže tijekom izvođenja.

Najpoznatije verzije BASIC-a koje se koriste u svrhu učenja programiranja u školama su QBASIC i Microsoftov Visual Basic koji se više razlikuje od BASIC-a po tome što ima mogućnost programiranja grafičkog sučelja, i posjeduje objektno orijentirane značajke.

Osnove jezika BASIC

BASIC ima jednostavnu sintaksu i koristi mali broj ključnih riječi. Prva naredba koji učenici nauče je *print*. *Print* naredba ispisuje poruku koja se nalazi nakon ključne riječi. Poruka koja je fiksna mora biti napisana s navodnim znacima na početku i na kraju poruke. *CLS* naredba znači *Clear Screen,* u prijevodu briši ekran, rezultat naredbe je lako zaključiti. Upit za unos podatka moguće je vršiti pomoću naredbe *input*. Za definiranje polja koristi se ključna riječ *dim,* koja uz sebe ima naziv polja i veličinu. Veličina polja mora biti definirana cijelim brojem. Kontrolu toka moguće je uz sljedeće jezične konstrukcije:

- Naredba *if ... then ... else*: evaluira se uvjet, ako je uvjet točan, nastavlja se kod nakon *then*, inače kod nakon *else,*
- *for ... to .. {brojKoraka} ... next:* ponavlja se dio koda onoliko puta koliko je to definirano, moguće je koristiti varijablu koje se koristi kao brojač,
- *repeat ... until:* dio koda se ponavlja, sve dok specificiran uvjet nije istinit,
- *goto*: skok na označeno mjesto (broj linije ili posebno definiran naziv u kodu),
- *gosub:* skok na označeno mjesto, izvršava se kod sve do ključne riječi *return*, nakon koje se tok egzekucije vraća na liniju nakon *gosub*.

```
1  10 LET BROJAC = 0
2  20 FOR I=1 TO 5
3  30 INPUT BROJ
4  40 IF BROJ > 3 THEN BROJAC = BROJAC + 1
5  50 NEXT I
6  60 PRINT BROJAC
7  70 END
```

Slika 5: Primjer programa u BASIC-u

Primjer na slici 5 prikazuje stvarne linije koda i linije koda namijenjene BASIC-u (10, 20, 30, 40, itd.). Linija koda 10 prikazuje dodjeljivanje broja 0 varijabli po nazivom *BROJAC* pomoću ključne riječi *LET.* Naredba *FOR I=1 to 5* ponavlja naredbe napisane do linije *NEXT I*. Primjer demonstrira kontroliranje toka koda pomoću ključnih riječi *if ... then* na liniji 4. Program nakon petlje ispisuje varijablu *BROJAC* i završava program naredbom END.

U BASIC programskom jeziku integrirane su matematičke i statističke funkcije. Statističke funkcije za ulazne podatke primaju jedan ili više argumenta, i uvijek samo numeričke vrijednosti. Neke od matematičkih funkcija su:

- Funkcija *avg:* aritmetička sredina,
- *max*: najveća vrijednost,
- *min*: najmanja vrijednost,
- *std*: standardna devijacija,
- *sum*: zbroj,
- *abs:* apsolutna vrijednost broja,
- *acs, asn, atn, cos, sin, tan:* ark kosinus, ark sinus, ark tangenta, kosinus, sinus, tangenta,
- *exp:* eksponencijalna funkcija.

5. PROGRAMSKI JEZIK PASCAL

Pascal je *high-level* proceduralni programski jezik, koji je imao veliku ulogu u učenju osnovnih koncepta programiranja. Pascal je odličan uvod u proceduralne jezike i pomaže kod prelaska na teže programske jezike tog tipa, kao na primjer C programski jezik.

Programski jezik Pascal posjeduje značajne jezične značajke koje omogućavaju korištenje jezika kao moćan alat za uvođenje učenika u tehnike strukturiranog programiranja. U jeziku su definirane primarni tipovi podataka kao što su: *Integer* – cijelo brojni brojevi, *Real* – realni brojevi, *Character* – jedan znak, *String* – grupa znakova, *Boolean* – logički tip podataka. Pascal omogućava definiranje različitih struktura podataka, poput *arrays* – polja, *records* – kolekcija različitih tipova podataka, *files* – manipulacija datoteka, *sets* – skup vrijednosti iz enumeracije. Kompilacija napisanog programa ukazuje na krivo korištenje tipova podatka, što znači da je Pascal striktan kod provjeravanja dodjeljivanja vrijednosti tipovima podatka, to jest varijablama. Programski jezik Pascal jednostavan je i ekspresivan, i to ga čini efektivnim za učenje tehnika programiranja.

Osnove jezika Pascal

Pascal je striktan kod provjeravanja tipova podataka i koristi blokovne strukture u programiranju. Tipovi varijabli definirani su pomoću imena, eksplicitnom veličinom opsega i primitivnim tipom. Kada su varijable deklarirane kao neki određeni tip, kompajler pretpostavlja da će njegov tip ostati isti tijekom njegovog života u programu. Ovo konzistentno korištenje varijabli, čini kod lakšim za održavanje.

Program napisan u Pascalu obično posjeduje sljedeće dijelove koda:

- Naziv programa,
- kolekcija naziva biblioteka, ako ih program koristi,
- deklaracija tipova, konstanti, varijabli, funkcija, procedura,
- glavni blok programa zajedno s izrazima i ekspresijama,

- komentari.

```
1  program countBgThree;
2  var
3   brojac : integer = 0;
4   broj : integer;
5   i : integer;
6
7  begin
8      for i:=1 to 5 do begin
9          readln(broj);
10         if broj > 3 then
11             brojac := brojac + 1;
12     end;
13
14     writeln(brojac);
15 end.
```

Slika 6: Primjer programa u Pascalu

U primjeru na slici 6 prikazana je struktura programa. Prva linija koda ukazuje na naziv programa. Svaki Pascal program mora imati definirani naziv. Na liniji 7 i 15 vidimo kako je blok programa određen. Početak se označava s ključnom riječi *begin,* a kraj sa *end* i točkom na kraju. Linija 14 sadrži naredbu *writeln* koja na ekran ispisuje sadržaj napisan u zagradama. Korištenje petlje možemo vidjeti na linijama 8 do 12. Na liniji broj 8 koristi se naredba *for i:1 to 5 begin* koja ukazuje da će se petlja izvršiti 5 puta. Blok petlje definiran je ključnim riječima *begin* i *end*. Ključna riječ za ulazni podatak je *readln* i ona za parametar prima varijablu u koju se ulazni podatak sprema. Na liniji 10 i 11 koristi se *if* naredba za kontrolu toka programa, zajedno s naredbom za povećanje varijable *brojac* za 1, u slučaju da je uvjet poslije naredbe *if* istinit. Za logičko odlučivanje i kontrolu toka, Pascal koristi sljedeće naredbe:

- Naredba *if ... then ... else:* evaluira se uvjet, ako je uvjet točan, nastavlja se kod nakon *then*, inače kod nakon *else* (*else* je neobavezan),
- *case ... else:* evaluira jednakost varijable sa svim definiranim slučajevima, ako je jednakost nije točna nastavlja se kod nakon *else* (*else* je neobavezan).

