Usporedba programskih jezika temeljenih na sintaksi jezika C

Filip, Crnko

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:620630

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-03-28**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Crnko

Usporedba programskih jezika temeljenih na sintaksi jezika C

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Crnko

Matični broj: 0016116698

Studij: Informacijsko i programsko inženjerstvo

Usporedba programskih jezika temeljenih na sintaksi jezika C

DIPLOMSKI RAD

Mentor/Mentorica:

Doc. dr. sc. Mario Konecki

Varaždin, kolovoz 2020.

Filip Crnko

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome radu bavim se usporedbom programskih jezika baziranih na sintaksi jezika C. Kao glavne jezike zbog njihove raširenosti i popularnosti odlučio sam detaljnije opisivati: C, C++, C#, PHP, JavaScript i Java programske jezike. Bavio bi se samim razlikama i sličnostima navedenih programskih jezika. Polazeći od same razlike u namjeni jezika, razvojnim okruženjima, sintaksi, brzini izvođenja te bi napisao subjektivni osvrt o težini učenja samog jezika. Kako bi mogao detaljnije predočiti same sličnosti i razlike, kreirat ću razne isječke programskog koda. Krenuvši od jednostavnijih stvari poput ispisa teksta na ekran do složenijih principa objektno orijentiranog programiranja.

Ključne riječi: Usporedba sintakse, C, C++, C#, PHP, JavaScript, Java

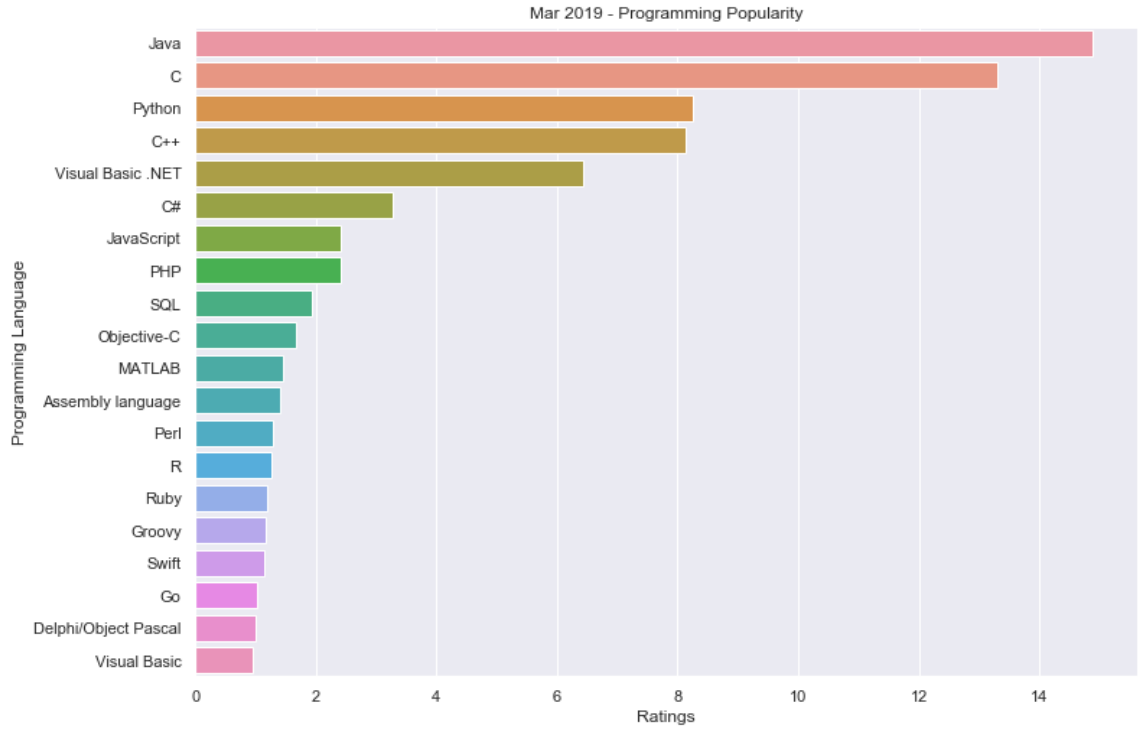
Sadržaj

Sadržaj iii
1. Uvod 1
2. Metode i tehnike rada 2
3. Proceduralno i objektno orijentirano programiranje 3
4. Općenito o programskim jezicima 4
4.1. C 4
4.2. C++ 6
4.3. C# 7
4.4. Java 9
4.5. JavaScript 11
4.6. PHP 12
5. Usporedba vanjskih osobina programskih jezika 13
5.1. Razvojno okruženje 13
5.1.1. Dev-C++ 14
5.1.2. Visual Studio 15
5.1.3. Eclipse 16
5.1.4. Brackets 16
5.2. Brzina izvođenja 17
5.2.1. Primjer provedenog istraživanja 18
5.2.2. Vlastita usporedba brzine izvođenja 19
5.3. Lakoća učenja 24
5.3.1. Primjer provedenog istraživanja 24
6. Usporedba sintakse 26
6.1. Osnovne razlike u sintaksi 26
6.1.1. Deklariranje varijabli 27
6.1.2. Deklariranje konstanti 28
6.1.3. Deklariranje polja i listi 29
6.1.4. Pisanje komentara 31
6.2. Usporedba primjera ispisa teksta na ekran 33
6.3. Usporedba naredbi kontrole toka 38
6.3.1. If else 38
6.3.2. For 39
6.3.3. While & Do While 40

6.3.4. Switch case 42
6.4. Uspoređivanje objektno orijentiranih principa rada 44
6.4.1. Definiranje klasa i objekata 44
6.4.2. Nasljeđivanje 48
7. Zaključak 49
Popis literature 51

1. Uvod

Tema ovoga rada je usporedba programskih jezika koji su temeljeni na sintaksi programskog jezika C. Iako je sam C jezik i njegova sintaksa utjecala na jako veliki broj programskih jezika, odlučio sam se za usporedbu C, C++, C#, JavaScripta, Java i PHP-a. Razlog tomu je njihova svjetska popularnost, ali isto tako i činjenica da su to jezici s kojima sam se najviše susretao tijekom studiranja. Samim time što su najzastupljeniji jezici današnjice osigurava bezbroj izvora informacija, tutoriala i primjera putem kojih se može vrlo lako naučiti sve od osnova pa do naprednih koncepata programiranja.



Slika 1: Popularnost programskih jezika 2019[1] [1]

Motivaciju za odabir i razradu ove teme pronalazim u želji da nakon pet godina studiranja zaokružim svoje znanje. Kako je svaka godina fakulteta nosila različite izazove, tako su i programski jezici u kojima smo morali rješavati te izazove bili drugačiji. Iz tog razloga studenti dobivaju široki raspon vještina i iskustva kako bi mogli donijeti bolju i informiraniju odluku o specijalizaciji u određenom području. U ovome radu obradom raznih jezika pružit ću detaljniji uvid u svaki od njih te na taj način omogućiti lakše donošenje odluke o budućoj karijeri.

[1] Popularnost programskih jezika računa se pomoću TIOBE indeksa - https://www.tiobe.com/tiobe-index//

2. Metode i tehnike rada

Prije nego što sam počeo pisati ovaj diplomski rad, napravio sam sistematizaciju kategorija u kojima želim uspoređivati ove programske jezike. Tako sam odlučio da je prvo potrebno dati uvod u svaki od programskih jezika posebno, reći nešto o samoj povijesti razvoja te razlogu zašto je taj jezik kreiran. No prije nego što se mogu uopće upustiti u uspoređivanje tih programskih jezika potrebno je objasniti njihovu glavnu razliku, a to je razlika između proceduralnog programiranja i objektno orijentiranog programiranja. Taj iskorak u načinu razmišljanja programera i načinu razvoja aplikacija je uvelike promijenio principe programiranja i razvoja softvera današnjice.

Nakon samog uvoda u cijelu temu, te detaljnijeg razumijevanja povijesti svakog programskog jezika sa kojim ću se baviti u ovome radu potrebno je objasniti i razvojna okruženja koja koristim ja i većina drugih programera koji se bave razvojem u ovim tehnologijama. Bez razvojnoga okruženja niti jedan programer ne može razviti aplikaciju bez obzira na njegovo znanje o programskome jeziku. Današnje aplikacije su toliko kompleksne da klasično programiranje u note padu je gotovo nemoguće. Svakome su potrebni današnji alati koje pružaju sva dobra razvojna okruženja.

Sa podlogom o razvojnim okruženjima odlučio sam se dotaknuti tehničkih specifikacija programskih jezika. Pod time govorim o brzini izvođenja. Nakon što sam usporedio njihove brzine izvođenja dotaknuo sam se i lakoće učenja u tim programskim jezicima. Naravno, lakoću učenja nije lako izmjeriti pa se oslanjam na subjektivan stav i na popularne komentare drugih programera.

Za praktičan dio ovoga rada odlučio sam uspoređivati isječke kodova iz svakog programskog jezika zasebno. Tako sam se odlučio krenuti od jednostavnih stvari poput deklariranja varijabli i pisanja komentara, zatim prikazati način rada sa naredbama za kontroliranje toka programa poput petlji i provjera. Krenuti sa jednostavnim primjerima ispisa teksta na ekran, a završiti sa složenijim principima rada u objektno orijentiranom programiranju. Na taj način smatram da sam dao dovoljno široku i detaljnu usporedbu ovih šest programskih jezika.

3. Proceduralno i objektno orijentirano programiranje

Prije same usporedbe programskih jezika na dubljoj razini, bitno je razumjeti glavnu razliku između programskog jezika C i ostalih jezika koje ćemo obraditi u ovom radu. C je proceduralni programski jezik, isto kao što su to i Pascal, AGOL, BASIC i drugi. Dok u jezike koji podržavaju objektno orijentirano programiranje uvrštavamo C++, C#, Javu, PHP, JavaScript, Python i druge.

Proceduralno programiranje je model programiranja koji je izveden iz strukturnog programiranja, baziran na konceptu pozivanja raznih procedura ili funkcija. Funkcije se sastoje od serije naredbi koje računalo treba izvršiti. Tijekom izvršavanja programa, bilo koja procedura može biti pozvana od strane drugih procedura ili unutar same sebe. [2]

Sa druge strane postoji objektno orijentirano programiranje. Ovu vrstu programiranja definiramo kao model koji je baziran na konceptu objekata. Današnji programski jezici su većinom bazirani na principima objektno orijentiranog programiranja. Programi su dizajnirani da koriste koncepte objekata koji sudjeluju u interakciji sa realnim svijetom. Iako postoji više vrsta objektno orijentiranih jezika, najpopularniji su oni koji koriste klase. Na taj način objekti predstavljaju instancu klase koja definira strukturu i tip samog objekta. [2]

Tablica 1: Razlike između proceduralnog i objektnog programiranja

Proceduralno programiranje	**Objektno orijentirano programiranje**
Bazirano na ne realnim principima	Bazirano na principima iz realnog svijeta
Program podijeljen na manje dijelove - funkcije	Program podijeljen na manje dijelove - objekti
Ne definira prava pristupa	Definira razinu pristupa (private, public, protected…)
Dodavanje nove funkcije u postojeći kod može biti teško	Dodavanje novih dijelova koda je lagano
Ne dopušta preopterećenje funkcija	Dopušta preopterećenje funkcija

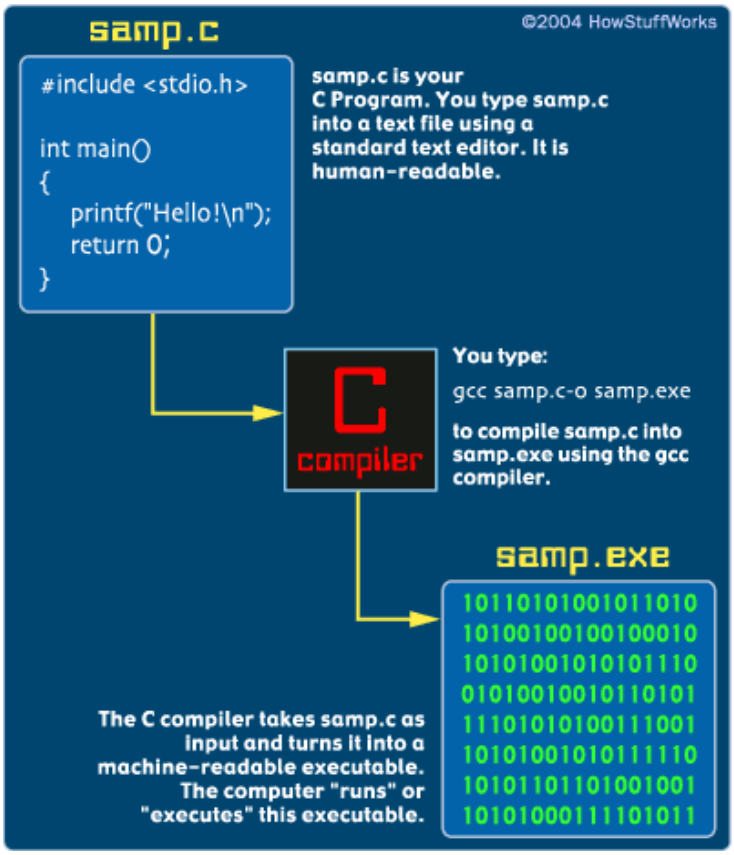
(Izvor: Geeks for geeks, „Differences between Procedural and Object Oriented Programming“)

4. Općenito o programskim jezicima

U ovom poglavlju opisati ću svaki od navedenih programskih jezika slične sintakse C jeziku. Počevši od same povijesti jezika i detalja o samom razvoju programskog jezika. Isto tako dotaknuti ću se zanimljivih činjenica o svakom jeziku posebno, te spomenuti za što se generalno taj jezik koristi i koje su neke općenite funkcionalnosti samog jezika.

4.1. C

C programski jezik je jezik visoke razine što znači da je apstraktniji i lakši za uporabu. No isto tako, C kao programski jezik ne može funkcionirati sam po sebi. Za izvršavanje programa napisanog u C jeziku, potreban nam je jezični prevoditelj. Kolokvijalno poznatiji kao kompajler. Kompajler služi za pretvaranje samog koda koji je čitljiv ljudima u naredbe koje su čitljive računalu. Primjer koda možemo vidjeti na slici 2, gdje samp.c predstavlja čitljiv dio koda a samp.exe predstavlja naredbe razumljive računalu koje je kompajler izradio.



