

# SWIFT - programski jezik budućnosti

Seminarski rad u okviru kursa  
Metodologija stručnog i naučnog rada  
Matematički fakultet

Anđelković Dragica, Mandić Igor, Nikolić Igor, Pejović Petar  
andjelkovic.dragica96@gmail.com, igormandic996@gmail.com,  
igor.nikolic032@hotmail.com, petar.pejovic8@gmail.com

29. april 2019

## Sažetak

Swift je odličan jezik za pisanje softvera, bilo da je to za mobilne telefone, desktop računare, servere ili bilo šta što pokreće kôd. To je bezbedan, brz i dinamičan programski jezik koji kombinuje najbolje u jednom savremenom jeziku. Swift spaja najbolja znanja iz široke Apple inženjering kulture i različite doprinose iz zajednice otvorenog kôda (eng. *open source*). Kompajler je optimizovan za performanse, a jezik je optimizovan za razvoj, bez kompromisa na bilo kom frontu.

## Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Nastanak i istorijski razvoj</b>	<b>2</b>
2.1	Mesto u razvojnom stablu i uticaji drugih programskih jezika	3
<b>3</b>	<b>Osnovna namena, svrha i mogućnosti</b>	<b>4</b>
<b>4</b>	<b>Osnovne osobine programskog jezika</b>	<b>5</b>
4.1	Podržane paradigme	6
<b>5</b>	<b>Okruženja i njihove karakteristike</b>	<b>6</b>
5.1	Xcode	6
5.2	Playground	6
5.3	Cocoa Touch	6
5.4	Sublime Text i Atom	6
<b>6</b>	<b>Instalacija i uputstvo za pokretanje</b>	<b>7</b>
6.1	Swift na Windows-u	7
6.2	Swift na Linux-u	8
<b>7</b>	<b>Primer jednostavnog koda i njegovo objašnjenje</b>	<b>9</b>
<b>8</b>	<b>Specifičnosti</b>	<b>11</b>
<b>9</b>	<b>Zaključak</b>	<b>11</b>
	<b>Literatura</b>	<b>11</b>

# 1 Uvod

Swift je novi programski jezik opšte namene razvijen od strane kompanije Apple za iOS, macOS, watchOS, tvOS, Linux i z/OS. Dizajniran je da radi u Apple radnim okruženjima Cocoa i Cocoa Touch. Podržava imperativni, objektno orijentisani i funkcionalni način programiranja. Napravljen je upotrebom LLVM programskog prevodioca otvorenog kôda i uključen je u Xcode, počev od verzije 6 [?]. Swift koristi izvršno okruženje programskog jezika

Objective-C, što omogućava izvršavanje C, C++, Objective-C i Swift kôda u okviru jednog programa [?]. Namera kompanije Apple je bila da Swift podrži mnoge ključne koncepte povezane sa programskim jezikom Objective-C.

U ovom seminarskom radu čitalac se upoznaje sa osnovnim osobinama i funkcionalnostima programskog jezika Swift. Drugo poglavlje biće posvećeno istoriji nastanka programskog jezika Swift, kao i njegovom razvoju od prve verzije pa sve do danas. U trećem poglavlju biće opisane glavne mogućnosti i namena, dok je u četvrtom poglavlju dat pregled osnovnih osobina. Peto poglavlje će biti posvećeno razvojnim okruženjima, biće reči o osnovnim karakteristikama sledećih razvojnih okruženja: Xcode, Playground, SublimeText i Atom. U šestom poglavlju biće opisan način instalacije i pokretanja Swift-a na Windows i Linux operativnim sistemima. Primeri i kratka objašnjenja kôda biće data u sedmom poglavlju. Tema poslednjeg poglavlja će biti specifičnosti ovog programskog jezika.

## 2 Nastanak i istorijski razvoj

Razvoj programskog jezika Swift započeo je 2010. godine Chris Lattner, koji je implementirao veći deo osnovne strukture jezika, za čije je postojanje znala samo nekolicina ljudi. Tek su krajem 2011. godine i drugi programeri počeli da sarađuju na projektu Swift, a u julu 2013. godine on je postao glavni fokus grupe Apple Developer Tools [?].

Swift je predstavljen na međunarodnoj konferenciji programera (eng. *Worldwide Developers Conference - WWDC*) 2014. godine, uz integrisano razvojno okruženje Xcode 6 i OS 8 [?]. Apple je zvanično objavio Swift u decembru 2015. godine, kao projekat otvorenog kôda i pokrenuo je veb sajt <http://swift.org>, koji je posvećen zajednici Swift. Swift skladište nalazi se na GitHub stranici kompanije Apple (<http://github.com/apple>). Swift razvojno skladište (<https://github.com/apple/swift-evolution>) prati napredak Swift-a, dokumentujući predložene promene. U razvojnom skladištu može se pronaći lista predloženih promena koje su prihvaćene i onih koje su odbijene. Swift 3 sadrži nekoliko poboljšanja koje je preporučila zajednica programera. U tabeli 1 se nalaze sve do sada izbačene verzije programskog jezika Swift, u hronološkom redosledu.