Korištenje petlji moguće je sa:

- Naredba *while ... do:* ponavlja izraz u bloku sve dok je uvjet točan, uvjet se testira prije izvršavanja bloka,

- *for ... do:* ponavlja izraz u bloku s dodatkom pomoćnog iteratora,
- *repeat ... until:* isto kao *while ... do,* samo što se uvjet ispituje poslije bloka.

Potprogram je program koji radi određenu stvar, oni se kombiniraju kako bi formirali veći program. Ova opcija slaganja potprograma zove se modularni dizajn. Pascal nudi dvije opcije potprograma:

- Funkcije: potprogrami koji vraćaju vrijednost,
- procedure: potprogrami koji ne vraćaju vrijednost.

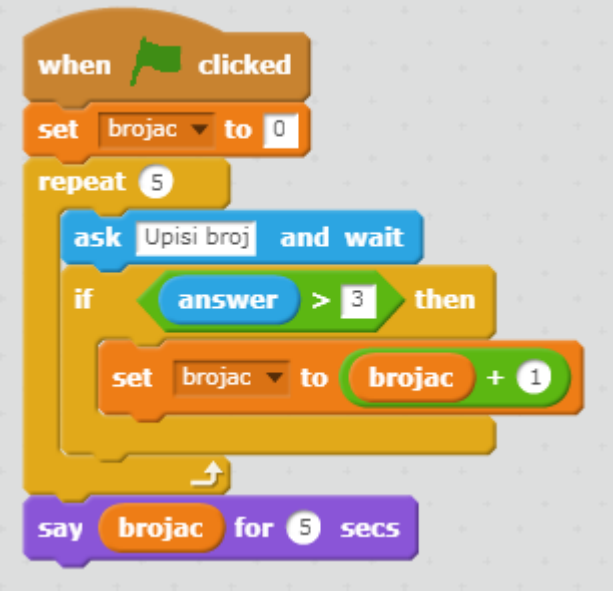
Funkcija je grupa izraza koja zajedno izvode zadatak. Funkcije se trebaju deklarirati i definirati. Deklaracija govori kompajleru o nazivu funkcije, njegovom tipu i parametrima koje prima. Definicija funkcije govori ono što se izvršava. Za definiranje funkcije koristi se ključna riječ *function.*

6. PROGRAMSKI JEZIK SCRATCH

Scratch je novi programski jezik otvorenog koda razvijen 2003. godine na Tehnološkom institutu države Massachusetts (*Massachusetts Institute of Technology* – MIT) kao projekt Lifelong Kindergarten grupe, a omogućuje lako kreiranje interaktivnih priča, igrica, animacija i projekata te njihovo dijeljenje s drugim korisnicima preko web-a. (Mujačić, 2015). Grupa Lifelong Kindergarten surađivala je s tvrtkom Lego na izradi robota Lego Mindstorms koji se, također, koriste za poučavanje programiranja. Uočilo se da djeci u radu s kockicama odmah počinju navirati ideje, mašta i kreativnost pa su napravili vizualni programski jezik koji podsjeća na slaganje kockica. Naredbe su napravljene u obliku slagalica, tako da je vizualno jasno koje se naredbe mogu složiti. Grupirane su tematski, a razlikuju se po obliku i bojama (Bubica i sur., 2014). Iako je nastao na idejama programa LOGO, programiranje u Scratchu razlikuje se od programiranja u drugim vizualnim programskim okolinama, jer se u njemu rabe naredbene strukture u obliku grafičkih programskih blokova, pa se na taj način eliminira mogućnost sintaksnih pogrešaka (Peppler, Kafai, 2005). Program omogućuje programiranje mišem povlačenjem i uklapanjem blokova koji se mogu spojiti samo ako to odgovara u određenom sintaksnom smislu i tako omogućuje učenicima da se fokusiraju na probleme koje oni žele riješiti. (Đurđević, 2013)

Osnove jezika Scratch

Scratch projekt čine objekti koji se zovu likovi (sprites). Njihov se izgled može mijenjati dodavanjem različitih kostima (costumes). Lik može izgledati poput osobe, vlaka, leptira ili bilo čega drugoga. Za kostim je moguće koristiti bilo koju sliku; izraditi crtež u Paint Editor-u, može se učitati slika s računala ili metodom uhvati-povuci-pusti ('drag and drop') preuzeti slika s internetske stranice. Liku se mogu dati različite upute: da se kreće, razgovara, reproducira glazbu ili ostvaruje interakciju s drugim likovima. Za izdavanje naredbi potrebno je složiti grafičke blokove u cjeline, nazvane skripte. (Otvoreno društvo za razmjenu ideja (ODRAZI), 2011). Scratch blokovi organizirani su u osam kategorija različitih boja: Kretanje, Izgled, Zvuk, Olovka, Upravljanje, Očitanja, Operacije i Varijable. Svaka vrsta blokova ima jedinstven izgled kojim pokazuje koji blok ide ispod, iznad ili unutar tako da se blokovi spajaju kao slagalice.