Slika 2: Funkcija kompajlera [3]

C programski jezik je razvijen 1972. godine od strane Dennis Ritchie-a, unutar AT&T Bell Labs-a. Inicijalno je zamišljen kao jezik za svrhu programiranja i razvoja UNIX[2] operativnog sustava. Kasnije, 1978. godine Brian Kernighan i Dennis Ritchie stvaraju prvu javno dostupnu dokumentaciju programskog jezika C. Danas poznatu kao K&R standard. [4]

C je danas jedan od najpopularnijih programskih jezika na svijetu. Postoji nekoliko razloga zašto je to tako. Kao prvo, C je jezik koji se relativno lagano uči. Početnici programiranja vrlo često kreću sa primjerima u C programu. Osim toga, C je strukturirani programski jezik koji koristi funkcije, petlje i if-then-else konstrukcije. Na taj način olakšava programeru da se cijeli program prikaže kao hijerarhija kojom se kreče tok izvođenja samog programa te na taj način omogućava lakše pisanje a i kasnije održavanje ili unaprjeđivanje koda. Vrlo važan element same popularnosti ovog jezika je i činjenica da se može kompajlirati na različitim računalnim platformama što je velika prednost. Samim time program automatski dobiva puno veću publiku koja je zainteresirana za njegovo korištenje. [4]

Kao što sam već spomenuo, C je inicijalno bio zamišljen kao jezik za razvoj operacijskih sustava. Specifično za razvoj raznih programa koji čine sam operacijski sustav. Razlog zašto je C vrlo brzo postao prihvaćen i globalno popularan je taj da se samo izvođenje programa tek neznatno sporije izvodi naspram asemblerskog jezika[3]. Danas se koristi i za razne druge primjene poput programiranja modernijih programa, raznih interpretera i asemblera te raznih baza podataka. [4]

[2] UNIX je računalni operativni sustav inicijalno osmišljen za programere a kasnije postaje osnova za sisteme poput Ubuntu, Solaris itd. - https://www.geeksforgeeks.org/introduction-to-unix-system/

[3] Asemblerski jezik – jezik niže razine orijentiran računalu, svaka instrukcija u asembleru predstavlja jednu instrukciju strojnog jezika (zamijenjuje binarni kod sa slovnom oznakom) - https://hr.wikipedia.org/wiki/Asemblerski_jezik

4.2. C++

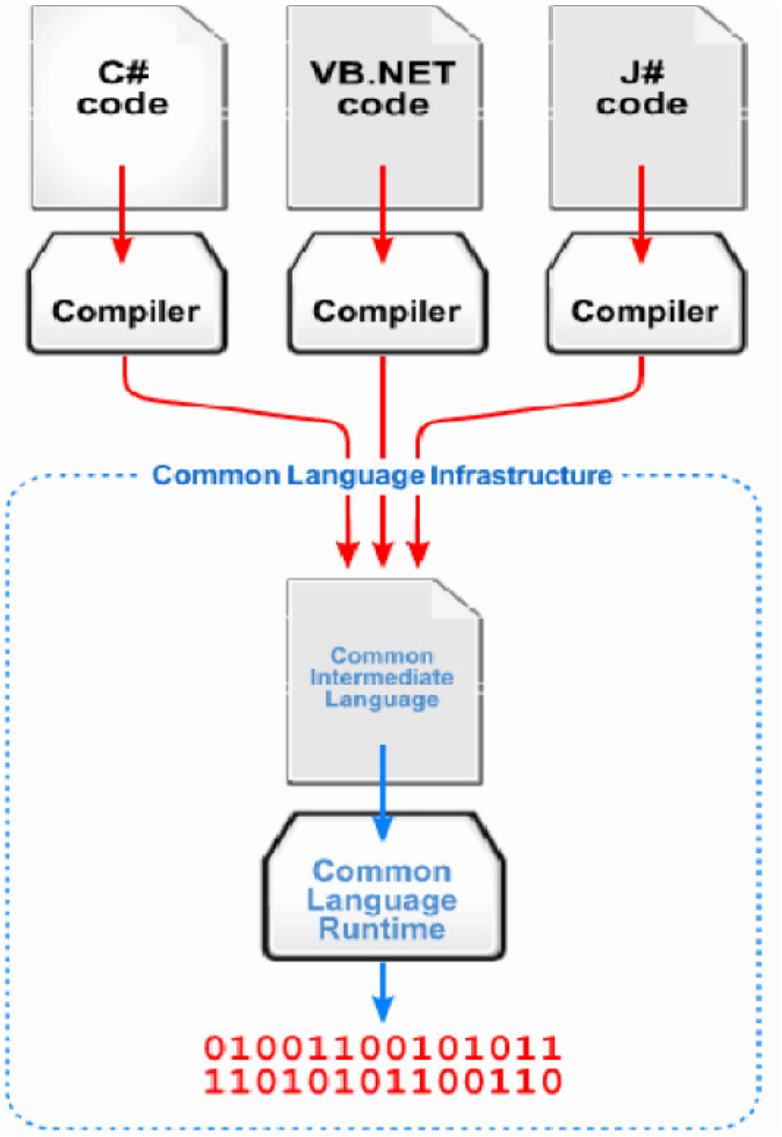
Početci C++ programskog jezika datiraju do 1979. godine kada je Bjarne Stroustrup započeo rad na svojem doktorskom radu. Jedan od jezika na kojemu je Stroustrup radio bio je Simula. Programski jezik Simula smatra se kao prvi programski jezik koji je podržavao princip objektno orijentiranog programiranja. Stroustrup je odmah prepoznao korisnost samog principa objektno orijentiranog programiranja, no isto tako uvidio je da je Simula pre spor jezik za bilo kakvu praktičnu uporabu. Iz tog razloga, počinje razvoj C jezika sa klasama i ostalim principima objektno orijentiranog programiranja. Tako 1984. godine dobivamo programski jezik C++ kao super set od C jezika. [5]

Kako C++ smatramo super setom jezika C, sam programski jezik je jako sličan C jeziku. Zbog toga iste karakteristike vrijede i za C++, poput činjenice da je to isto jezik visoke razine. Glavna razlika, kao što je već spomenuto je ta da C++ podržava objektno orijentirano programiranje. Osim samog fokusa na objektno orijentirano programiranje, C++ podržava i veliki broj biblioteka. Programeri koriste biblioteke kako bi proširili funkcionalnost samog programskog koda sa već unaprijed izrađenim dijelovima programa. Tako postoje biblioteke za razne funkcije za rad u raznim kategorijama poput rada sa zvukom ili kriptografijom. Važno je i spomenuti samu brzinu izvršavanja C++ programa. Ukoliko nam je potrebna brzina i izvršavanja i kompajliranja C++ je jedan od glavnih izbora na današnjem tržištu. I za kraj treba spomenuti pokazivače. C++ omogućuje podršku korištenja pokazivača, što mnogi drugi jezici današnjice ne podržavaju. [6]

Kako je C++ jedan od najpopularnijih jezika današnjice, možemo i očekivati da se koristi gotovo svugdje. Tako operacijski sustavi poput Windowsa, iOS-a pa čak i Linuxa koriste C i C++ kao osnovu. Isto tako ovaj programski jezik igra ključnu ulogu u renderiranju slike zbog same brzine izvođenja, pa tako C++ postaje prvi izbor u raznim aplikacijama koji koriste grafičke elemente, internet pretraživači i razne igrice koje sve trebaju imati što bolje specifikacije kako bi korisnik imao bolji doživljaj same slike. Brzina izvođenja jezika dobro dođe i bankarskim aplikacijama gdje sekunde čekanja se vrlo lako pretvaraju u velike novčane gubitke. Te treba spomenuti i velike baze podataka poput Postgresa i MySQLa koji su oboje pisani u C++ i C programskim jezicima. [6]

4.3. C#

C# je razvijen kao dio Microsoftovog .NET razvojnog okvira kolokvijalno poznatijim kao .NET framework. Zamišljen je kao jezik široke primjene 2002. godine u timu vođenim Andres Hejlsbergom. Razvijen je kao jedan od prvih jezika namijenjenih CLI infrastrukturi (*eng. Common Language Infrastructure*). CLI infrastruktura omogućuje da aplikacija napisana u nekom od češće korištenijih programskih jezika može biti izvršena na bilo kojem operacijskom sustavu koristeći jedinstveno sučelje za pokretanje (*eng. common runtime program*). [7]



Slika 3: Princip CLI infrastrukture [8]

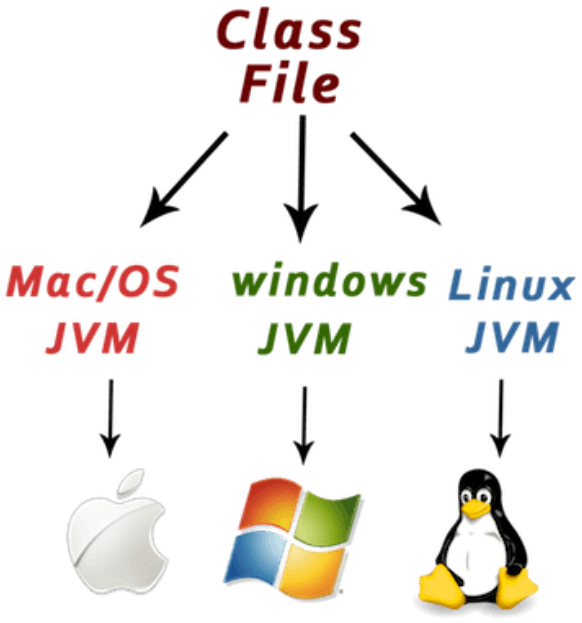
C# je jedan od mlađih programskih jezika koji je vrlo brzo stekao svoju popularnost iz nekoliko razloga. Njegova sličnost jezicima C, C++ i Javi omogućava korisnicima lakše učenje programiranja unutar samog jezika. Pošto je C# dizajniran kao jezik široke primjene, danas se koristi u razne svrhe poput razvoja mobilnih aplikacija, desktop aplikacija, aplikacija u oblaku (*eng. cloud-based services*), web stranica, igrica i drugih stvari. Posebnu pažnju treba skrenuti razvoju igrica, jedan od najpopularnijih okvira za razvoj igrica Unity koristi C#. Prije desetak godina razvoj igrica nije bio niti blizu glavnih naslova u medijima. Danas sa druge strane, industrija video igara je prestigla filmsku industriju u novčanoj vrijednosti. Vrlo često C# koristimo za razvoj web stranica, razlog tomu je .NET platforma koja je vrlo prilagodljiva samom razvoju za web. No isto tako, ukoliko programer nije zainteresiran za Microsoftovu arhitekturu, i dalje može koristiti C# za kreiranje potpuno funkcionalne web stranice. [9]

Postoji još mnogo drugih prednosti zašto bi se netko posvetio C# kao programskom jeziku. Kao prvo, programiranjem u C# jeziku štedimo na vremenu. Kako je jezik statično pisan i lako čitljiv, uzima manje vremena programeru da pronađe traženi dio koda. Mentalitet samog razvoja jezika fokusiran je na jednostavnosti i efikasnosti tako da programeri ne moraju trošiti vrijeme na pisanje kompliciranih dijelova koda koji se ponavljaju kroz program. Isto tako sam pristup velikoj kolekciji biblioteka i unaprijed definiranih funkcija omogućava brz i lagan razvoj željenog programa. Važno je i napomenuti nisku krivulju učenja, C# se smatra laganim za učenje kako početnicima tako i iskusnim programerima koji dolaze sa drugih programskih jezika. Pregršt primjera i tutoriala na internetu, te opširna dokumentacija i intuitivno razvojno okruženje utječu na brzo i jednostavno savladavanje problema. A svi ovi parametri utječu i na lako održavanje već napisanog koda. Preglednost i skalabilnost su uvijek bile jedne od poznatijih vrlina ovoga programskog jezika. [9]

4.4. Java

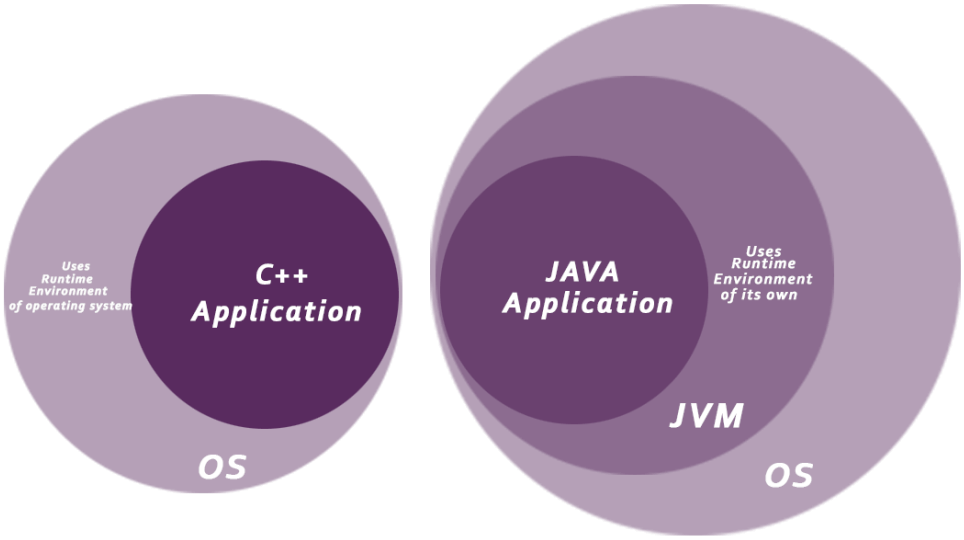
Projekt razvoja jave datira u ranim 90-ih, cilj je bio razviti programski jezik za interaktivnu televiziju ali u to vrijeme takva ideja bila je pre napredna za tv industriju. Iako je prvotna ideja bila razvoj jezika za digitalne uređaje poput televizije, java je bila prigodna i za programiranje na internetu. Tako vrlo brzo glavni principi za razvoj jave postaju jednostavnost, portabilnost, sigurnost, neovisnost o platformi, arhitekturalna neutralnost, visoke performanse i naravno objektno orijentirani pristup. Ocem jave smatra se James Gosling koji je sa svojim timom razvio sam programski jezik java, a sam početak jave smatra se godina 1995. Iako u to vrijeme, java nije imala naziv koji poznajemo danas. U početku nosila je naziv „Greentalk“, a malo kasnije je preimenovana u „Oak“. Problem je nastao vrlo brzo kada su shvatili da je Oak već tada bio registriran na tržištu kao dio imena druge tvrtke. Iz tog razloga tim se odlučio preimenovati jezik u javu. Sam naziv nije nikakav akronim, već je naziv otoka na kojem se proizvodila kava. Sam naziv je bio jedinstven i upečatljiv pa se tim odlučio da će se jedan od najpopularnijih programskih jezika današnjice nazivati java. [10]

Posebnost jave proizlazi iz JVM-a (*eng. Java Virtual Machine*) koji služi za interpretiranje napisanog java programa u kod koji će se izvršavati na računalu. Sama prednost jave i jedan od glavnih razloga popularnosti ovog programskog jezika je JVM koji omogućava da je javina platforma softverski bazirana za razliku od većine drugih jezika koji imaju hardverski baziranu platformu. Zbog toga kod napisan u javi se može pokretati na više platformi, poznatiji kao WORA princip (*eng. Write Once and Run Anywhere*). [11]



Slika 4: Java Virtual Machine [11]

JVM ne daje samo prednost u izvršavanju na različitom broju platformi, već stvara dodatan sloj sigurnosti po kojem je java poznata. Izbjegavanjem izravnih pokazivača i činjenica da se programi pisani u javi izvršavaju unutar kontroliranog okruženja JVM sandbox-a.