Slika 7: Primjer programa u Scratchu

Svojim izgledom blokovi sugeriraju koja su ograničenja i mogućnosti te tako onemogućavaju javljanje pogrešaka. Iako nema dojave pogrešaka, sam program može funkcionirati na "krivi način", ali eksperimentiranjem i iskustvom može se doći do željenih rezultata. Nakon pokretanja skripte, Scratch prolazi kroz blokove i izvršava naredbe od vrha do dna. Ako korisnik samo želi provjeriti kako djeluje određeni blok, dovoljno je kliknuti na njega i lik na ekranu će naredbu odmah izvesti. Skripte koje se grade se uvijek mogu izvest, iako nisu gotove ni potpune, te pojedini blokovi neće smetati ako se odvoje van skripte. (Valić i sur., 2013)

7. PROGRAMSKI JEZIK C++

C++ je programski jezik opće namijenjene, razvio ga je Bjarne Stroustrup i ekstenzija je programskog jezika C. Inicijalni naziv jezika je bio "*C s klasama*", jer je imao sva svojstva jezika C zajedno sa mogućnostima i konceptima klasa. Poboljšanja nad jezikom C nastavila su se sa značajkama poput: virtualne funkcije, preopterećenje operatora, višestruko nasljeđivanje, predlošci, korištenje izuzetaka. C++ je jedan od najpopularnijih programskih jezika koji se u velikoj mjeri koristi kod sistemskih i aplikacijskih softvera, drivera, embedded sustavima i većinom u softverima koji su kompleksnije strukture i koji zahtijevaju brzinu izvođenja. C++ je standardiziran i ratificiran 1998. godine pod nazivom *ISO/IEC 14882:1998*, a posljednja verzija je *ISO/IEC 14882:2014* izdana 2014. godine.

U knjizi *The Design and Evolution of C++,* Bjarne Stroustrup navodi pravila i principe koja su praćena kod dizajniranja programskog jezika C++:

- C++ je dizajniran da bude jezik opće namijene i da je efikasan i prenosiv kao jezik C,
- C++ pruža mogućnost korištenja višestrukih stilova programiranja (proceduralno programiranje, apstrakcija podataka, objektno orijentirano programiranje i generičko programiranje),
- C++ je dizajniran tako da je kompatibilan s programskim jezikom C koliko je to moguće,
- C++ izbjegava implementiranje značajki koje su specifične za neku platformu ili nisu opće namijene
- C++ je dizajniran kako bi nudio programerima izbor načina korištenja značajki jezika, jer restrikcije u većini slučajeva neće spriječiti programera da nešto radi na svoj način,
- C++ treba izbjeći korištenje značajki koje su nepotrebne i koje pridodaju općim troškovima performansi jezika,
- C++ je dizajniran da bude funkcionalan bez kompleksnih programskih okruženja.

Osnove jezika C++

C++ nasljeđuje većinu svoje sintakse iz programskog jezika C. C++ je hibridni jezik i u njemu je moguće programirati koristeći samo značajke iz C jezika (C stil), koristeći objektno-orijentirani stil, ili oboje. Za pokretanje C++ potrebno je prevesti izvršni kod u izvršni program pomoću kompajlera. C++ je statički pisan jezik, to jest svaki tip bilo kojeg entiteta mora biti prepoznat kompajleru. Svaki C++ program mora imati točno jednu globalnu funkciju nazvanu *main* i svaki program započinje s izvršavanjem te funkcije. *Main* funkcija je tipa *integer* i uobičajeno je da vrijednost koju *main* funkcija vraća je nula. Sve vrijednosti osim nule označavaju greške, koje su namijenjene okruženju u kojem se program izvodi, npr. Linux ili Windows operacijski sustavi. (Stroustrup, Bjarne, 2013.) U C++ je moguće koristiti ključnu riječ *namespace,* koja omogućava grupiranje deklaracija u svojevrsnu grupu. Korištenjem *namespace*-a želi se izbjeći miješanje naziva deklaracija. Po Stroustrupu, na*mespace* najprije služi za organiziranje većih dijelova programa, kao npr. Biblioteka. Skupine izraza i izjava označavaju se unutar blokova, to jest unutar vitičastih zagrada. Svaki nazivi koji su deklarirani unutar vitičastih zagrada, izlaze iz blokovnog djelokruga na kraju vitičastih zagrada.

Za kontrolu toka C++ koristi sljedeće naredbe:

- Naredba *if (...) ... else ... :* evaluira se uvjet unutar zagrada, ako je uvjet točan, nastavlja se kod nakon zagrade, inače kod nakon *else* (*else* je neobavezan),
- *switch (...) ...*: evaluira jednakost varijable sa svim definiranim slučajevima (slučajevi su definirani pomoću ključne riječi *case*), ako jednakosti nisu zadovoljene izvršava se kod poslije ključne riječi *default*.

Programiranje petlji moguće je s naredbama:

- Naredba *while (...) ...:* ponavlja se kod nakon zagrada, sve dok je uvjet u zagradi točan (uvjet se ispituje prije izvršavanja koda),
- *do ... while (...):* ponavlja se kod nakon ključne riječi *do,* sve dok je uvjet u zagradi poslije ključne riječi *while* točan (uvjet se ispituje poslije izvršavanja koda, pa se kod minimalno izvrši jednom),
- *for (... ; ... ; ...) ... : for* petlja definira se pomoću inicijalizacije iteratora (obično nazvan i)

u djelu zagrade do prve točke-zarez, od prve točke-zarez do druge točke-zarez upisuje se uvjet koji odlučuje o prekidu ili nastavku petlje (uvjet obično ispituje iterator), u zadnjem djelu zagrade navodi se izraz koji mijenja iterator.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int brojac = 0;
6
7    for (int i = 0; i < 5; ++i) {
8        int broj;
9        cin >> broj;
10
11       if (broj > 3) ++brojac;
12   }
13
14   cout << brojac << endl;
15   return 0;
16 }
```

Slika 8: Primjer programa u C++-u

U prvoj liniji prikazanoj na slici 8, vidimo korištenje ključne riječi *include* zajedno s nazivom standardne biblioteke *iostream*. Biblioteka *iostream* sadrži kod koji omogućava korištenje ulaznih i izlaznih tokova. Bez naredbe *#include <iostream>*, kompajler "ne zna" za naredbe na linijama 9 i 14 (*cin >>, cout <<*). Program *main* definiran je od linije 4 do linije 16. Na liniji broj 2 vidimo korištenje ključne riječi *using* koja govori kompajleru da koristi namespace *std,* to jest da kod zadrži dijelove koda iz namespace-a *std* (standardna biblioteka). Bez linije 2, programer bi trebao pisati liniju 9 pomoću oznake *std:: ,* to jest *std::cin >> broj;*. U petlji vidimo način deklariranja varijable tipa *int*, ulazni tok koji sprema ulaz u varijablu *broj* pomoću naredbe *cin >>* i uvjet koji povećava brojač ovisno o točnosti uvjeta. Za ispis brojača koristi se naredba *cout << broj << endl;* koja ujedno i vraća liniju u novi redak pomoću naredbe *endl* (skraćeno od *end line*).

8. PROGRAMSKI JEZIK JAVA

Razvoj programskog jezika Java započela je tvrtka *Sun Microsystems* pod vodstvom Jamesa Goslinga. Prva verzija Jave izlazi u 1995. godini. Java je objektno-orijentirani jezik u kojemu svaki program posjeduje najmanje jednu klasu. Programski jezik Java poznat je po tome što ne ovisi o platformi na kojoj se izvodi. Kod napisan u Javi može se izvršavati na svakom okruženju koji posjeduje *Java Virtual Machine*. Za razvoj programa napisanih u Javi potrebno je imati postavljenu Java razvojnu opremu (engl. Java Development Kit).

Po Oracle-u temeljne značajke Java programskog jezika su:

- Jednostavni, objektno-orijentirani i prepoznatljivi jezik,
- jezik koji je robustan i siguran,
- neovisan o arhitekturi na kojoj se pokreće,
- prenosiv,
- podržava više dretvi,
- interpretirani jezik.

Osnove jezika Java

Većinu svoje sintakse Java nasljeđuje iz jezika C++. Za razliku od C++, u Javi je teže programirati u funkcionalnom stilu, Java je više prilagođena za objektno-orijentirani stil. Kod napisan u Javi nalazi se u klasama, pa čak i glavna izvršna metoda *main* mora biti unutar klase. Klasa je temeljni sastavni dio objektno-orijentiranog programiranja u Javi i sastoji se od atributa i metoda. Svaki podatak u javi je objekt, iznimka je primitivni tipovi podataka (na primjer *integer, float, boolean, character),* koji nisu objekti zbog brzine izvođenja. Java za razliku od C++ ne podržava preopterećenje operatora, višestruko nasljeđivanje, pokazivače, ali najveća razlika je, da se upravljanje memorije u Javi rješava pomoću sakupljača smeća (engl. *Garbage collector*). Sakupljač smeća brine se o alokaciji i delokaciji memorije, i tako olakšava posao programerima. U programskim jezicima poput C i C++ potrebno je eksplicitno navesti kada se

memorija nekog objekta treba otpustiti. Javin sakupljač smeća to radi automatski, pa je teže dobiti curenje memorije (engl. me*mory leak*).

```
1  import java.util.Scanner;
2
3  public class CountBgThree  {
4      public static void main(String args[]) {
5          int brojac = 0;
6          Scanner in = new Scanner(System.in);
7
8          for (int i = 0; i < 5; ++i) {
9              int broj;
10             broj = in.nextInt();
11
12             if (broj > 3) ++brojac;
13         }
14
15         System.out.println(brojac);
16     }
17 }
```

Slika 9: Primjer programa u Javi

Primjer na slici 9 pokazuje glavnu metodu *main* napisanu u klasi *CountBgThree*. Na početku koda koristi se ključna riječ *import,* koja ukazuje da se će se koristiti klasa *Scanner,* i da prilikom izvođenja koda interpreter razumije od kuda dolazi objekt *Scanner* definiran na liniji 6. Klasa *Scanner* pomaže sa raščlanjivanjem primitivnih tipova i tipa *String* kod čitanja tekstova. Metoda *main* u Javi mora biti definirana sa potpisom *public static void*. *Public* označava da se *main* metoda može pozvati iz bilo kojeg objekta. *Static* označava da je *main* metoda statična, to jest da se ne treba instancirati pomoću ključne riječi *new. Void* pokazuje da metoda *main* ne vraća nikakvu vrijednost. *For* petlja izgleda sintaksno isto kao i *for* petlja u C++-u. Na liniji 6 vidimo korištenje ključne riječi *new*, koja stvara novu instancu tipa *Scanner* i sprema njezinu referencu u varijablu *in*. Korištenje objekta *in* vidi se na liniji 10, gdje se koristi metoda *nextInt* koja prima na ulaz tip podataka *String* i vraća tip *integer.*

9. PROGRAMSKI JEZIK PYTHON

Python programski jezik razvija se u kasnim osamdesetim godinama i razvio ga je Guido Van Rossum. Pyhton je *high-level* programski jezik opće namijene, u kojemu je moguće koristi jezik kao skriptni jezik (na primjer Perl) i kao sistemski jezik (na primjer C). Python je interpretirani jezik i za njegovo izvođenje potrebno je kod prevesti u *bytecode*. Glavna značajka programskog jezika Python je njegova jednostavnost i lako korištenje. Jednostavnost je ostvarena kroz laku čitljivost i lako shvatljivu sintaksu. Python je jezik koji podržava više paradigmi pisanja koda; strukturirano programiranje, objektno-orijentirano programiranje i pomoću određenih jezičnih značajki (poput alata *map, filter, reduce*) podržano je i funkcionalno programiranje. Pomoću ekstenzija podržane su i paradigme poput: dizajn prema ugovoru i logičko programiranje. Glavna filozofija jezika opisana je u dokumentu *The Zen Of Python*, koja uključuje aforizme poput:

- Lijepo je bolje od ružnog,
- eksplicitno je bolje od implicitnog,
- jednostavno je bolje od kompleksnog,
- kompleksno je bolje od kompliciranog,
- čitljivost je bitna,
- ako je implementacija teška za objasniti, onda je loša ideja,
- ako je implementacija laka za objasniti, onda bi mogla biti dobra ideja.

Najpoznatija implementacija Pythona je CPython, napisan u C-u. Javina verzija Pythona zove se Jython i može se koristiti prirodno kroz Java kod. Postoji i C# verzija, nazvana Iron Python, koja je namijenjena .Net okviru. Za istraživanje i razvoj, osnovan je PyPy projekt (implementacija Pythona pomoću Pythona) u 2003. godini, kako bi omogućio Python programerima da mijenjaju interpreter po njihovoj volji, na već poznat način. PyPy projekt je projekt otvorenog koda, razvijan zajednički preko zajednice programera, ujedno i podržan od strane Europske Unije. Python je distribuiran zajedno s licencom podržanom od strane *Open Source Initiative*, što ga čini besplatnim za korištenje, čak i za komercijalne proizvode. (Lovrenčić i sur., 2009.)

Osnove jezika Python

Glavna namjena Python programskog jezika je njegova laka čitljivost, koju ostvaruje pomoću urednog vizualnog rasporeda i jednostavnih engleskih ključnih riječi. Python ne koristi vitičaste zagrade za označavanje blokovih izraza i točka-zarez poslije izraza je nepotrebna, za razliku od većinu ostalih programskih jezika. Za označavanje blokovnih izraza Python koristi uvlačenje dijelova koda kao oznaku za početak blokovne strukture. Python koristi *duck typing*[18] i objekti se uvijek definirani po tipu, a varijable bez tipa. Iako Python koristi dinamične tipove, Python ne dozvoljava operacije koje nisu dobro definirane (na primjer zbrajanje varijable tipa broja i tipa *string*). U Pythonu je moguće definirati vlastite tipove pomoću klasa, koje se koriste za objektno-orijentirano programiranje. Za kontrolu toka Python koristi klasične konstrukcije poput: *for ... in ...* petlje, *while* petlje, *if* i *if else* izraze.