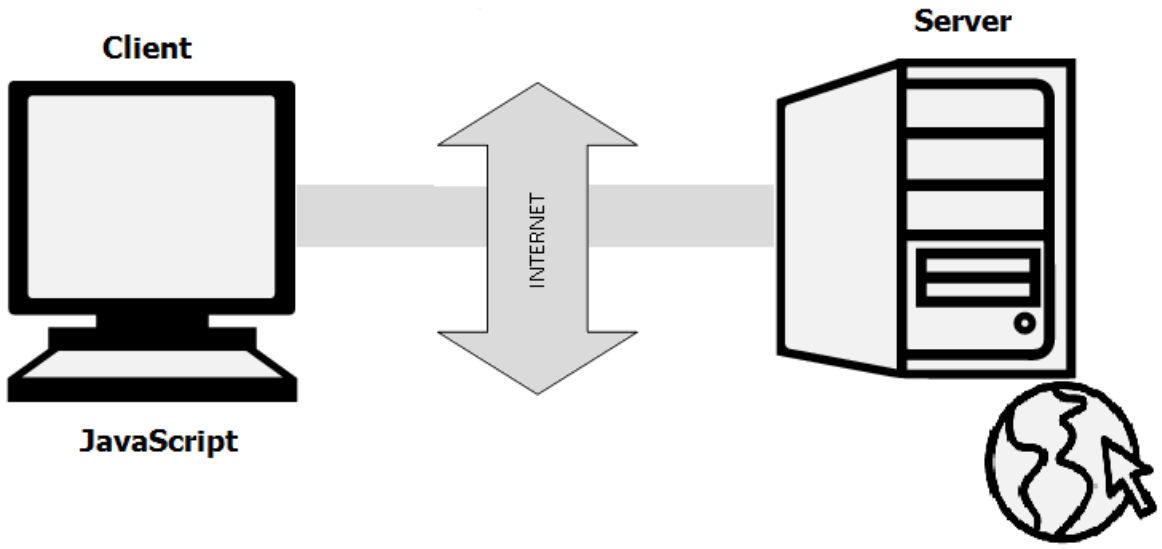
Slika 5: Sigurnost JVM sloja [11]

Važno je i spomenuti jednostavnost učenja programiranja u javi. Dizajnirana je sa ciljem lakoće učenja zbog logičnosti naredbi i sličnosti C++ jeziku. Vrlo bitno je imati i veliki broj primjera i tutoriala dostupnih na internetu. Ali podrška u učenju ne dolazi samo iz zajednice, već i od samog razvojnog tima jave. Sa vremenom unaprjeđenja u javi su rezultirali uklanjanjem nepotrebnih i kompliciranih funkcionalnosti, sve sa svrhom preglednijeg i jednostavnijeg programskog koda. I za kraj bitno je spomenuti samu performansu ovog jezika. Iako su jezici poput C, C++ i drugih unaprijed kompajliranih jezika brži od jave zbog same činjenice da je java programski jezik drugačije kategorije, java se i dalje smatra kao jezik vrlo visoke performanse zbog izvornosti samog koda pisanog u javi sa byte kodom računala. U samoj performansi izvršavanja java koda pomaže i kolektor smeća (*eng. garbage collector*) te dinamičko učitavanje klasa. Oba principa omogućavaju bolje upravljanje resursima i generalno bolje i stabilnije izvršavanje programskog koda. [11]

4.5. JavaScript

Povijest JavaScript-a počinje 1995. godine. Brendan Eich razvija JavaScript kao jezik namijenjen za programiranje web stranica. U to vrijeme Netscape Communications je držao gotovo 80% tržišne vrijednosti internet pretraživača, sa svojim Netscape pretraživačem. Eich je radio za Netscape, te je razvio JavaScript kao jezik za programiranje stranica koje bi Netscape otvarao. U samim početcima JavaScript je nosio naziv Mocha, ali kasnije dobiva naziv koji nosi danas zbog utjecaja java programskog jezika na tadašnje tržište. [12]

Zbog samog naziva, iako je doći do zaključka da su JavaScript i java slični jezici, no to ne može biti dalje od istine. JavaScript je jedan od tri osnovna jezika koja se koriste za razvoj web stranica. HTML i CSS daju strukturu i stil stranice, a JavaScript dodaje funkcionalnost i ponašanje same stranice na strani klijenta. Bitno je razumjeti što znači na strani klijenta. Sam kod programa izvršava se na računalu korisnika, dok sa druge strane izvršavanje na server znači da je server taj koji preuzima izvršavanje samoga koda. Primjer takvog jezika je PHP o kojem ćemo govoriti više kasnije u ovome radu. Iako je JavaScript primarno namijenjen izvršavanju na strani klijenta, u novije vrijeme implementacijom Node.js razvojnog okvira moguće je izvršavati JavaScript i na strani servera. [13]



Slika 6: Izvršavanje na strani klijenta [14]

JavaScript se smatra jezikom koji se lagano uči, zbog sintakse slične gore navedenim jezicima. Danas je JavaScript prisutan u preko 90% web stranica. Stranice poput Facebooka, YouTubea, eBaya i drugih koriste JavaScript za funkcionalnost same stranice. Daljnjim razvojem interneta taj broj može samo rasti. [13]

4.6. PHP

PHP kakav znamo danas je ustvari nasljednik proizvoda zvanog PHP/FI kreiranog 1994. godine od strane Rasmusa Lerdorfa. To je tada bio program pisan u C programskom jeziku koji je služio za praćenje broja posjetitelja na njegovoj osobnoj stranici. Kasnije je Lerdorf objavio javno taj set skripti zvanih PHP Tools kako bi i drugi developeri mogli koristiti njegove skripte. Sa vremenom kako je razvoj tekao dalje, Rasmus je unaprijedio njegov set naredbi te ga nazvao FI (*eng. Forms Interpreter*). FI je sadržavao neke od osnovnih funkcionalnosti kakve poznajemo u današnjem PHP-u. Današnji naziv PHP je akronim za *Hypertext Preprocessor*. [15]

PHP isto kao i JavaScript, pripada posebnoj kategoriji programskih jezika a to su skriptni jezici. Skriptni jezici se za razliku od uobičajenih programskih jezika ne koriste za razvoj kompletnih aplikacija sa raznim funkcionalnostima, već se koriste za obavljanje rutinskih operacija unutar web stranice ili sličnog. [16]

U opisivanju JavaScript programskog jezika, spomenuo sam kako se JavaScript izvršava na strani klijenta, dok PHP sa druge strane se izvršava na serveru. Na taj način se teret procesiranja naredbi prebacuje sa klijenta na server. Kako bi server mogao izvršavati PHP kod potrebno je prethodno instalirati PHP na željeni server, a nakon toga programer može pisati PHP skripte u svome kodu. Kada se ta skripta bude izvršavala, server će preuzeti odgovornost pokretanja samog koda unutar PHP tagova te će se izvršiti željene naredbe. Zbog toga sam korisnik ne mora imati instaliran PHP na svome računalu.

5. Usporedba vanjskih osobina programskih jezika

Ovim poglavljem baviti ću se detaljnijim opisom vanjskih osobina prije spomenutih programskih jezika. Dotaknuti ćemo se teme popularnih razvojnih okruženja i brzine izvođenja samih programskih jezika. Te ćemo isto tako komentirati lakoću učenja samog programskog jezika što uvelike ovisi i o samoj međusobnoj sličnosti tih jezika.

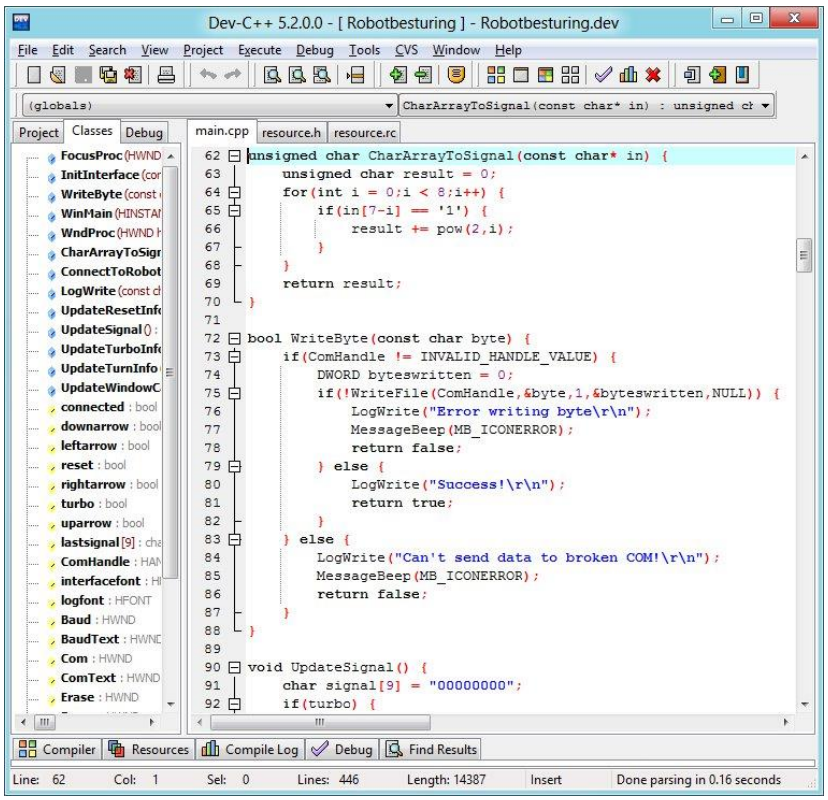
5.1. Razvojno okruženje

Razvojnim okruženjem nazivamo aplikaciju koja pruža mogućnosti developeru da razvije programski proizvod u željenom programskom jeziku. Razvojno okruženje sadrži alate i funkcije za uređivanje i izvršavanje programskog koda. Dok naprednija razvojna okruženja omogućavaju i dodatne funkcionalnosti poput testiranja samog koda.

Sam odabir razvojnog okruženja može uvelike utjecati na sposobnost korištenja programskog jezika. Ukoliko se početnik odluči za loše razvojno okruženje, može mu biti jako teško savladati programski jezik. Isto tako, bitno je i odabrati razvojno okruženje koje je dovoljno popularno i zastupljeno da ukoliko zapnemo u nekom procesu, možemo otići na internet i potražiti savjet putem nekog foruma ili videa Kod učenja novog jezika to je jedan od najbitnijih elemenata, ukoliko postoji dovoljno primjera i videa kako se nešto radi, učenje samog programiranja postaje puno lakše i zabavnije. Ako je razvojno okruženje popularno, osim što ima puno primjera za korištenje, vrlo je i vjerojatno da će takav program biti održavan dugo godina. Održavanje samog softvera jednako je bitno kao i sam razvoj softvera. Sa vremenom svaki programski jezik dobiva unaprjeđenja, pa je isto tako bitno da i samo razvojno okruženje u kojemu radimo prati ta unaprjeđenja.

U nastavku navesti ću i objasniti nekoliko bitnijih razvojnih okruženja koja su popularna i korisna. Bitno je i spomenutu da skoro sva razvojna okruženja podržavaju rad sa više programskih jezika, tako da ponekad u istom razvojnom okruženju možemo raditi na nekoliko projekata u nekoliko tehnologija od jednom.

5.1.1. Dev-C++



Slika 7: Izgled Dev-C++ razvojnog okruženja [17]

Dev-C++ je besplatno razvojno okruženje za programiranje u jezicima C i C++. Razvijen je u programskom jeziku Delphi[4]. Ovo je razvojno okruženje koje smo koristili tijekom učenja programskog jezika C++ na fakultetu. Sama aplikacija je dosta stara te nije bila puno unaprjeđivana što se vidi i po samom izgledu grafičkog sučelja. No baš zbog te jednostavnosti daje dobre temelje početnicima za učenje samog programiranja u C ili C++ jeziku.

Veliki nedostatak ovog razvojnog okruženja je problem da originalni razvojni tim je odustao od daljnjeg razvoja 2006. godine. Te od tada su razvoj nastavili dva nezavisna tima: wxDev-C++ i Orwell. Problem sa time je što samo razvojno okruženje nije aktivno ažurirano, te je samo pitanje vremena kada više ne bude pratilo unaprijeđena koja se razvijaju na samom C i C++ jeziku. [18]

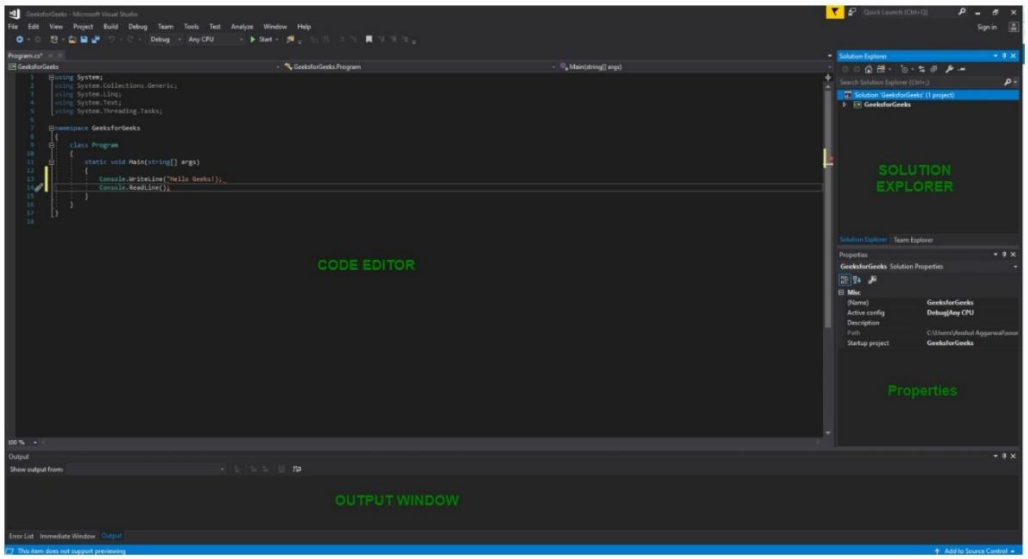
[4] Delphi – programski jezik baziran na programiranju pogonjenom događajima

5.1.2. Visual Studio

Visual Studio je jedan od najpopularnijih razvojnih okruženja današnjice. Razvijen je od Microsofta u svrhu pružanja razvojnog okruženja za razvoj konzolnih i web aplikacija, mobilnih aplikacija i raznih web servisa, desktop aplikacija i drugo. Iako se najčešće koristi za razvoj u C# programskom jeziku, sam Visual Studio podržava 36 različitih programskih jezika kao što su: C, C++, Java, JavaScript, PHP i drugi. [19]

Razvijen je 1997. godine te mu je glavna svrha bila podržavanje .NET Framework-a[5]. Danas Visual Studio ima tri različite verzije: Community, Professional i Enterprise. Community verzija je besplatna svim korisnicima. Sadrži skoro sve funkcionalnosti kao i profesionalna verzija ali ima određena ograničenja poput legalnog ograničenja da ukoliko vaša organizacija ima više od 250 računala i prihod veći od 1 milijuna dolara ne smijete koristiti ovu verziju programa. Profesionalna verzija je komercijalna verzija Visual Studia te glavna svrha te verzije je pružanje fleksibilnosti, produktivnosti i kolaboracije većim organizacijama i profesionalnim korisnicima. Enterprise verzija je namijenjena pružanju velike skalabilnosti i isporuke visoko kvalitetnih softvera. [19]

Visual Studio je alat koji se može preporučiti svakome. Sa bezbroj primjera korištenja na internetu do jednostavnog sučelja pruža odličnu platformu početnicima za učenje programskih jezika. Sa druge strane, skalabilnost i otvorenost raznim ekstenzijama ovo je odlično razvojno okruženje i za profesionalne korisnike.

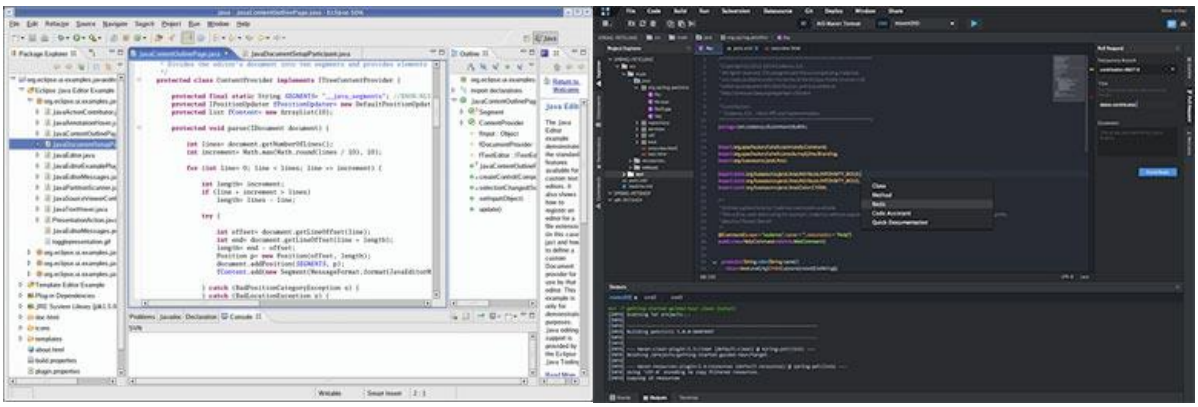


Slika 8: Izgled Visual Studio razvojnog okruženja [19]

[5] .NET Framework – developerska platforma sa alatima, bibliotekama i programskim jezicima sa ciljem kreiranja uniformnih proizvoda koji se mogu pokretati na raznim mjestima koja podržavaju .NET kod

5.1.3. Eclipse

Ukoliko želimo programirati u Javi, jedan od prvih izbora biti će Eclipse razvojno okruženje. Iako podržava i programiranje u C, C++, JavaScript/TypeScriptu te PHP i drugima, Eclipse je najpoznatiji po podršci razvoja u Javi. Pruža jednostavno i čisto moderno sučelje sa svim mogućnostima koje očekujemo od modernog razvojnog okruženja. U svakom slučaju sa odabirom Eclipse razvojnog okruženja se ne može pogriješiti.