```
1  brojac = 0
2
3  for i in range(5):
4    broj = int(input())
5    if broj > 3: brojac += 1
6
7  print(brojac)
```

Slika 10: Primjer programa u Pythonu

Jednostavnost i čitljivost sintakse možemo dokazati primjerom na slici 10. Varijabla *brojac* definirana je na prvoj liniji na primjeru bez deklariranja tipa. Interpreter će prilikom izvođenja programa odrediti tip podatka varijable *brojac*. Kada interpreter vidi da je vrijednost varijable 0, odredit će tip podataka kao *integer*. Na liniji 3 definirana je *for in* petlja kojom je definiran iterator *i,* zajedno s opsegom izvođenja (početak i kraj). Opseg izvođenja petlje definiran je jednostavnom funkcijom *range(5)*, koja određuje izvođenje petlje 5 puta (iterator počinje s 0 i završava sa 4). Izrazi koji se izvršavaju u petlji uvučeni su za dva razmaka. Za unos i prikaz podatka koriste se funkcije *input* i *print*.

18 Dinamičan način određivanja tipova koji se provjerava tek kada se kod izvodi

10. USPOREDBA PROGRAMSKIH JEZIKA NAMIJENJENIH UČENJU

Usporedbu programskih jezika namijenjenih učenju možemo vršiti pomoću kriterija navedenih u drugom poglavlju, a ti kriteriji su:

- Jezik je pogodan za nastavu: za ispunjavanje ove kriterije programski jezik treba biti dizajniran tako da je namijenjen učenju. Jezik treba imati jednostavnu sintaksu i prirodnu semantiku, izbjegava kriptične simbole, kratice i druge izvore konfuzije.
- Nudi opći okvir: za ispunjavanje ove kriterije jezik treba pružati mogućnost učenja temeljnih ideja i principa programiranja, koje će služiti kao osnova za daljnje učenje drugih programskih jezika.
- Promovira novi pristup učenju: za ispunjavanje ove kriterije jezik ne bi trebao biti samo ograničen za svoju implementaciju, već pokrivati širi aspekt razvojnog procesa. Jezik bi trebao biti dizajniran kao cijela metodologija za stvaranje programa (dizajn uključuje set principa, alata i biblioteka zajedno s dizajnom jezika).
- Promovira pisanje ispravnih programa: za ispunjavanje ove kriterije učenici jezika bi trebali imati načine za provjeravanje točnosti koda.
- Dopušta rješavanje problema koji se mogu riješiti u malim dijelovima koda: za ispunjavanje ove kriterije jezik bi trebao imati podršku za modularizaciju (kroz funkcije, procedure, ili druge ekvivalente načine).
- Nudi besprijekorna i jednostavna razvojna okruženja: za ispunjavanje ove kriterije razvojno okruženje mora imati intuitivno grafičko okruženje za dizajn i implementaciju i mora nuditi pristup značajkama i bibliotekama jezika (za osnovno i napredno programiranje).
- Podržan na različitim okruženjima: za ispunjavanje ove kriterije jezik treba biti dostupan na različitim platformama.
- Ima korisničku podršku kroz zajednice: za ispunjavanje ove kriterije jezik mora imati adekvatnu podršku za učenje i korištenje jezika. Podrška može biti u obliku web stranica, knjiga, vodiča, zbirka zadataka, dokumentacije.

- Otvoreni kod: za ispunjavanje ove kriterije jezik bi trebao biti invencija grupe koja ne teži stvaranju komercijalnog proizvoda i kojemu je moguće doprinijeti po želji.
- Ima prateći adekvatni nastavni materijal: za ispunjavanje ove kriterije, udžbenici i ostali materijali bi trebali biti dostupni.
- Ne koristi se samo u obrazovanju: za ispunjavanje ove kriterije jezik bi trebao biti relevantan u drugim područjima osim edukacije (pogodan za razvijanje velikih aplikacija iz stvarnog svijeta).

Programski jezici Logo, Pascal, Scratch i Python pogodni su za nastavu koja je namijenjena učenju programiranja. Sadrže sintaksu koja je jednostavna za početnike, nemaju kriptične simbole koji zbunjuju programere početnike. Logo, Pascal i Python to omogućavaju sa sintaksom koja ima lako shvatljive ključne riječi, zajedno s vizualnom lakše shvatljivim metodama označavanja blokova koda (uvlačenje koda i ključne riječi *to* i *end*, umjesto vitičastih zagrada). Izbjegava se implementacija kratica za ključne riječi, pa je i konfuzija kod shvaćanja značenja riječi manja. Programski jezik Scratch ima prednost nad ostalim navedenim jezicima, jer ima pristup slaganja koda pomoću vizualnih blokova. Svi dijelovi koda vizualno su reprezentirani i nema potrebe za pisanjem koda, nego se dijelovi koda povlače iz ponuđene palete. Ovaj način učenja programiranja uklanja sintaktičke greške, koje su uobičajene kod ostalih ručno pisanih programskih kodova. Naravno ovaj način učenja ima svoje mane, a to je da učenici imaju probleme kod prelaska na druge programske jezike, to jest moraju se prilagoditi i naučiti pisati kod ručno.

U usporedbi, svi programski jezici opisani u prijašnjim poglavljima ispunjuju kriterij u kojem je potrebno pružati mogućnost učenja temeljnih ideja i principa programiranja. Programski jezici C++, Java i Python prirodno podržavaju objektno-orijentirano programiranje, pa se isti principi mogu primijeni na drugim programskim jezicima koji omogućavaju objektno-orijentirano programiranje. Za jezike BASIC i Pascal postoje verzije poput FreeBASIC i FreePascal koje podržavaju objektno-orijentirano programiranje.

Jezici Logo i Scratch jedini ostvaruju kriterij promoviranja novog pristupa učenju. Geometrija kornjače omogućava učeniku da uči o programiranju kroz svijet kornjače. Kod rješavanja problema, učenik se postavlja na mjesto kornjače i razvija rješenje kroz njegov svijet (mikro svijet). U istoj mjeri je ostvaren kriterij kod programskog jezika Scratch.