Slika 9: Izgled Eclipse razvojnog okruženja [20]

5.1.4. Brackets

Jedan od alata za razvoj web stranica sa kojim sam se nedavno susreo je Brackets. To je tekst uređivač koji razumije web razvoj. Pruža mogućnost uvida u realnom vremenu, kako programer piše kod tako se u web pretraživaču stranica mijenja. Na taj način se uvelike olakšava razvoj web stranica. Iz tog razloga ovaj tekst uređivač je dobar izbor za rad sa PHP-om i JavaScript-om na raznim web stranicama.



Slika 10: Izgled Brackets tekstualnog uređivača [21]

5.2. Brzina izvođenja

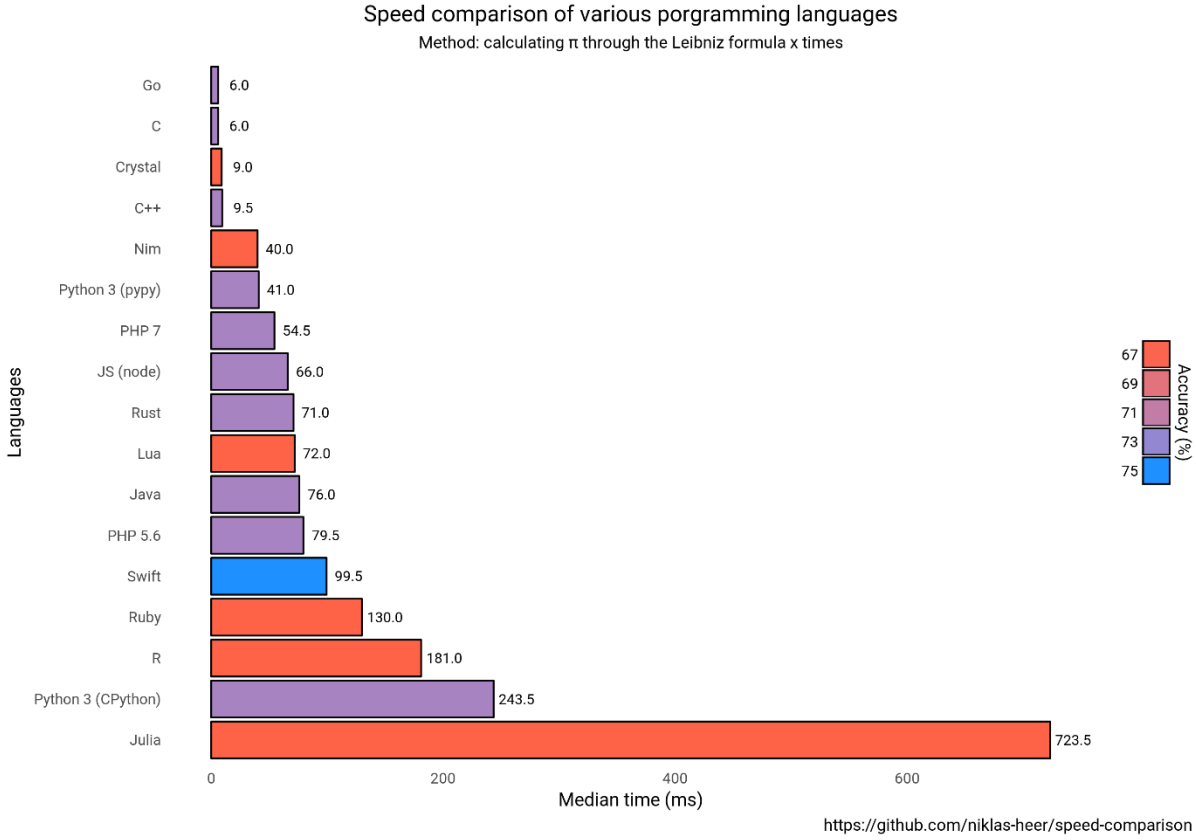
Mjerenje performansi programskog jezika može biti poprilično komplicirano. Ne postoji specifično decidirana procedura kojom možemo doći do specifikacija samog programskog jezika. Razlog tomu je sama raznolikost programskih jezika, te još veća raznolikost okruženja u kojima se ti programski jezici pokreću. Tako program pisan u C# ili bilo kojem drugom jeziku, se neće jednako brzo izvršavati na dva različita računala.

Usprkos tomu, postoje neke generalne činjenice na temelju kojih možemo reći da je jedan programski jezik brži od drugoga. Poput činjenice da C i C++ jezici su bili dizajnirani da budu apstraktnije verzije asemblera. Iz tog razloga kod napisan u C ili C++ jeziku je unaprijed kompajliran te jako blizak računalnom kodu. Java se ne može mjeriti sa brzinom izvođenja C ili C++ programa jer nije bila dizajnirana na taj način.

Pitanje je da li je sama brzina izvođenja programa presudan faktor u donošenju odluke koji programski jezik koristi. U današnje vrijeme, svi gore spomenuti jezici se izvršavaju brzo. Toliko brzo da razlike u izvršavanju ne predstavljaju bitan faktor većini programera. No postoji i mali postotak primjena gdje milisekunde imaju veliki utjecaj na korisnost samog programa. Aplikacije za banke i razni softveri za upravljanje burzom moraju biti vrlo brzi, vrijeme promjene tečajne valute između Londona i New Yorka može imati velikog značaja u takvoj profesiji. Isto tako aplikacije poput autopilota letjelica. Takvi softveri moraju biti brzi i pouzdani, te aplikacije takvog ranga prolaze godine i godine testiranja i mjerenja performansi kako bi se moglo sa sigurnošću reći da greške koje bi mogle ugroziti ljudske živote ne postoje.

5.2.1. Primjer provedenog istraživanja

Iako nikada ne možemo dobiti savršeno preciznu mjeru brzine programskoga jezika, postoji jedan način koji je relativno jednostavan a može pridonijeti uvelike u donošenju odluke o odabiru jezika. Najbolje je napisati određeni kod u programskim jezicima koje želimo testirati, a zatim te kodove izvršavati na istom računalu. Na taj način možemo vidjeti kako će se programski jezici ponašati na temelju koda koji je nama relevantan. A osim toga možemo vidjeti i ostale parametre poput same veličine datoteke koda i slično. Primjer jednog takvog istraživanja možemo vidjeti na slici 11. U ovom istraživanju napravljen je kod za računanje decimala broja pi temeljem Leibnizove formule.

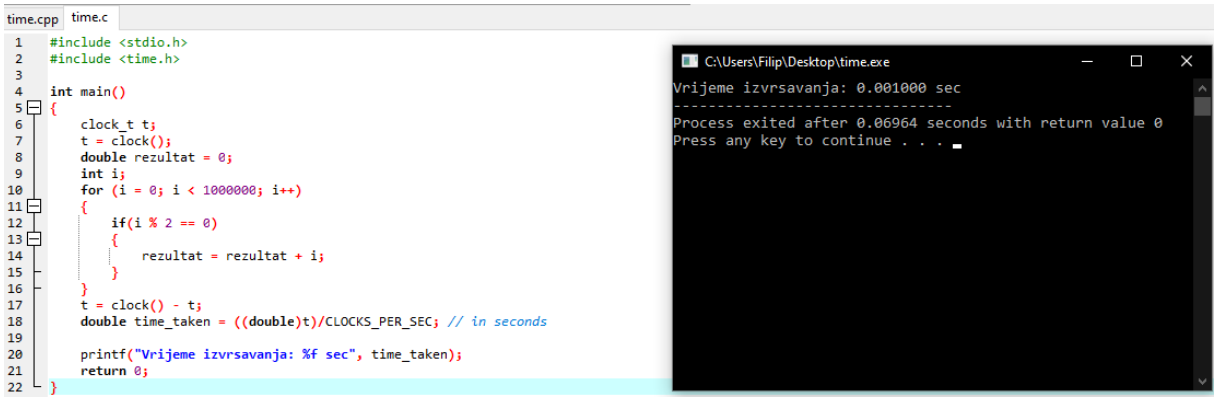


Slika 11: Brzina izvođenja programa za računanje pi decimala [22]

U rezultatima ovog istraživanja vidimo da uistinu C i C++ jezici vode sa brzinom izvođenja zbog njihovog arhitekturalnog dizajna. Te vidimo isto tako da JavaScript neznatno radi brže od PHP programskog jezika. Činjenica da u ovom istraživanju JavaScript radi brže od PHP-a nas vodi natrag na prije spomenute greške kod ovakvog načina mjerenja brzine. Iako uistinu JS je brži od PHP-a na istom računalu, JS se izvodi na strani klijenta dok se PHP izvodi na strani servera. U praksi to znači da sama brzina izvođenja uvelike ovisi o jačini samog servera ili jačine računala klijenta.

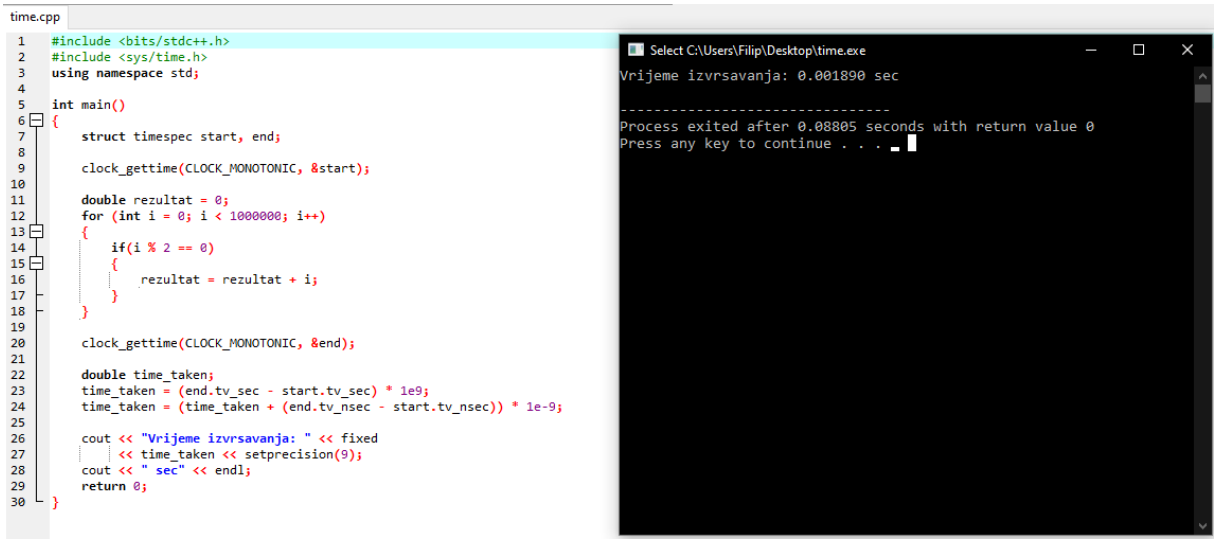
5.2.2. Vlastita usporedba brzine izvođenja

Kako bih izmjerio brzinu izvođenja ovih programskih jezika, osim što sam pronašao gore navedeno istraživanje usporedbe raznih programskih jezika. Odlučio sam se provesti i vlastito istraživanje na svom računalu. Kao operaciju za mjerenje brzine odabrao sam računanje zbroja svih parnih brojeva od 0 do 1000000. Iako je sama operacija zbrajanja vrlo jednostavna za odrađivanje programskome jeziku, samo ponavljanje operacije do milijun stvara dovoljno vremena kako bi samo mjerenje bilo precizno. Iako sama preciznost uvelike varira o samoj opterećenosti i snazi računala na kojemu se izvršava mjerenje.



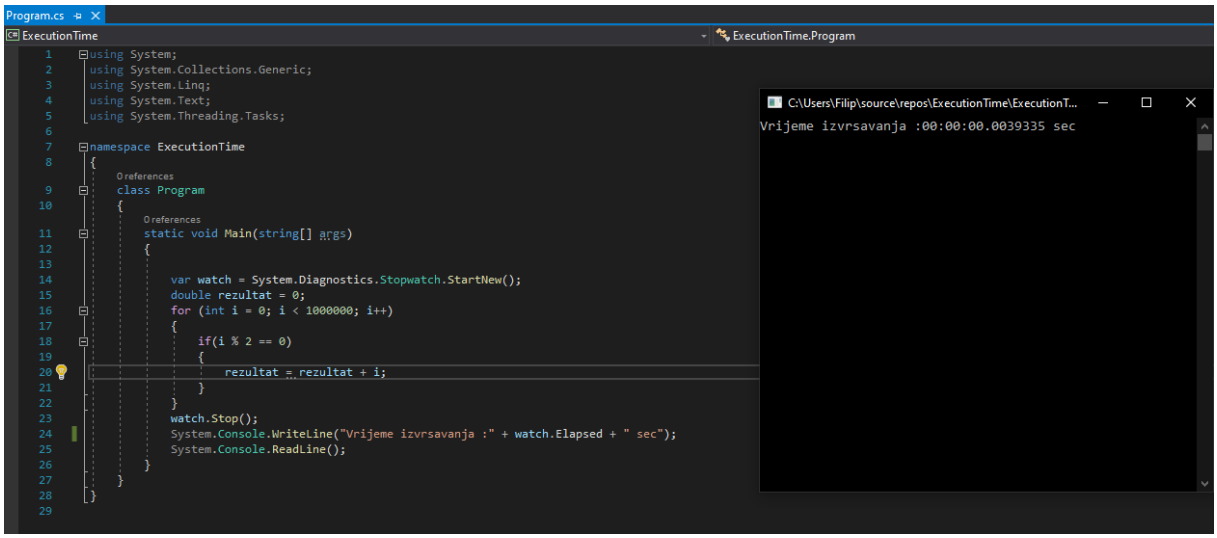
Slika 12: Brzina izvođenja programa u C programskom jeziku

Na slici 12 vidimo prikaz koda unutar C programskog jezika koji sam koristio za mjerenje brzine izvođenja. Koristio sam funkciju clock() koja se poziva za mjerenje trenutnog vremena točno prije početka izvođenja koda za zbrajanje parnih brojeva. Zatim se izvodi petlja zbrajanja od 0 do 1000000, a samim krajem petlje ponovno uzimamo vrijeme putem clock() funkcije, te odmah oduzimamo vrijeme početka. Zatim je to vrijeme potrebno pretvoriti u oblik double kako bi bio čitljiviji, te ga zatim ispisujem na ekran. U ovome primjeru vrijeme je zaokruženo na 3 decimale. Što znači da je za izvršavanje ovoga isječka bila potrebna 0.001 sekunda. Radi provjere točnosti mjerenja izvršio sam još nekoliko kontrolnih mjerenja te su vrijednosti varirale od 1 milisekunde do 2 milisekunde.