Za promoviranje pisanja ispravnih programskih jezika, potrebno je informirati učenika o potencijalnim greškama u programu. Mnoga okruženja u programiranju, nude način provjeravanja greški vezanih za sintaksu. Provjeravanje se izvodi tijekom pisanja koda i u slučajevima kada razvojno okruženje prepoznaje grešku, ona obično bude podcrtana valovitom crtom. U programski jeziku Scratch, onemogućene su sintaktičke kombinacije koje su neispravne, pa su programi ispravniji. Pojedine blokove u Scratchu je moguće pokrenuti klikom na blok koda i učenik može odmah provjeriti da li je taj dio koda ispravan. Kod korištenja jezika C++, Java, Python i Pascal uobičajeno je korištenje Unit testova, koji omogućavaju testiranje dijelova koda. Unit testovi zahtijevaju njihovo definiranje, pa je to određenim učenicima početnicima dodatno opterećenje kod učenja programiranja.

Programski jezici koji su objektno-orijentirani imaju izraženu veću mjeru kod ispunjavanja kriterija modularizacije. Objektno-orijentirani pristup s klasama, omogućava modularnost na većoj razini nego jezici koji koriste samo funkcije ili procedure za modularizaciju. Svi jezici u usporedbi, ispunjuju kriterij modularizacije kroz klase, funkcije, procedure ili metode. U Scratchu je moguće definirati posebne blokove koji se mogu izvršavati kao procedure, to jest funkcije.

Svaki jezik u usporedbi ima veći broj izbora razvojnog okruženja koji nude intuitivno grafičko okruženje zajedno s pristupom jezičnih značajki, bibliotekom za osnovno i napredno programiranje i integracijom sa sukladnim kompajlerom. Neki od poznatijih integriranih razvojnih okruženja su: Visual Studio (C++), Eclipse, BlueJ, NetBeans (Java), PyDev, PyCharm (Python).

Svi opisani jezici nude široku podršku kroz mnogobrojne web stranice koje nude upute i vodiče za korištenje programskog jezika. Sve dokumentacije su u velikoj mjeri detaljne i pružaju veliku količinu informacija o programskom jeziku.

Implementacije programskih jezika Logo (UCBLogo), BASIC (FreeBasic), Pascal (FreePascal), Scratch, C++ (gcc), Java (javac), Python (PyPy) su otvorenog koda i njihovi kompajleri dostupni su za modifikacije i promjene. Svi navedeni su licencirani kroz GPLv2 licencu, osim PyPy implementacije, koja je licencirana s MIT licencom.

Svi navedeni programski jezici nude adekvatni nastavni materijal. Udžbenici i ostali materijali dostupni su na različitim svjetskim jezicima.

Logo programski jezik je izrazito moćan i s njime je moguće stvarati aplikacije iz stvarnog svijeta, ali njegovo korištenje možemo naći samo u edukaciji. Danas se programski jezik Visual Basic koristi u raznim poduzećima za razvijanje velikih aplikacija, ali verzije poput FreeBasic se jako malo koriste izvan obrazovanja. Programski jezici Pascal, C++, Java i Python koriste se u velikoj mjeri kao jezici za razvijanje naprednih aplikacija. Java programski jezik poznat je po svojim zahtjevnim aplikacijama za poduzeća, i aplikacijama namijenjenim Android mobilnim uređajima. Jezik Python, osim svoje namijene u edukaciji, koristi se kao skriptni jezik i jezik za razvijanje *back-enda.* Programski jezik Scratch ne ispunjuje kriterij i koristi se samo u edukaciji.

Tablica 1: Usporedba programskih jezika namijenjenih učenju po kriterijima

Kriterij / Jezik	Logo	BASIC	Pascal	Scratch	C++	Java	Python
pogodan za nastavu	+ dizajniran s ciljem učenja programiranja, jednostavna sintaksa bez kriptičnih simbola, nema vitičastih zagrada, jednostavne ključne riječi	+ jednostavna sintaksa - zastarijeli koncepte (GOTO), ne strukturirano programiranje	+ dizajniran s ciljem učenja programiranja, jednostavna sintaksa bez kriptičnih simbola, nema vitičastih zagrada, jednostavne ključne riječi	+ dizajniran s ciljem učenja programiranja, jednostavni vizualni pristup, programiranje pomoću blokova - problemi kod prelaska na druge ne vizualne jezike	+ omogućava objektno orjentirani ili funkcionalni stil pisanja - sintaksa nije jednostavna za početnike, zbunjujući simboli	- sintaksa nije jednostavna za početnike, zbunjujući simboli	+ dizajniran s ciljem učenja programiranja, jednostavna sintaksa bez kriptičnih simbola, nema vitičastih zagrada, jednostavne ključne riječi
nudi opći okvir	+ sadži temeljne ideje i principe programiranja - ne nudi OOP	+ sadži temeljne ideje i principe programiranja, moguće učenje OOP pomoću nekih verzija BASICA	+ sadži temeljne ideje i principe programiranja, moguće učenje OOP pomoću nekih verzija Pascala	+ sadži temeljne ideje i principe programiranja - ne nudi OOP	+ sadži temeljne ideje i principe programiranja, moguće učenje OOP	+ sadži temeljne ideje i principe programiranja, moguće učenje OOP	+ sadži temeljne ideje i principe programiranja, moguće učenje OOP
promovira novi pristup učenju programiranja	+ koristi geometriju kornjače kao novi pristup učenju (mikrosvijetovi) - ne ispunjuje kriterij	- ne ispunjuje kriterij	- ne ispunjuje kriterij	+ koristi vizualni pristup kao novi pristup učenju (slaganje blokova i mikrosvijetovi)	- ne ispunjuje kriterij	- ne ispunjuje kriterij	- ne ispunjuje kriterij
promovira pisanje ispravnih programa	- ne ispunjuje kriterij	+ moguće prikazati nepravilnu sintaksu kroz integrirano okruženje, pisanje unit testova	+ moguće prikazati nepravilnu sintaksu kroz integrirano okruženje, pisanje unit testova	+ sintaksa nije pisana, već pravilno oređena u blokovima, blokove je moguće pokrenuti zasebno radi provjere	+ moguće prikazati nepravilnu sintaksu kroz integrirano okruženje, pisanje unit testova	+ moguće prikazati nepravilnu sintaksu kroz integrirano okruženje, pisanje unit testova	+ moguće prikazati nepravilnu sintaksu kroz integrirano okruženje, pisanje unit testova