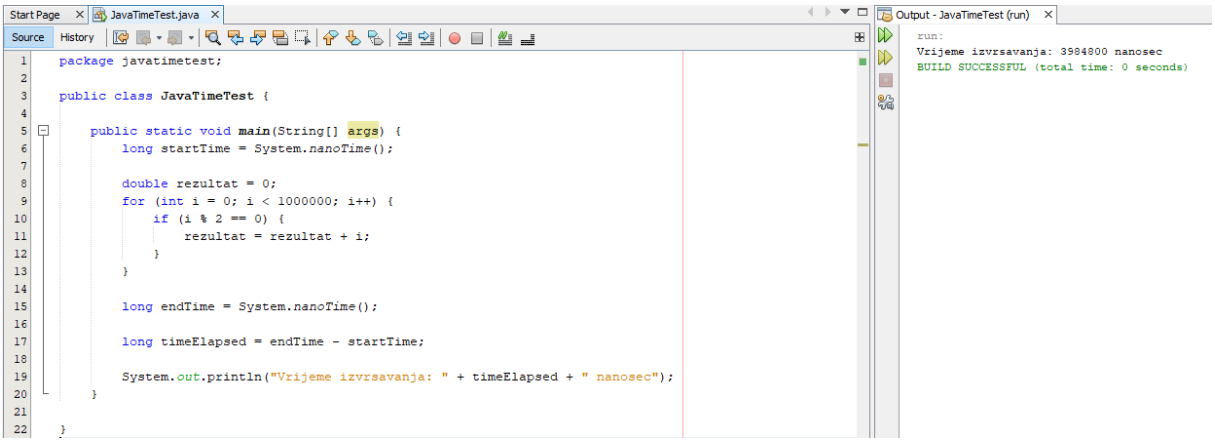
Slika 13: Brzina izvođenja programa u C++ programskom jeziku

Na slici 13 prikazano je mjerenje brzine izvođenja unutar C++ programskog jezika. Kao i kod prijašnjih primjera za glavni isječak koda koji mjerimo koristim računanje zbroja parnih brojeva, a za samo mjerenje vremena koristim clock_gettime metodu koju pozivam prije samog izvršavanja i nakon izvršavanja petlje. Zatim opet oduzimam početno vrijeme od krajnjeg kako bi dobio točno trajanje u sekundama. U mjerenju sa primjera izvršavanje je trajalo 0.001890 sekundi. Mjerenja sam opet ponovio nekoliko puta kako bih dobio točniju vrijednost, te su sva mjerenja bila u rasponu od 0.001820 sekundi i 0.001909 sekundi.



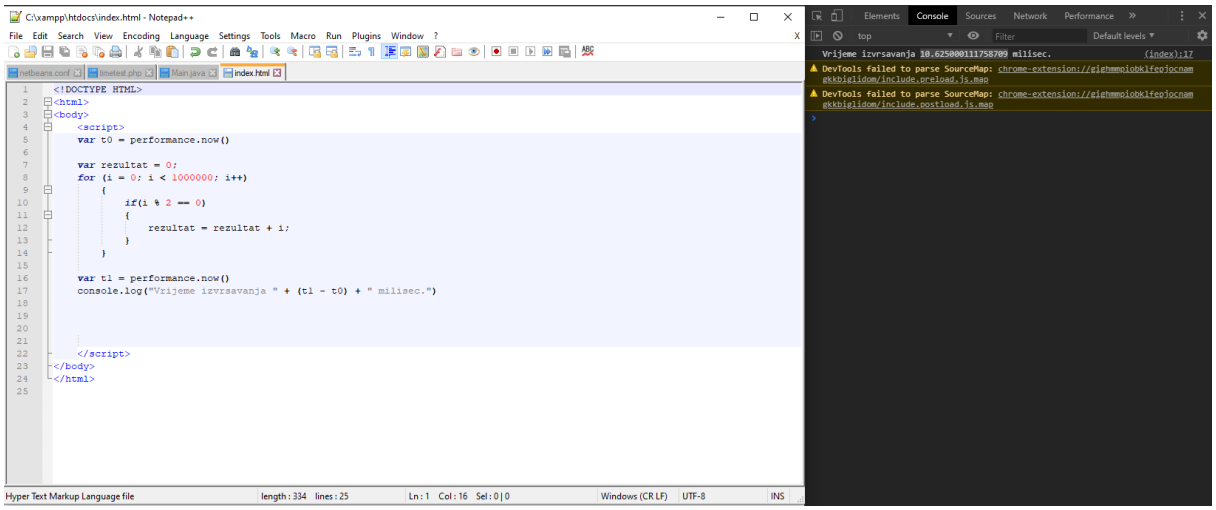
Slika 14: Brzina izvođenja programa u C# programskom jeziku

Na slici 14 može se vidjeti prikaz koda za mjerenje brzine izvršavanja pisan u C# programskome jeziku. Radi konzistentnosti korišten je isti isječak koda za računanje zbroja parnih brojeva. Za mjerenje brzine izvođenja korištena je Stopwatch metoda iz System.Diagnostics biblioteke. Ta metoda radi na principu štoperice, pokretanjem kreće mjerenje vremena, a nakon izvršavanja koda se mjerenje zaustavlja te nije potrebno oduzimati početno od krajnjeg vremena. Vrijeme proteklo u ovome primjeru bilo je 0.0039335 sekundi, a radi konzistentnosti ostala mjerenja bila su u rasponu od 0.0037542 i 0.0042060 sekundi.



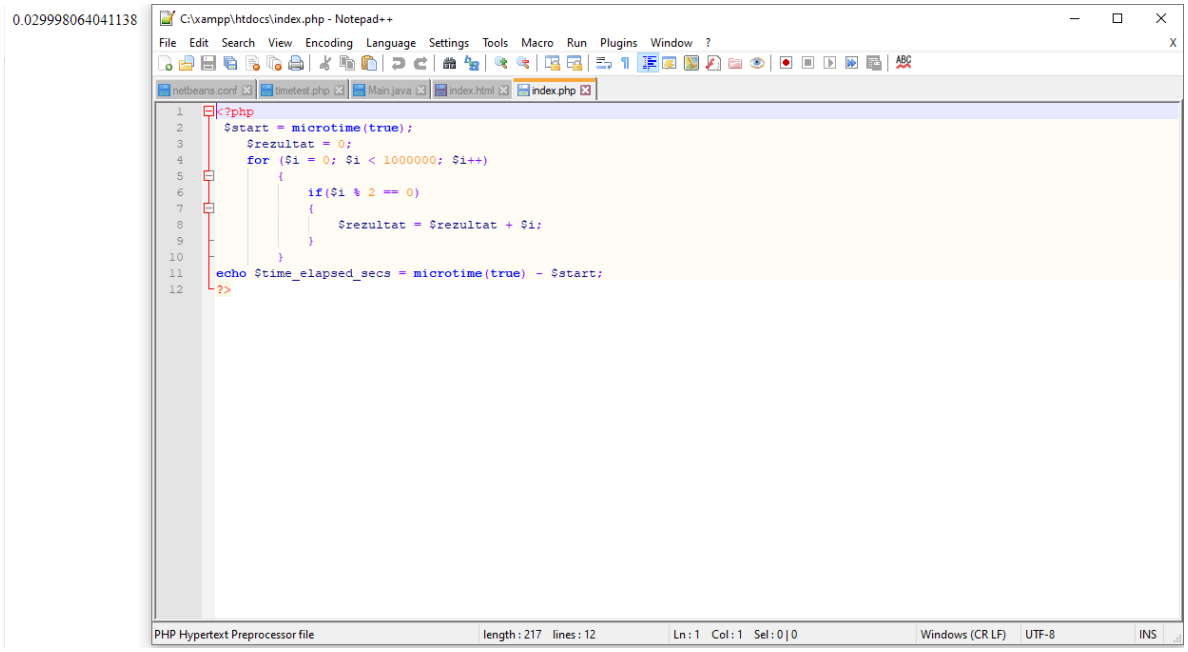
Slika 15: Brzina izvođenja programa u Java programskom jeziku

Nadalje na slici 15 vidimo prikaz mjerenja brzine izvođenja pisanog u Java programskom jeziku. Iako postoji nekoliko načina kao i do sada, unutar Java programskoga jezika odlučio sam se koristiti System.nanoTime() metodu. Metoda radi na principu sličnom C i C++ programskom jeziku. Na početku se uzme trenutno vrijeme, na kraju se opet uzme trenutno vrijeme te se onda oduzima početno od krajnjeg vremena kako bi se dobilo proteklo vrijeme u nanosekundama. Mjerenje u ovome primjeru je bilo 3984800 nanosekundi, što nakon konverzije ispadne 0.0039848 sekundi. Ostala kontrolna mjerenja bila su u rasponu od 0.003794 i 0.004209 sekundi.



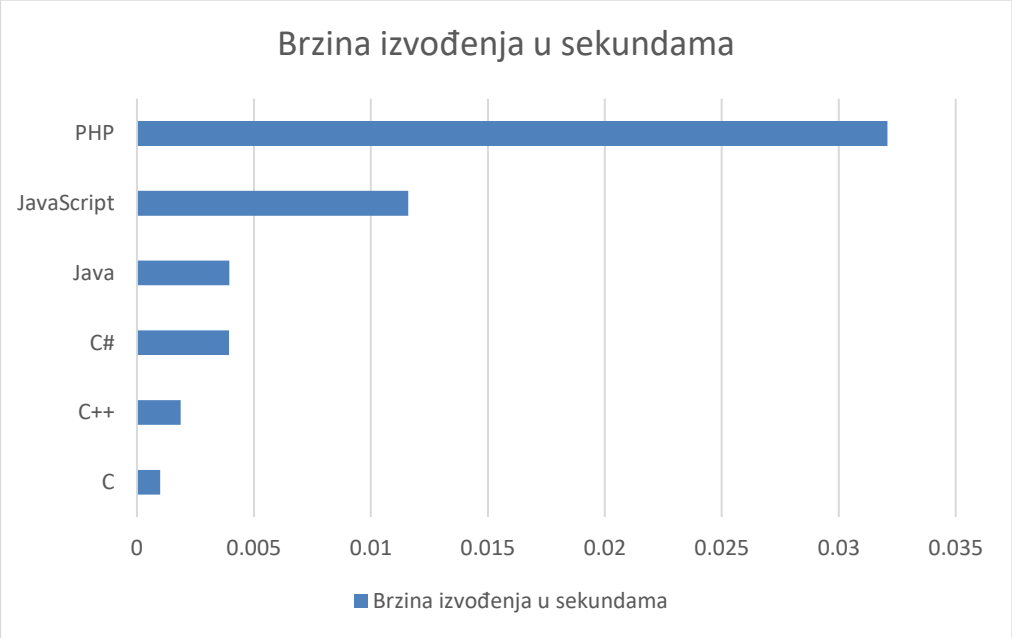
Slika 16: Brzina izvođenja programa u JavaScript programskom jeziku

Slika 16 prikazuje kod za mjerenje brzine izvođenja zbrajanja parnih brojeva pisanog u JavaScript programskom jeziku. Metoda koju sam odlučio koristiti je performance.now() i radi kao i većina ostalih metoda koje sam koristio. Prije samog izvršavanja petlje uzme se trenutno vrijeme, nakon petlje uzme se trenutno vrijeme. Te se početno vrijeme oduzme od krajnjeg i dobijemo proteklo vrijeme u milisekundama. U ovome mjerenju vrijeme je bilo 10.625 milisekundi, što nakon konverzije u sekunde ispada 0.010625 sekundi. Ostala kontrolna vremena su u rasponu od 0.010625 i 0.013549 sekundi. Već na prvu možemo uočiti veliku razliku naspram prijašnjih vremena, no više o samoj usporedbi malo kasnije.



Slika 17: Brzina izvođenja programa u PHP programskom jeziku

Za kraj slika 17 prikazuje provedeno mjerenje brzine izvođenja koda pisanoga u PHP programskome jeziku. Metoda za mjerenje vremena unutar PHP programskoga jezika je microtime(true). Sam princip mjerenja ostaje isti, uzimamo vrijeme prije i nakon izvršavanja a zatim oduzimamo razliku. Te u ovome primjeru dobivamo vrijeme izvršavanja od 0.029998 sekundi. Ostala kontrolna mjerenja bila su u rasponu od 0.029998 do 0.035839 sekundi.



Slika 18: Graf brzine izvođenja temeljem vlastitog mjerenja

Kao generalnu usporedbu na slici 18 možemo vidjeti graf generalne usporedbe brzine izvođenja. Mjere prikazane na grafu predstavljaju prosječne mjere dobivene temeljem nekoliko uzastopnih mjerenja. Tako za C prosjek izvršavanja je bio 0.001 sekundi, za C++ prosjek je 0.001866 sekundi, za C# bilježimo prosjek od 0.003930 sekundi a za Javu je prosjek 0.003950 sekundi. Sa druge strane JavaScript ima prosjek od 0.01160 sekundi a PHP 0.032088 sekundi.

Na temelju ovih rezultata možemo primijetiti sličnosti i sa gore navedenim istraživanjem. C kao jezik koji je najbliži jeziku samoga računala je prvi po brzini izvođenja te ga vrlo blizu prati C++. Java i C# su oboje jezici koji su više razine, ali i dalje vrlo brzi te gotovo podjednaki u brzini izvršavanja. Problem možemo vidjeti sa JavaScript i PHP jezicima gdje možemo vidjeti da je JavaScript znatno sporiji, a PHP možda i ne prikazuje realnu vrijednost. Kako znamo izvršavanje PHP koda ovisi o serveru na kojem se izvršava. Kako sam ja izvršavanje provodio na lokalnome računalu korištenjem virtualnog XAMPP Apache servera vrlo je vjerojatno da je sama brzina limitirana iz tog razloga.

5.3. Lakoća učenja

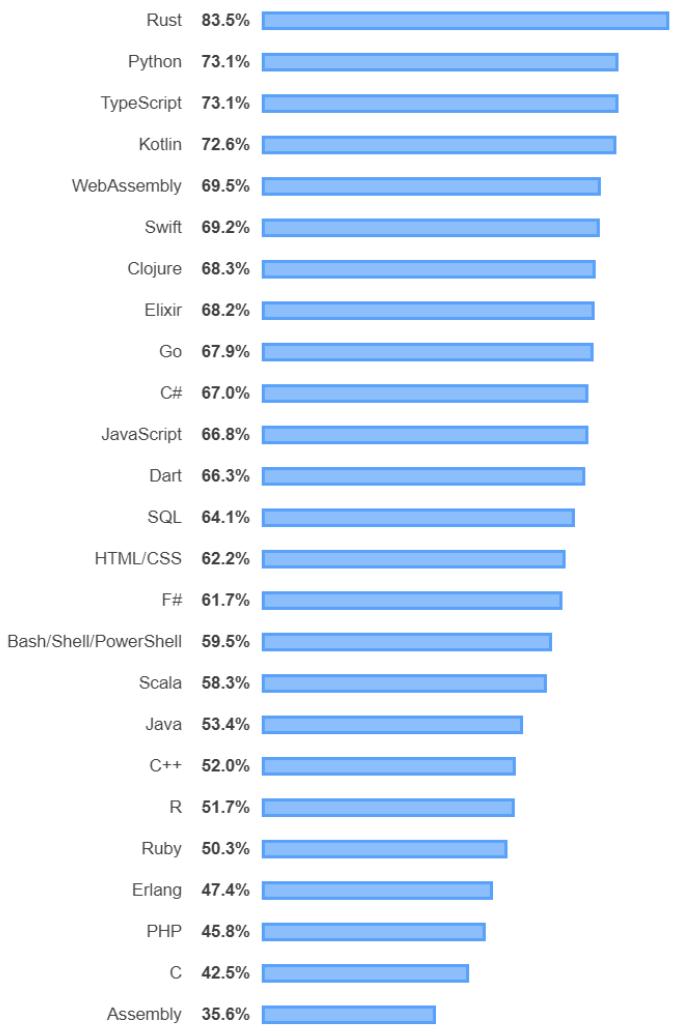
Lakoća učenja jezika je poprilično subjektivna karakteristika jezika. Iako postoje generalni elementi i ankete po kojima možemo donekle rangirati programske jezike, samo učenje istih može uvelike varirati zbog interesa i pred znanja individualne osobe. Što se tiče jezika kojima se bavim u ovome radu, oni su svi vrlo slične sintakse. Samim time, poznavanje jednog programskog jezika uvelike pomaže u učenju drugog.

Iako većina izvora na internetu svrstava JavaScript kao najlakši programski jezik za naučiti od ponuđenih šest koje obrađujem u ovome radu, ja se sa time ne bi složio iz nekoliko razloga. Kao prvo JavaScript iako je danas vrlo zastupljen jezik i vrlo koristan za naučiti, po mojemu mišljenju nije nimalo lagan za savladati. Razlog tomu je činjenica da je JavaScript loš primjer objektno orijentiranog programiranja jer se JavaScript uglavnom koristi za male isječke kodova kako bi web stranica dobila određene funkcionalnosti. Osim toga, greške u kodu JavaScripta neće biti prikazane dok se sam kod ne pokrene.

Po mojemu mišljenju, najlakše je naučiti programirati u C# ili Java programskim jezicima. Oba jezika daju dobar primjer objektno orijentiranog programiranja. Oba jezika imaju jako dobra razvojna okruženja koja pružaju odlične alate za početnike, te postoji jako puno primjera koda i raznih drugih tutoriala i videa putem kojih se lagano može krenuti programirati. Nakon C# i Jave svrstao bi C i C++. Kako su ti jezici bliži asembleru, sami koncepti koji su potrebni sa savladavanje programiranja u takvim jezicima mogu biti poprilično komplicirani početnicima. Učenje pokazivača je tema koju svakog početnika zbuni iz prve. No sa druge strane, ukoliko početnik savlada takve koncepte, kasnije će mu biti puno lakše savladati druge programske jezike jer će imati puno dublje znanje principa programiranja.

5.3.1. Primjer provedenog istraživanja

Tijekom istraživanja lakoće učenja naišao sam na provedeno istraživanje gdje se anketiralo okvirno 90.000 programera širom svijeta o najpopularnijim i najdražim programskim jezicima. Istraživanje je provedeno na stranici Stack Overflow a rezultate možete vidjeti na slici ispod.



Slika 19: Brzina izvođenja programa u PHP programskom jeziku

U rezultatima ovoga istraživanja vidimo kako je C# najdraži programski jezik od naših šest kojima se bavimo u ovome radu. JavaScript je vrlo blizu na drugom mjestu, a zatim slijede Java, C++, PHP i C. Osim što ovo istraživanje pokazuje koji jezici su najdraži u profesiji programiranja, ovo ima veliki utjecaj i na samu lakoću učenja. Kako je lakoća učenja vrlo subjektivna tema, vrlo često je uspješnost učenja povezana sa zainteresiranosti za samu temu. Temeljem toga možemo zaključiti da jezici koji su generalno draži programerima, su isto tako i lakši za učenje početnicima.

6. Usporedba sintakse

Tema ovoga poglavlja obuhvaća sintaksu pojedinih programskih jezika. Baviti ću se sintaksom pisanja osnovnih naredbi unutar svakog jezika zasebno. Pojasniti ćemo kako se deklariraju varijable, kako pisati komentare te ćemo proći kroz pisanje osnovnih naredbi za kontrolu toka izvršavanja programa. Sintaksa na ovoj razini pokazati će ključne sličnosti i razlike u sintaksi ovih šest programskih jezika.

6.1. Osnovne razlike u sintaksi

Kada krenemo programirati u novom programskom jeziku, prva stvar koju ćemo naučiti raditi u tom programskom jeziku je deklariranje varijabli. Varijable služe za rad sa podatcima te njihovo privremeno spremanje tijekom izvršavanja samoga programa. Točnije rečeno, varijable predstavljaju imenovani dio memorije sa kojom naš program raspolaže. Ovisno o tipu varijable program zauzima više ili manje mjesta za tu varijablu. Prije nego objasnimo kako se varijabla deklarira u pojedinom programskom jeziku potrebno je razumjeti osnovne tipove varijabli sa kojima raspolažemo.

Tablica 2: Prikaz glavnih tipova varijabli

TIP VARIJABLE	VELIČINA (BYTES)	RASPON
short int	2	-32,768 do 32,767
int	4	-2,147,483,648 do 2,147,483,647
long	8	$-2^{63}\ do\ 2^{63}-1$
float	4	
double	8	
TIP VARIJABLE	**OPIS**	
string	Varijabla koja sadrži alfanumerički tekst	
boolean	Varijabla koja sadrži true ili false	
char	Varijabla koja sadrži jedan znak	

Svaka od ovih varijabli ima signed i unsigned verziju. Signed verzije su navedene u tablici te one uzimaju negativni i pozitivni raspon. Unsigned verzija varijable zanemaruje negativan raspon varijable, te taj dio memorije prebacuje u pozitivan raspon varijable. Tako unsigned integer ima raspon od 0 do 4,294,967,295. To je vrlo korisno ukoliko su nam potrebne velike pozitivne brojke. Umjesto da trošimo polovicu memorije na negativan raspon sa kojim znamo da ne budemo radili, iskoristimo taj dio memorije za veći pozitivni broj.

6.1.1. Deklariranje varijabli

Kada razumijemo tipove varijabli sa kojima raspolažemo, potrebno je deklarirati željenu varijablu. C, C++, C# i Java imaju jednak način deklariranja varijable. Prvo se definira tip željene varijable, zatim naziv same varijable. Nakon naziva same varijable možemo joj dodijeliti vrijednost, ali isto tako možemo ju ostaviti praznu te joj vrijednost dodijeliti kasnije u kodu. Primjer sintakse definiranja varijable izgleda ovako:

```
type name = value;
```

Konkretan primjer definiranja varijable tipa integer sa dodijeljenom vrijednosti 10:

```
integer broj = 10;
```

Te je isto moguće napraviti na ovaj način:

```
integer broj;
broj = 10;
```

Ovakav princip rada sa varijablama ne vrijedi za JavaScript i PHP. U ta dva programska jezika rad sa varijablama se dešava na drugačiji način. Programer sam ne odabire tip varijable nego pri samom definiranju varijable kada se unese vrijednost, program sam prepoznaje kojeg tipa je ta vrijednost te dinamički dodjeljuje tip varijable toj vrijednosti. JavaScript kod definiranja varijable koristi ključnu riječ var, a program sam prepoznaje da u primjeru ispod, varijabla godina je tipa integer:

```
var godina = 2020;
```

Zbog ovakvog načina deklariranja varijabli, postoji nekoliko stvari na koje treba paziti tijekom programiranja. Poput deklariranja varijable u kojoj se koriste aritmetički izrazi, program izraze čita sa lijeva prema desno te na taj način tretira tipove podataka. Tako na primjer ako za deklariranje varijable prvo želimo zbrojiti dva integera a zatim nadopisati string na kraj,

program prvo čita dva broja koja su integer a zatim ih zbraja, a onda dolazi do podatka tipa string koji nadopisuje na prvotni rezultat. Tako finalna varijabla postaje tipa string. Varijabla x postaje tipa string te sadrži vrijednost 77. Jer se prvo čitaju integer brojevi koji se zbrajaju, a zatim se dolazi do stringa koji se nadopisuje.

```
var x = 5 + 2 + „7“;
```

Varijabla x ostaje tipa string te sadrži vrijednost 527.

```
var x = „5“ + 2 + 7;
```

Na istom principu funkcionira i PHP programski jezik, koji se jedino razlikuje u sintaksi kojom se varijabla deklarira. Unutar PHP-a ne postoji ključna riječ var, već se koristi simbol $.

```
$broj = 5;
$tekst = „Hello, World!“;
```

6.1.2. Deklariranje konstanti

Vrlo često osim običnih varijabli, potrebne su nam varijable koje želimo da služe kao konstante tijekom izvršavanja programa. Konstante omogućuju dodavanje varijable koja se ne mijenja tijekom izvršavanja, a samu korist takvih varijabli možemo vidjeti u nekoliko slučajeva. Korištenjem konstanti naš kod postaje puno čitljiviji te lakši za razumijevanje i kasnije promjene. Na primjer postavljanje varijable pdv-a deklariramo kao konstantu od 0.25, te tu varijablu kasnije koristimo na više mjesta. Samim time olakšavamo razumljivost toga koda, a isto tako olakšavamo buduće promjene ukoliko želimo promijeniti postotak pdv-a. Samom konstantom osiguravamo jednu razinu sigurnosti od pogreške, tako u našem računu gdje koristimo konstantu pdv-a možemo biti sigurni da se ta vrijednost ne bude slučajno promijenila.

Deklariranje konstanti podržavaju svih šest programskih jezika koje uspoređujem u ovome radu. Sama sintaksa u tim jezicima izgleda ovako:

C i C++: `#define pdv 0.25`

C#: `public const double pdv = 0.25;`

Java: `static double pdv = 0.25;`

JavaScript: `const pdv = 0.25;`

PHP: `define(„pdv“, 0.25);`

Iz ovog primjera možemo uočiti da razlike između jezika C i C++ nema, u tim jezicima konstanta se deklarira na vrhu slično kako se uključuje nova biblioteka u kod. Sa druge strane

C#, Java i JavaScript deklariraju konstante kao varijable sa ključnom riječi const za C# i JavaScript dok Java koristi static. Jedina razlika je poznata činjenica da JavaScript ne deklarira tip varijable već se on dodjeljuje dinamički. PHP se na prvi pogled najviše ističe sa metodom define kojoj se u zagradi prvo prosljeđuje naziv varijable a zatim vrijednost varijable.

6.1.3. Deklariranje polja i listi

Rad sa osnovnim tipovima varijable nije dovoljan za obavljanje velike većine programerskih zadaća. Zbog toga su nam potrebna polja i liste. Polja su jednostavnije verzije vezanih listi jer se unaprijed definira veličina samoga polja. Sintaksa izgleda ovako:

C i C++ `int polje[20];`

C# `int[] polje = new int[20];`

Java `int[] polje = new int[20];` ili `int polje[] = new int[20];`

JavaScript `var polje = [];`

PHP `$polje = array();`

Razlike u sintaksi su poprilično velike kada se uzme u usporedbu sa osnovnim deklariranjem varijabli. Dok C i C++ dijele isti princip deklariranja polja, C# i Java već imaju drugačiji princip. Dok Java podržava dva načina, C# podržava samo jedan. A JavaScript i PHP kao i uobičajeno do sada, imaju skroz drugačiji princip definiranja polja.

Sa druge strane rad sa vezanim listama predstavlja veći problem kod C i C++ jezika nego kod Jave i C#, dok PHP i JavaScript u svojoj osnovi ne podržavaju koncept vezanih lista. C vezane liste se mogu napraviti samo putem pokazivača, dok C++ podržava biblioteku za rad sa listama.

Primjer definiranja i unošenja elemenata u vezanu listu unutar C++ programa [24] :

```
struct element {
	int vrijednost;
	element *sljedeci;
};
typedef element *lista;
int main(){
	lista L = NULL;
	element *el;

	int A;
	do{
		do{
			cout << „A = „;
			cin >> A;
		} while (A < 0);
		if (A > 0) {
			el = new element;
			(*el).vrijednost = A;
			(*el).sljedeci = L;
			L = el;
		}
	} while (A > 0;
}
```

Sa druge strane ukoliko se odlučimo na jednostavniji pristup korištenja biblioteke <list> sintaksa izgleda ovako:

```
list <int> mojaLista;

mojaLista.push_front(1);
```

Još jednostavniji princip rada sa listama možemo vidjeti u C# i Javi gdje se liste deklariraju ovako:

C#: `List<int> mojaLista = new List<int>();`

Java: `List<int> mojaLista = new ArrayList<int>();`

Iz ove usporedbe možemo vidjeti najradikalniju razliku u sintaksi ovih programskih jezika. C# i Java su najbliži sintaktički, te zahtijevaju najmanje prilagodbe sa jednog na drugi jezik. Dok ostali jezici imaju razne finese u kojima se uvelike razlikuju pa i u naizgled jednostavnim konceptima poput vezanih listi.

6.1.4. Pisanje komentara

Osim deklariranja varijabli, iako manje važno za samu funkcionalnost programa ali podjednako važno za razumljivost i korisnost programa su komentari. Pisanje komentara u kodu se može koristiti za puno stvari, od jednostavnih podsjetnika da se dio koda mora nadopuniti do cijele dokumentacije koja se piše po raznim standardima kao opis dijela koda. Komentari u kodu postaju itekako važni u velikim sustavima sa velikim timovima programera gdje nakon što se kod napiše, isti dio će se morati refaktorirati više puta od strane više ljudi. Čitanje tuđeg koda bez dokumentacije je jako težak posao.

U C programskom jeziku pisanje komentara je bilo moguće unutar /* */ simbola. Komentar je mogao biti jedna linija koda ili više linija koda. Te je to bio jedini način za pisanje samog komentara.

```
/* Komentar u C kodu */

/*
* Više linijski komentar u C kodu
*/
```

Razvojem C++ jezika, pisanje komentara je dobilo novi način pisanja jednolinijskih komentara koji se zadržao sve do danas. Isti princip pisanja komentara koriste i C#, Java, JavaScript i PHP. Ukoliko se želi napisati jednolinijski komentar koristi se simbol // dok pisanje vise linijskih komentara ostaje isto kao i kod C programskoga jezika.

```
// Jednolinijski komentar u C++, C#, Java, JavaScript i PHP kodu

/*
Više linijski komentar u C++, C#, Java, JavaScript i PHP kodu
*/
```

Jedinu razliku u pisanju komentara predstavlja PHP programski jezik, koji uz gore navedene načine pisanja komentara sadrži još jedan način za pisanje jednolinijskoga komentara koristeći simbol #.

```
# Jednolinijski komentar u PHP kodu
```

6.2. Usporedba primjera ispisa teksta na ekran

Kada nekome želimo pokazati osnovni primjer programa u bilo kojem programskom jeziku, svima pada na pamet „Hello, World!“. To je program koji ispisuje jednu liniju teksta na ekran, te je koristan za prikazivanje najosnovnije operacije i principa kodiranja u većini programskih jezika. Osim što ispis teksta na ekran predstavlja savršen uvod u programiranje za početnike, ovaj program je koristan i za iskusnije programere kako bi provjerili da li sam programski jezik radi kako bi trebao. Ukoliko nakon instalacije programskog jezika ne možemo prikazati niti osnovni ispis teksta na ekran, očito je da niti kompleksniji zadatci se ne budu izvršavali unutar programskog jezika te je prvo potrebno otkloniti poteškoće i ispravno izvršiti instalaciju samog programskog jezika. Iz toga razloga, prva usporedba koda programskih jezika obuhvaćenih ovom radu biti će upravo primjer „Hello, World!“ ispisa.

Započeti ćemo sa primjerom „Hello, World!“ programa pisanog u C programskom jeziku:

```
#include <stdio.h>
int main() {
     printf(„Hello, World!“);
     return 0;
}
```

Analizirajući sintaksu ovoga koda možemo uočiti nekoliko važnih koncepata programiranja unutar C programa. Tako na početku koda uočavamo ključnu riječ #include putem koje uključujemo biblioteku za standardni input i output programa. Drugim riječima, dajemo programu mogućnost prepoznavanja ključnih riječi za ispis teksta na ekran. Osim ispisa omogućavamo programu i za čitanje teksta koji bi korisnik mogao unijeti u program putem ključne riječi scanf() no to se u ovom primjeru koda ne koristi. Ukoliko bi pokušali koristiti printf naredbu bez korištenja stdio.h biblioteke, program se ne bi izvršio jer ne bi znao prepoznati navedenu naredbu. Nakon što smo uključili željenu biblioteku, potrebno je kreirati funkciju unutar koje će se izvršavati naš kod. Main funkcija predstavlja glavnu funkciju programa koja se uvijek izvršava prva. Unutar main funkcije pišemo naš program ili pozivamo druge funkcije u željenom redoslijedu. U osnovnom primjeru ispisa teksta na ekran, pozivamo funkciju za ispis teksta printf te u zagradu te funkcije kao parametar prosljeđujemo željeni tekst za ispis. Primijetimo da je naša main funkcija tipa int (integer) što znači da se očekuje da ta funkcija vraća podatak tipa integer. Integer je programski tip podatka za brojeve, a u ovome slučaju vraća se 0 pomoću return ključne riječi. 0 predstavlja status kraja programa te se na taj način daje do znanja programu da je završio.