modularnost	+ modularnost podržana pomoću procedura	+ modularnost podržana pomoću funkcija i metoda - podžano samo na nekim verzijama	+ modularnost podržana pomoću procedura i funkcija	+ modularnost podržana pomoću rulno napravljenih blokova - ograničena svojstva blokova	+ modularnost podržana pomoću metoda i funkcija	+ modularnost podržana pomoću metoda i funkcija	+ modularnost podržana pomoću metoda i funkcija

Tablica 1: Usporedba programskih jezika namijenjenih učenju po kriterijima (nastavak)

Kriterij / Jezik	Logo	BASIC	Pascal	Scratch	C++	Java	Python
nudi besprijekorna i jednostavna razvojna okruženja	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler	+ scratch.mit.edu nudi odlicno integirano okruzenje u browseru za razvijanje programa u Scratchu - okruženja vezana za kompajler	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler	+ pristup velikom broju razvojnih okruženja, jednostavna i intuitivna - okruženja vezana za kompajler
podržan na različitim okruženjima	+ dostupan na različitim platformama	+ dostupan na različitim platformama - visual basic vezan za Microsoft okruženje	+ dostupan na različitim platformama	+ dostupan na različitim platformama	+ dostupan na različitim platformama	+ dostupan na različitim platformama	+ dostupan na različitim platformama
ima korisničku podršku kroz zajednice	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka	+ široka i opsežna dokumentacija, razni vodiči i zbirke zadatka
otvorenog koda	+ poznata verzija UCBLogo je otvorenog koda	+ poznata verzija FreeBasic je otvorenog koda - razne verzije koje su pod vlasničkom licencom	+ poznata verzija FreePascal je otvorenog koda - razne verzije koje su pod vlasničkom licencom	+ scratch.mit.edu je otvorenog koda	+ poznata verzija gcc je otvorenog koda - razne verzije koje su pod vlasničkom licencom	+ poznata verzija javac je otvorenog koda - razne verzije koje su pod vlasničkom licencom	+ sve verzije Pythona su otvorenog koda
ima prateći adekvatni nastavni materijal	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima	+ postoje udžbenici na različitim svijetskim jezicima
ne koristi se samo u obrazovanju	+ moguće pisanje kompleksnih i naprednih aplikacija	+ moguće pisanje kompleksnih i naprednih aplikacija	+ moguće pisanje kompleksnih i naprednih aplikacija	- koristi se samo u edukaciji	+ moguće pisanje kompleksnih i naprednih aplikacija	+ moguće pisanje kompleksnih i naprednih aplikacija	+ moguće pisanje kompleksnih i naprednih aplikacija

ZAKLJUČAK

Naučiti programiranje je zahtjevan proces i većina predmeta koji su namijenjeni početnom učenju programiranja imaju slabiju prolaznost. Neki od uobičajenih problema s kojima se studenti susreću kod učenja programskih jezika su manjak predznanja i manjak fokusa zbog ostalih dužnosti na studiju. Postoje i teži problemi, poput straha od programiranja i percepcija programiranja kao veoma teška radnja (Radoševic i sur., 2009). U većini slučajeva učiti programirati znači učiti sintaksu, zanemarujući algoritamsko i logičko rješavanje problema. Problem zanemarivanja razvijanja načina rješavanja problema dovodi do poteškoća shvaćanja koda i shvaćanja šire slike programiranja. Mnoge programere početnike, pri pisanju prvog računalnog programa, prati osjećaj neuspjeha jer programiranje izaziva neočekivana ponašanja programa kao što su sintaktička pogreška, pogreška kod izvođenja programa ili neočekivani izlazni podaci. Sve navedeno može se ubrojiti u oblike povratnih informacija. Povratne informacije ključne su za pomaganje učenicima u razumijevanju samog programa te načina na koji računala interpretiraju program. (Bubica, 2015)

Odabir jezika namijenjenom učenju je važan faktor kod učenja programiranja. Ovaj rad opisuje pojedine programske jezike namijenjene učenju. Programski jezici Logo i Scratch preporučljivi su učenicima osnovnih škola kao prvi programski jezici, zbog svoje vizualne naravi. Učenici imaju lakše shvaćanje ideje programiranja i svoje naučene koncepte mogu primijeniti u drugim područjima, kao na primjer kroz mikro-svjetove, robote i programiranje igara.

Programski jezik BASIC je u današnjem dobu zastario, uči loše programerske prakse i nije preporučljiv kao prvi programski jezik. Programski jezik Pascal je isto u pojedinim shvaćanjima zastario, ali se još uvijek koristi hrvatskim srednjim školama. Programski jezici C++ i Java manje su prikladni za odabir kao prvi programski jezik, nego jezik Python. Razlog je što Python nudi jednostavniju i lakše shvatljivu sintaksu. Jezici koji omogućavaju pisanje koda u objektno-orijentiranom stilu, pogodniji su kod ponovnog iskorištavanja koda i s takvim stilom početnici uče o pisanju programa u manjim i jasnim programima.

Bitno je naglasiti da je kod učenja programiranja i programskih jezika bitan angažman učenika, kako bi se naučeno moglo ponovo iskoristiti.

LITERATURA

1) Bjelica M. (2016.): Programski jezik Python, URL: http://www.etf.bg.ac.rs/etf_files/udzbenici/python.pdf (pristupljeno: 27.8.2017.)
2) Bubica N. (2014.): Strategije poučavanja i faktori koji utječu na unapređenje znanja programera početnika, URL: http://www.pmfst.unist.hr/wp-content/uploads/2014/06/Istraziva--ki-seminar1-Bubica.pdf (pristupljeno: 17.7.2017.)
3) Bubica N., Mladenović M., Boljat I. (2013.): Programiranje kao alat za razvoj apstraktnog mišljenja, URL: https://bib.irb.hr/datoteka/702093.Programiranje_kao_alat_za_razvoj_apstraktnog_miljenja-CUC-zbornik.pdf (pristupljeno: 17.7.2017.)
4) CARNet, online tečaj "Programiranje u Pascal-u", URL: https://tesla.carnet.hr/ (pristupljeno: 19.7.2017.)