Slijedi primjer „Hello, World!“ programa pisanog u C++ programskom jeziku:

```
#include <iostream>
int main() {
     std::cout << „Hello World!“;
     return 0;
}
```

Uspoređujući ovaj isječak koda sa prijašnjim, možemo vidjeti da C i C++ imaju puno sličnosti što je i logično za očekivati. Program započinje sa uključivanjem biblioteke na isti način kao i u C jeziku, razlika ovoga puta je u nazivu biblioteke, gdje je C koristio stdio.h, C++ koristi iostream. Funkcija ove biblioteke je ista kao i kod prošle, a to je uključivanje opcije ispisa teksta na ekran. Zatim slijedi deklaracija main funkcije na isti način kao i prije. Main funkcija je tipa integer, princip toga nam je već poznati jer i C++ program zahtjeva povratni status programa koji je u ovom slučaju 0. Glavna razlika u ovome kodu je način na koji se tekst ispisuje na ekran. Prva stvar koju treba primijetiti je prefiks std:: koji se može ali i ne mora pisati ispred svake linije. On služi za odabir iz koje biblioteke želimo koristiti naredbu, u ovom slučaju govorimo programu da želimo pozvati naredbu cout iz biblioteke ili točnije (*eng. namespacea)* zvanog standard library. Drugi način je da se na vrh programa ispod deklariranja biblioteke, deklarira namespace std. Na taj način predefiniramo da koristimo std kao osnovu rada naredbi pa nije potrebno definirati ispred svake linije posebno. U nastavku vidimo da se ovoga puta tekst unosi nakon << simbola, za razliku od C načina rada koji tekst unosi kao parametar funkcije. Na prvu ovakav način izgleda kompliciranije, no razlog tomu je logičan. Ukoliko želimo nešto ispisati na ekran, cout dio naredbe gledamo kao ekran a strelice nam govore da tekst sa desne strane naredbe želimo prikazati na lijevom dijelu (na ekranu). Sa druge strane ukoliko korisnik nešto treba upisati, a mi to želimo spremiti u varijablu tekst, program bi izgledao ovako: cin >> tekst; Na taj način logički vidimo da tekst unesen od strane korisnika na ekranu sa lijeve strane želimo putem strelica unijeti u našu varijablu na desnoj strani. Iako je ovaj način malo kompliciraniji, lako se može vizualizirati te naučiti kao dio sintakse, te kao takav ne otežava programiranje unutar C++ jezika.

Sada ćemo pokazati primjer „Hello, World!“ programa pisanog u C# jeziku:

```
namespace HelloWorldPrimjer
{
     class HelloWorld{
             static void Main(string[] args)
             {
                     System.Console.WriteLine(„Hello, World!“);
             }
     }
}
```

Uspoređujući ovaj isječak koda sa prijašnja dva, već možemo uočiti veliku razliku u sintaksi. Za početak unutar C# programa moramo definirati namespace unutar kojega radimo. Namespace možemo gledati kao prostor unutar kojega kreiramo druge klase. Razlog tomu je činjenica da je C# u cijelosti objektno orijentirani programski jezik, a namespace nam omogućava lakše snalaženje u kodu sa puno klasa. Govoreći o klasama, možemo vidjeti kako se u drugoj liniji primjera definira klasa HelloWorld. Kao što sam već spomenuo, C# nas tjera da definiramo klasu i za pisanje jednostavnog koda poput HelloWorld programa. Sav kod se mora nalaziti unutar klasa kako bi se održao koncept objektno orijentiranog programiranja, za razliku od C jezika koji uopće ne podržava OOP, i za razliku od C++ koji podržava OOP ali podržava i proceduralno programiranje te nas ne tjera da koristimo klase. Nastavljajući dalje vidimo da je potrebno definirati main metodu. Isto kao i u prijašnjim primjerima main metoda predstavlja početnu točku programa. Ovoga puta main metoda sadrži i argument tipa string polja args. To služi za rad sa ulaznim parametrima koje možemo postaviti pri pokretanju programa, u ovome primjeru je to nebitno. Unutar metode nalazi se naredba za ispis teksta na ekran. Ovoga puta je sličnija principu rada u C više nego C++ jer je to opet funkcija kojoj se prosljeđuje parametar teksta. Nema rada sa << i >> simbolima koji su predstavljali zbunjujuće probleme u C++ sintaksi.

Slijedi primjer „Hello, World!“ programa pisanog u Javi:

```
class HelloWorld{
     public static void main(String[] args){
             System.out.println(„Hello, World!“);
     }
}
```

Može se primijetiti da je primjer koda pisan u Javi jako sličan C# primjeru. Znamo od prije da su i Java i C# noviji jezici te su vrlo slični u sintaksi. To je vrlo korisno za početnike koji uče oba jezika jer se vrlo lako priviknuti sa jednog jezika na drugi. Isto kao i kod C# primjera, potrebno je kod pisati unutar definirane klase. Unutar same klase ponovno je potrebno definirati main metodu koja isto sadrži polje argumenata za rad sa parametrima pri pokretanju programa. Dok unutar same main metode nalazimo liniju za ispis na ekran. Po strukturi je slična prijašnjoj. Isto se poziva System kao glavna naredba, te postoje samo minimalne razlike u nazivu daljnjih naredbi. Sam tekst za ispis se prosljeđuje kao argument.

Sljedeći primjer obuhvaća „Hello, World!“ program pisan u JavaScript programskome jeziku, samim time vidjet ćemo da postoji velika razlika u samoj sintaksi:

```
<script>
     alert('Hello, World!');
</script>
```

Na prvi pogled JavaScript primjer izgleda puno drugačije od prijašnjih primjera, štoviše izgleda puno jednostavnije. Jedan od razloga tome smo već spomenuli ranije, JavaScript a kasnije ćemo vidjeti i PHP su jezici koji se generalno ne koriste za razvoj funkcionalnih aplikacija u cijelosti. Njihova svrha je nadopunjavanje, točnije obavljanje rutinskih operacija unutar već postojeće aplikacije ili web stranice. Tako primjer koda koji vidimo bi se u realnoj situaciji nalazio unutar html dokumenta poput ovoga:

```
<!DOCTYPE HTML>
<html>
<body>
     <script>
            alert('Hello, World!');
     </script>
</body>
</html>
```

Sama sintaksa JavaScript programa za ispis teksta na ekran je vrlo jednostavna, početak JavaScript skripte otvaramo sa <script> oznakom, a zatvaramo sa </script>. Na taj način dajemo do znanja da se unutar tih oznaka nalazi JavaScript kod. Sama naredba za ispis teksta je alert kojoj prosljeđujemo tekst kao parametar isto kao i u prijašnjim primjerima.

Zadnji primjer „Hello, World!“ programa koji obrađujemo je pisan u PHP-u:

```
<?php
     echo „Hello, World!“;
?>
```

Isto kao i kod JavaScript primjera, PHP primjer je vrlo jednostavan jer stoji kao nadopuna funkcionalnosti osnovne HTML stranice. Kao što su kod JavaScritpa postojale script oznake za prepoznavanje koda, tako PHP ima <?php za otvaranje i ?> za zatvaranje blokova koda pisanih u PHP-u kako bi se tijekom izvršavanja znalo da je ovaj dio koda potrebno kompajlirati kao PHP kod. Zatim slijedi jednostavna naredba echo za ispis teksta na ekran. Ovoga puta se tekst ne unosi kao parametar unutar zagrade, već se samo piše nakon naredbe unutar navodnika.

Generalno gledajući možemo grupirati primjere po sličnosti u tri razine. C i C++ su vrlo slični po svojoj sintaksi u pisanju „Hello, World!“ programa. Oba jezika prvo uključuju biblioteku, zatim definiraju funkciju unutar koje se poziva slična naredba za ispis samoga teksta, te za kraj se prosljeđuje status 0 za kraj programa. U drugu grupu sličnosti svrstao bih C# i Javu koji od programera traže da definira klasu, a zatim unutar klase se definira main funkcija koja poziva liniju za ispis teksta na ekran. Obije naredbe su slične sintakse kao i ostatak koda. U treću grupu svrstao bih JavaScript i PHP jer oba programska jezika služe za nadopunu funkcionalnosti, te su sintaktički vrlo slični i puno jednostavniji od ostalih.

6.3. Usporedba naredbi kontrole toka

Kada programer razvija programski kod, služi se sa dijelovima koda pomoću kojih gradi svoju aplikaciju. Bez obzira kakvu aplikaciju programer razvija ona će sigurno sadržavati naredbe za kontroliranje toka programa. Pomoću njih određujemo da li će se određeni dio programa izvršiti i koliko puta će se izvršiti. U tu grupu svrstavamo: if-else, for, while, switch, break i continue te goto naredbu.

6.3.1. If else

Najjednostavniji primjer naredbe za upravljanje toka programa je if else. Tom naredbom se služimo kako bi provjerili da li je određeni uvjet zadovoljen. Ukoliko je zadovoljen izvršava se dio koda unutar zagrada, ukoliko uvjet nije zadovoljen možemo ili nastaviti dalje sa normalnim izvršavanjem programa, ili možemo postaviti još nekoliko uvjeta za provjeru.

Sintaksa if else naredbe u C, C++, C# i Java programskom jeziku izgleda ovako:

```
int uvjet = 10;
if (uvjet > 5){

} else if (uvjet = 5) {

else {

}
```

Sintaksa if else naredbe u JavaScript programskom jeziku razlikuje se samo u deklaraciji varijable:

```
var uvjet = 10;
if (uvjet > 5){

} else if (uvjet = 5) {

else {

}
```

Sintaksa if else naredbe u PHP programskom jeziku piše elseif spojeno:

```
$uvjet = 10;
if ($uvjet > 5){

} elseif ($uvjet = 5) {

else {

}
```

Ovisno o tome što želimo napraviti, možemo birati želimo li koristiti samo if, if else u kombinaciji ili if else if nekoliko puta. Sama sintaksa ove provjere ne ovisi o programskome jeziku jer je potpuno identična i u C, C++, C#, Java i JavaScript programskim jezicima. Jedina zanemariva razlika se može primijetiti u PHP programskome jeziku gdje se elseif naredba piše spojeno. Isto vrijedi i za sam expression koji se piše unutar zagrada kao uvjet, svih šest programskih jezika koje obrađujem unutar ovoga rada imaju ista pravila za pisanje uvjeta provjere, uz naravno manje sintaktičke razlike. Poput načina na koji se definiraju varijable unutar uvjeta i slično.

6.3.2. For

U programiranju postoje tri tipa petlji: for petlja, while petlja i do while petlja. Prva petlja koju ću obraditi je for petlja, ona služi za ponavljanje dijela koda sve dok se zadani uvjet ne zadovolji.

Sintaksa for petlje u C programskom jeziku izgleda ovako:

```
int i;
for (i = 0; i < 10; i++)
{

}
```

Sintaksa for petlje u C++, C# i Java programskom jeziku izgleda ovako:

```
for (int i = 0; i < 10; i++)
{

}
```

Sintaksa for petlje u JavaScript programskom jeziku izgleda ovako:

```
var i;
for (i = 0; i < 10; i++)
{

}
```

Sintaksa for petlje u PHP programskom jeziku izgleda ovako:

```
for ($i = 0; $i < 10; $i++)
{

}
```

Sama sintaksa je opet identična u svih šest programskih jezika. Osim razlike u C programskome jeziku gdje se varijabla mora deklarirati izvan for petlje, te klasične razlike u načinu deklariranja varijabli JavaScript i PHP jezika. Kao što vidimo iz primjera imamo ključnu riječ for kojoj u zagradi prosljeđujemo tri parametra. Prvi parametar služi za inicijalizaciju varijable koju ćemo koristiti. U primjeru ispod je prikazana inicijalizacija integer varijable i na broj 1. Zatim drugi parametar odvojen sa točkom zarez predstavlja uvjet naše petlje. Drugim riječima naša petlja će se ponavljati onoliko puta sve dok uvjet nije zadovoljen. U primjeru uvjet je postavljen da je varijabla manja od broja 10. Kako su u programiranju beskonačne petlje moguće, što znači da je moguće napraviti pogrešku da se dio koda beskonačno puta ponavlja i samim time ruši programski kod. Potrebno je proslijediti i treći parametar koji služi kao iterator.

6.3.3. While & Do While

U programiranju postoje tri tipa petlji: for petlja, while petlja i do while petlja. Prva petlja koju ću obraditi je for petlja, ona služi za ponavljanje dijela koda sve dok se zadani uvjet ne zadovolji.

Primjer while petlje pisane u C, C++, C# i Java programskom jeziku:

```
int i = 0;
while (i <= 10)
{
     i++;
}
```

Primjer while petlje pisane u JavaScript programskom jeziku:

```
var i = 0;
while (i <= 10)
{
     i++;
}
```

Primjer while petlje pisane u PHP programskom jeziku:

```
$i = 0;
while ($i <= 10)
{
     $i++;
}
```

Sintaktički je vrlo jednostavan i opet vrlo sličan u svih šest programskih jezika. Ključna riječ while praćena sa uvjetom petlje u zagradi. Bitno je razumjeti da se u ovome slučaju prvo provjerava da li je uvjet zadovoljen, ako je zadovoljen kod unutar petlje se izvršava. Dok postoji i do while petlja, koja prvo izvršava kod a onda provjerava uvjet da li je zadovoljen, ako je onda se kod ponovno izvršava.

Primjer do while petlje pisan u C, C++, C# i Java programskom jeziku:

```
int i = 0;
do
{
     i++
} while (i < 10)
```

Primjer do while petlje pisan u JavaScript programskom jeziku:

```
var i = 0;
do
{
     i++
} while (i < 10)
```

Primjer do while petlje pisan u PHPprogramskom jeziku:

```
$i = 0;
do
{
     $i++
} while ($i < 10)
```

I do while verzija ove petlje je jednaka sintaktički u svim navedenim programskim jezicima. Kao što primjećujemo za sada sve glavne naredbe za upravljanje toka su gotovo identične u svih 6 jezika. To nam govori o samoj sličnosti tih jezika iako su po namjeni različiti.

6.3.4. Switch case

Pomoću switch naredbe programer stvara kod putem kojega na temelju uvjeta bira željeni ishod programa. Isto se može logički postići i sa if else petljama, ali switch case je puno lakša opcija kako za programiranje tako i za kasnije čitanje i razumijevanje koda.

Primjer sintakse switch naredbe pisane u C, C++, C# i Java programskom jeziku:

```
int uvjet = 2;
switch (uvjet)
{
     case 1:

             break;
     case 2:
             //izvršava se ovaj case kod
             break;
     default:
}
```

Primjer sintakse switch naredbe pisane u JavaScript programskom jeziku:

```
var uvjet = 2;
switch (uvjet)
{
     case 1:

             break;
     case 2:
             //izvršava se ovaj case kod
             break;
     default:
}
```

Primjer sintakse switch naredbe pisane u PHP programskom jeziku:

```
$uvjet = 2;
switch ($uvjet)
{
     case 1:

             break;
     case 2:
             //izvršava se ovaj case kod
             break;
     default:
}
```

Switch naredba spada u još jednu naredbu koja je identična u svih šest programa obrađenih u ovome radu. Logika same petlje je isto tako vrlo jednostavna za razumijevanje. Unutar switch ključne riječi prosljeđuje se uvjet kao argument unutar zagrada. Ovisno o izlaznoj vrijednosti toga uvjeta, kod će izvršiti određeni case blok. Ako niti jedan case nije zadovoljen, izvršava se default blok koda.

6.4. Uspoređivanje objektno orijentiranih principa rada

Sada kada smo prošli usporedbe osnovnih koncepata programiranja u svih šest jezika, vrijeme je da se dotaknemo i teme objektno orijentiranog programiranja. Kao što znamo C programski jezik ne podržava objektno orijentirane koncepte programiranja pa njega uopće nećemo spominjati u ovome poglavlju. Sa druge strane JavaScript i PHP iako podržavaju koncepte objektno orijentiranog programiranja, rijetko se koriste za takve svrhe. Puno češće se koriste samo za nadopunjavanje osnovnih funkcionalnosti web stranica. No dotaknuti ćemo se i OOP principa u tim jezicima.

6.4.1. Definiranje klasa i objekata

Glavni koncept objektno orijentiranog programiranja je rad sa klasama i objektima. Tako ćemo i mi u ovome radu započeti sa usporedbom načina kreiranja klasa i objekata unutar navedenih programskih jezika.

Unutar C++ programskog jezika klasa se kreira pomoću ključne riječi class, zatim je potrebno definirati ime same klase te unutar zagrada unosimo varijable i metode te klase. Važno je i razumjeti parametar dostupnosti varijable, tako za varijable koje želimo da budu javno dostupne unutar cijelog programa definiramo pod ključnom riječi public. Slično vrijedi i za varijable koje želimo da budu privatne ili zaštićene. Nakon što je klasa definirana, potrebno je i instancirati objekt te klase. Nakon instance samog objekta, tom objektu možemo dodijeliti vrijednosti unutar njegovih varijabli.

U primjeru navedenom ispod možemo vidjeti definiranje klase knjiga sa tri javno dostupne varijable. Zatim unutar main metode se instancira objekt klase pod imenom knjiga1 kojoj se dodjeljuju vrijednosti za sve tri varijable naslova, autora i broja stranica. Dalje je moguće kreirati beskonačno mnogo objekata te klase.

Primjer definiranja klase i instanciranja objekta u C++ programskom jeziku:

```
class Knjiga {
     public:
             string naslov;
             string autor;
             int brojStranica;
};
```

```
int main() {
     Knjiga knjiga1;
     knjiga1.naslov = „Petar pan“
     knjiga1.autor = „James Matthew Barrie“
     knjiga1.brojStranica = 53;
     return 0;
}
```

Ako prijeđemo na programiranje unutar C# programskoga jezika, proces definiranje klasa i objekata je vrlo sličan. Za početak kao što smo već vidjeli u primjeru ispisa teksta na ekran, C# nas odmah tjera da programiramo sa klasama. Tako pri samom kreiranju novoga programa dobivamo kreiranu klasu unutar koje se nalazi main metoda. Za razliku od C++ programskoga jezika gdje definirano imamo main metodu zasebno, iznad koje možemo definirati klase. Isto tako razliku primjećujemo u sintaksi instanciranja objekta. Sama sintaksa naredbe za kreiranje objekta tipa Knjiga je drugačija od C++ jezika. Te se na drugačiji način definiraju parametri vidljivosti varijabli, u ovome slučaju public se piše ispred svake linije posebno.

Primjer definiranja klase i instanciranja objekta u C# programskom jeziku:

```
class Knjiga
{
     public string naslov;
     public string autor;
     public int brojStranica;

     static void Main(string[] args)
     {
             Knjiga knjiga1 = new Knjiga();
             knjiga1.naslov = „Petar pan“
             knjiga1.autor = „James Matthew Barrie“
             knjiga1.brojStranica = 53;
     }
}
```

Rad sa klasama i objektima unutar programskog jezika Java dosta sliči C# programskome jeziku. Razlike gotovo i nema u samoj osnovnoj definiciji klase, a niti u instanciranju objekta ne možemo uočiti razliku u sintaksi. To nam ponovno potvrđuje prijašnju tezu kako su ta dva programska jezika jako slični, te su oboje dizajnirani da budu jednostavni za učenje i pristupačni za razumijevanje.

Primjer definiranja klase i instanciranja objekta u Java programskom jeziku:

```
public class Knjiga
{
     public String naslov;
     public String autor;
     public int brojStranica;

     public static void Main(String[] args)
     {
             Knjiga knjiga1 = new Knjiga();
             knjiga1.naslov = „Petar pan“
             knjiga1.autor = „James Matthew Barrie“
             knjiga1.brojStranica = 53;
     }
}
```

Već sam spominjao da JavaScript nije često korišten za principe objektno orijentiranog programiranja, zbog same jednostavnosti primjene JavaScripta kao nadopune funkcionalnosti web stranica nema pre velike potrebe za OOP. Ali ako programer želi koristiti principe objektno orijentiranog programiranja oni su mu na raspolaganju u dvije varijante. Prvi jednostavan način uzima definiranje obične funkcije koja se ponaša kao klasa. Unutar funkcije definiramo varijable sa kojima želimo definirati klasu, zatim u samom pozivu te funkcije kao kod konstruktora prosljeđujemo parametre tim varijablama.

Primjer objektno orijentiranih principa putem funkcija unutar JavaScript jezika:

```
function Knjiga(naslov, autor, brojStranica) {
     this.naslov = naslov;
     this.autor = autor;
     this.brojStranica = brojStranica;
}

const knjiga1 = new Knjiga('Petar Pan', ' James Matthew Barrie', 53);
```

Drugi način kreiranja objekta je korištenjem Object() konstruktora. Pomoću Object() konstruktora kreiramo novi prazni objekt kojemu onda definiramo varijable i vrijednosti za željene varijable.

Primjer objektno orijentiranih principa putem Object() konstruktora unutar JavaScript jezika:

```
let knjiga1 = new Object();
knjiga1.naslov = 'Petar pan';
knjiga1.autor = 'James Matthew Barrie';
knjiga1.brojStranica = 53;
```

Za razliku od JavaScript načina rada sa objektno orijentiranim principima, PHP je po tome pitanju puno sličniji klasičnim jezicima poput C# i Jave. Unutar PHP programskoga jezika postoji ključna riječ class pomoću koje definiramo našu željenu klasu, a nakon same klase i instanciranje objekata je vrlo jednostavno i slično principima na koje smo naviknuti. Razlika je u radu sa varijablama klase gdje je potrebno kreirati get i set funkcije kako bi se moglo pristupiti (čitati i pisati) u varijable objekta.

Primjer objektno orijentiranih principa unutar PHP programskog jezika:

```
<?php
     class Knjiga {
             var $naslov;
             var $autor;
             var $brojStranica;

             function setNaslov($naslov){
                     $this->naslov = $naslov;
             }
             function setAutor($autor){
                     $this-> autor = $ autor;
             }
             function setBrojStranica($brojStranica){
                     $this-> brojStranica = $ brojStranica;
             }
     }
     $knjiga1 = new Knjiga;
     $knjiga1->setNaslov(„Petar Pan“);
     $knjiga1->setAutor(„James Matthew Barrie“);
     $knjiga1->setBrojStranica(53);
?>
```

6.4.2. Nasljeđivanje

Osim samog deklariranja klasa vrlo važan princip objektno orijentiranog programiranja je nasljeđivanje. Nasljeđivanje je koncept objektno orijentiranog programiranja gdje klasa nasljeđuje metode druge klase ili sučelja. Sintaksa nasljeđivanja izgleda ovako:

C++ i C# `class A: B, ISucelje { }`

Java i PHP `class A extends B implements Sucelje { }`

JavaScript `A.prototype.B`

Iz ovoga primjera vidimo da C++ i C# imaju istu sintaksu za nasljeđivanje dok Java i PHP logički rade istu stvar, no umjesto korištenja zareza koriste extends ključnu riječ za nasljeđivanje klasa. Dok za nasljeđivanje sučelja koriste ključnu riječ implements. JavaScript je sa svoje strane skroz drugačiji te ne koristi istu logiku nasljeđivanja već koristi prototip za modificiranje objekata.

7. Zaključak

U ovome radu usporedili smo programske jezike C, C++, C#, Java, JavaScript i PHP u nekoliko bitnih kategorija. Prošli smo samu povijest i teoriju nastanka tih programskih jezika, govorili smo o samoj svrsi kreiranja i programiranja u tim programskim jezicima. Proučili smo i najkorištenija razvojna okruženja koja se koriste za programiranje u tim jezicima. Isto tako smo i komentirali njihove brzine izvođenja. Dao sam i subjektivan stav o lakoći učenja tih jezika potkrepljen sa stavom drugih programera. Vidjeli smo da svaki od tih jezika je imao svoju svrhu zbog čega je kreiran, od unaprjeđivanja koncepata programiranja do specificiranih zadaća poput proširivanja funkcionalnosti web stranica svaki od tih jezika je jedinstven.

No iako su jedinstveni, vrlo brzo smo zaključili da su svi ti jezici građeni na vrlo sličnim konceptima i sintaksi. Shvaćanjem osnovnih koncepata programiranja u jednom programskom jeziku, vrlo brzo se može prilagoditi programiranju u drugom programskom jeziku. To smo mogli vidjeti na praktičnom dijelu usporedbe sintaksi. Poput primjera ispisa teksta na ekran gdje se na primjerima koda moglo vidjeti koliko su te sintakse slične. Kasnije smo isto mogli vidjeti na raznim naredbama za kontroliranje toka izvršavanja programa gdje razlike gotovo da nisu ni postojale. Za kraj prošli smo i principe objektno orijentiranog programiranja gdje su razlike postale očitije, pogotovo između grupe C, C++, C# i Jave naspram grupe JavaScript i PHP. No bez obzira i na te manje razlike, njihov logički koncept u pozadini sintakse je ostao isti.

Kada bi morali odrediti koji od ovih jezika su najsličniji, odredio bih 4 grupe jezika. C i C++ bih svrstao zasebno, jer iako je C++ nadogradnja samog C jezika. Razlika u sintaksi je dovoljna da je potrebno utrošiti malo više vremena na prilagodbu. Na primjer, C nas tjera da vezanu listu radimo putem pokazivača dok C++ ima biblioteku za deklariranje i rad sa vezanim listama. Sam princip poput jednostavnog ispisa na ekran je dovoljno različit između ta dva jezika da se početnik programer izgubi u samoj sintaksi.

Sa druge strane u jednu grupu programskih jezika slične sintakse svrstao bih C# i Javu. Ta dva programska jezika funkcioniraju na vrlo sličnim principima, njihova sintaksa je gotovo identična. Velike razlike se primjećuju jedino u kompleksnijim principima programiranja koji zahtijevaju veće znanje, dok za same početnike učenje sintakse u jednom od tih dvaju jezika ne predstavlja veliki problem promjene na drugi programski jezik.

U zadnju grupu programskih jezika koji su najsličniji svrstao bih JavaScript i PHP. Osim što su slični po svojoj namjeni, imaju nove i različite principe od prijašnja 4 programska jezika. Poput načina dinamičkog deklariranja tipa varijabli. Iako je PHP puno sličniji jezicima poput C# i Jave, i dalje je najsličniji JavaScriptu zbog svoje namjene.

Globalno ovaj rad daje dobar temelj za razumijevanje svrha, sličnosti i razlika ovih programskih jezika. Te kao takav može služiti kao dobar uvod u programiranje i odabir željenog programskog jezika. Osim što pruža uvod i usporedbu ovih programskih jezika, ovaj rad pruža grupaciju znanja o ovim programskim jezicima u kojima sam se osobno učio programirati tijekom fakultetskog obrazovanja. Samim time dobivam generalnu sliku o globalnoj cjelini popularnih programskih jezika današnjice, a time dobivam temelje za rad i usavršavanje znanja u bilo kojem od ovih šest programskih jezika.

Popis literature

[1] Towards data science, „Visualize Programming Language Popularity using tiobeindexpy“, [Na internetu]. Dostupno na: https://towardsdatascience.com/visualize-programming-language-popularity-using-tiobeindexpy-f82c5a96400d. [Pristupljeno: 11-Velj-2020].

[2] Geeks for geeks, „Differences between Procedural and Object Oriented Programming“, [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/. [Pristupljeno: 12-Velj-2020].

[3] How Stuff Works, „The Basics of C Programming“, [Na internetu]. Dostupno na: https://computer.howstuffworks.com/c1.htm. [Pristupljeno: 13-Velj-2020].

[4] Tutorials point, „C Language - Overview“, [Na internetu]. Dostupno na: https://www.tutorialspoint.com/cprogramming/c_overview.htm. [Pristupljeno: 13-Velj-2020].

[5] Cplusplus, „History of C++“, [Na internetu]. Dostupno na: http://www.cplusplus.com/info/history/. [Pristupljeno: 14-Velj-2020].

[6] Hackr.io, „Uses and Applications of C++ Language“, [Na internetu]. Dostupno na: https://hackr.io/blog/features-uses-applications-of-c-plus-plus-language. [Pristupljeno: 18-Velj-2020].

[7] C-sharpcorner, „History of C# Programming Language“, [Na internetu]. Dostupno na: https://www.c-sharpcorner.com/blogs/history-of-c-sharp-programming-language. [Pristupljeno: 18-Velj-2020].

[8] ResearchGate, „Overview of CLI“, [Na internetu]. Dostupno na: https://www.researchgate.net/figure/Overview-of-the-Common-Language-Infrastructure-Source_fig4_305194589 [Pristupljeno: 18-Velj-2020].

[9] Pluralsight, „Everything you need to know about C#“, [Na internetu]. Dostupno na: https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c- [Pristupljeno: 18-Velj-2020].

[10] JavaTPoint, „History of Java“, [Na internetu]. Dostupno na: https://www.javatpoint.com/history-of-java [Pristupljeno: 21-Velj-2020].

[11] JavaTPoint, „Features of Java“, [Na internetu]. Dostupno na: https://www.javatpoint.com/features-of-java [Pristupljeno: 21-Velj-2020].

[12] Medium, „A brief history of JavaScript“, [Na internetu]. Dostupno na: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17 [Pristupljeno: 21-Velj-2020].

[13] Springboard, „The History of JavaScript“, [Na internetu]. Dostupno na: https://www.springboard.com/blog/history-of-javascript/ [Pristupljeno: 21-Velj-2020].

[14] Guru99, „What is JavaScript“, [Na internetu]. Dostupno na: https://www.guru99.com/introduction-to-javascript.html [Pristupljeno: 21-Velj-2020].

[15] Php, „History of PHP“, [Na internetu]. Dostupno na: https://www.php.net/manual/en/history.php.php [Pristupljeno: 24-Velj-2020].

[16] Guru99, „What is PHP? Write your first PHP Program“, [Na internetu]. Dostupno na: https://www.guru99.com/what-is-php-first-php-program.html [Pristupljeno: 24-Velj-2020].

[17] Sourceforge.net, „Dev-C++“, [Na internetu]. Dostupno na: https://sourceforge.net/projects/orwelldevcpp/ [Pristupljeno: 25-Velj-2020].

[18] Wikipedia, „Dev-C++“, [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/Dev-C%2B%2B [Pristupljeno: 26-Velj-2020].

[19] Geeks for geeks, „Introduction to Visual Studio“, [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/introduction-to-visual-studio/ [Pristupljeno: 26-Velj-2020].

[20] Eclipse foundation, „Eclipse Home“, [Na internetu]. Dostupno na: https://www.eclipse.org/ide/ [Pristupljeno: 26-Velj-2020].

[21] Brackets, „Brackets“, [Na internetu]. Dostupno na: http://brackets.io/ [Pristupljeno: 26-Velj-2020].

[22] Github, „Niklas Heer: Speed Comparison“, [Na internetu]. Dostupno na: https://github.com/niklas-heer/speed-comparison [Pristupljeno: 28-Velj-2020].

[23] TechRepublic, „The most loved and most disliked programming languages revealed in Stack Overflow survey“, [Na internetu]. Dostupno na: https://www.techrepublic.com/article/the-most-loved-and-most-disliked-programming-languages-revealed-in-stack-overflow-survey/ [Pristupljeno: 17-Trav-2020].

[24] Lovreničić A., Konecki M. (2017), „Programiranje u 14 lekcija“, Sveučilište u Zagrebu Fakultet organizacije i informatike, 224-225