5) Catambay B. (2001.): The Pascal Programming Language, URL: http://pascal-central.com/ppl/ (pristupljeno: 16.7.2017.)
6) Computer History, Timeline of Computer History, URL: http://www.computerhistory.org/timeline/software-languages/ (pristupljeno: 19.7.2017.)
7) Deljac S. (2015.) Predavanje "Informatika u srednjim školama" , URL: http://www.ieee.hr/_download/repository/Informatika_u_srednjim_skolama_novo.pdf (pristupljeno: 15.7.2017.)
8) DuCharme B. (2012.): Logo for Kids: An Introduction, URL: http://www.snee.com/logo/logo4kids.pdf (pristupljeno: 15.7.2017.)
9) Gebauer H., Hromkovič J., Keller L. (2012.): Programming in LOGO, URL: http://abz.inf.ethz.ch/wp-content/uploads/unterrichtsmaterialien/primarschulen/logo_heft_en.pdf (pristupljeno: 15.7.2017.)
10) Gosling J. i McGilton H. (1996.): The Java Language Environment, URL: http://www.oracle.com/technetwork/java/index-136113.html (pristupljeno: 26.8.2017.)
11) Grinfeld-Gradiški M. (1998.): Logo programiranje 1: Ja hoću i Ja mogu programirati na

Logu, Zagreb: vlastiti nakladnik

12) Hercigonja Z. (2017.): Programiranje u Pythonu, URL: https://bib.irb.hr/datoteka/884671.foi-programiranje-u-pythonu.pdf (pristupljeno: 27.8.2017.)

13) Krpan D., Mladenović S., Zaharija G. (2014.): Vizualni programski jezici u visokom obrazovanju, URL: http://www.pmfst.unist.hr/wp-content/uploads/2014/06/Istraziva--ki-seminar1-Bubica.pdf (pristupljeno: 19.7.2017.)

14) Kruglyk V. i Lvov M. (2012.): Choosing the First Educational Programming Language, URL: http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-37-p-188-198.pdf (pristupljeno: 20.7.2017.)

15) Lewis, C.M. (2010.): How programming environment shapes perception, learning and goals: logo vs. scratch., URL: http://ims.mii.lt/ims/konferenciju_medziaga/SIGCSE%2710/docs/p346.pdf (pristupljeno: 19.7.2017.)

16) Lovrenčić A., Konecki M., Orehovački T. (2009.): 1957-2007: 50 Years of Higher Order Programming Languages, Zagreb: FOI, Journal of Information and Organizational Sciences

17) Mannila L. i Raadt M. (2006): An Objective Comparison of Languages for Teaching Introductory Programming, URL: https://pdfs.semanticscholar.org/dc53/91d339571d7914f2fddf18ce725d64b9abb1.pdf (pristupljeno: 20.7.2017.)

18) Mesar, V. (2008.): Programiranje, udžbenik programiranja u Pascalu za 2. i 3. razred srednje škole, Zagreb: Školska knjiga

19) Miklec D. (2013.): Programiranje u nastavi informatike, Zagreb: Sveučilište u Zagrebu, URL: https://bib.irb.hr/datoteka/695646.DMiklec_diplomski.pdf (pristupljeno: 19.7.2017.)

20) Programming in BASIC, URL: https://materdeiit.wikispaces.com/file/view/Programming-in-BASIC_MS.pdf (pristupljeno 15.7.2017)

21) Programming Paradigms, URL: http://cs.lmu.edu/~ray/notes/paradigms/ (pristupljeno:

19.7.2017.)

22) QBASIC za Početnike, URL: https://milenakostadinovic.files.wordpress.com/2016/03/qbasic_za_apsolutne_pocetnike.pdf (pristupljeno: 19.7.2017.)

23) Radošević D., Orehovački T., Lovrenčić A. (2009.): New Approaches and Tools in Teaching Programming, URL: http://bib.irb.hr/datoteka/427643.CECIIS2009_Radosevic_Orehovacki_Lovrencic.pdf (pristupljeno: 19.7.2017.)

24) Radošević D., Orehovački T., Lovrenčić A. (2009.): Verificator: Educational Tool for Learning Programming, URL: http://bib.irb.hr/datoteka/429295.INFE154.pdf (pristupljeno: 19.7.2017.)

25) Ristić O., Milošević D., Urošević V. (2016): The importance of programming languages in education, URL: http://www.ftn.kg.ac.rs/konferencije/tio2016/Radovi%20TIO%202016/EN/3)%20Information%20and%20Educational%20Technologies/319_046_Ristic%20i%20sar_EN.pdf (pristupljeno: 20.7.2017.)

26) Stroustrup B. (1994.): The Design and Evolution of C++ (1. izdanje), Addison-Wesley Professional

27) Stroustrup B. (2013.): The C++ Programming Language (4. izdanje), Addison-Wesley Professional

28) Topolnik M. i Kušek M. (2008.): Uvod u programski jezik Java, URL: http://www.fer.unizg.hr/_download/repository/Skripta-Java.pdf (pristupljeno: 26.8.2017.)

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom završnom radu, čija je tema "Programski jezici namijenjeni učenju programiranja", cilj je upoznavanje s programskim jezicima, koji za cilj imaju olakšati učenje programiranja. Odabir programskog jezika namijenjenoga učenju, umjesto jezika koji takve ciljeve nema, znatno smanjuje probleme kod kojih se početnici susreću kod učenja programiranja. Neke od glavnih značajki programskih jezika namijenjenih učenju su: jednostavna i lako shvatljiva sintaksa, jednostavno razvojno okruženje i drugačiji pristup učenju. Jezici koji su opisani i uspoređivani u ovom radu su: Logo, BASIC, Pascal, Scratch, C++, Java i Python.

Ključne riječi: programski jezici namijenjeni učenju, učenje programiranja, sintaksa, Logo, BASIC, Scratch, C++, Java, Python

ABSTRACT

In this bachelor's thesis, whose topic is "Programing languages meant for learning programming", the goal is to get to know the programming languages, that aim to make learning programming easier. Selecting a programing language meant for learning, instead of a language that isn't for learning, considerably lessens the problems that beginners face when learning programming. Some of the main features of programming language meant for learning are: simple and easy to understand syntax, simple developer environment and different teaching approach. The languages that are described and compared in this paper are: Logo, BASIC, Pascal, Scratch, C++, Java and Python.

Keywords: programing languages meant for learning, learning programming, syntax, Logo, BASIC, Scratch, C++, Java, Python