Prvu verziju karakteriše REPL alat koji omogućava izvršavanje manjih fragmenata Swift koda i njegovo testiranje sa komandne linije. Ova verzija je donela poboljšanja performansi kompajlera i smanjila vreme potrebno za kompajliranje Swift programa. Prilikom pokretanja projekta, kompajliraju se samo fajlovi kod kojih je detektovana izmena, što je posebno značajno kod većih projekata [?].

U drugoj verziji, kao deo novog projekta, predstavljen je Swift paket menadžer (eng. *packet manager*) za upravljanje Swift bibliotekama. Kao priprema za naredne verzije dodata je provera verzije izvršnog okruženja (eng. *framework*).

Tabela 1: Istorijski razvoj programskog jezika Swift

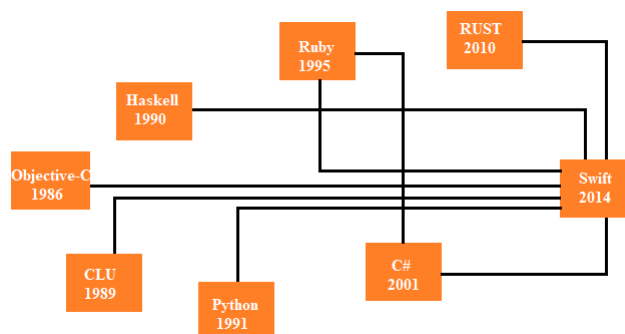
Datum	Verzija
2014-09-09	Swift 1.0
2014-10-22	Swift 1.1
2015-04-08	Swift 1.2
2015-09-21	Swift 2.0
2016-09-13	Swift 3.0
2017-09-19	Swift 4.0
2018-03-29	Swift 4.1
2018-09-17	Swift 4.2
2019-02-28	Swift 4.3
2019-03-25	Swift 5.0

Treća verzija sadrži osnovne promene u samom jeziku i biblioteci Swift standarda, zbog toga nije kompatibilna sa prethodnim verzijama jezika. Jedan od osnovnih ciljeva bio je da bude kompatibilan na više platformi, to znači da će kôd koji se napše za MAC OS funkcionisati i na Linuxu. Tranzicija od druge verzije je bila veoma velika i teška programerima za ispravljanje, projekti su prijavljivali gomilu grešaka tako da su mnogi projekti počinjani ispočetka.

Nakon treće verzije, jezgro programskog jezika Swift nije se drastično menjalo, stoga su četvrta i peta verzija kompatibilne sa trećom. U narednim verzijama radilo se na poboljšanjima implementacije pojedinih koncepata jezika. Poslednja verzija koja je izbačena je verzija 5.0.

## 2.1 Mesto u razvojnem stablu i uticaji drugih programskih jezika

Na razvoj Swift-a uticali su mnogi programski jezici, od kojih su najznačajniji: Objective-C, Rust, Haskell, Ruby, Python, C#, CLU [?]. Mesto programskog jezika Swift u razvojnem stablu je predstavljeno na slici 1.



Slika 1: Razvojno stablo

Preuzeti su određeni delovi iz različitih programskih jezika i poboljšani. Pregled preuzetih koncepata se nalaze u tabeli 2.

Tabela 2: Preuzeti koncepti iz drugih programskih jezika

Programski jezik	Šta je preuzeto
JavaScript	Struktura podataka - rečnik
Scala i Opa	Zaključivanje tipova
Cold Fusion i JSP	Interpolacija Stringa
Python	Opciono naznačavanje kraja naredbe
Java i C#	Protokoli (Interfejsi)
Lisp i Python	Torke (Tuples)
Lisp i JavaScript	Closure funkcije
C# i Objective-C	Označeni i neoznačeni brojevi

### 3 Osnovna namena, svrha i mogućnosti

Pomoću Swift programskog jezika moguće je razviti bilo koji tip iOS i macOS aplikacija. Cilj Swift projekta je da stvori najbolji raspoloživi jezik za upotrebu, od programiranja sistema, preko razvoja mobilnih i desktop aplikacija do cloud usluga. Takođe, ubrzan je proces razvoja proizvoda, poboljšane su performanse i povećana sigurnost aplikacija.

Jedna od glavnih namena Swift programskog jezika je kreiranje mobilnih aplikacija za iPhone i iPad uređaje. Swift je moguće izvršavati i na Linux operativnom sistemu. Članovi zajednice rade na stvaranju Swift aplikacija koje će se izvršavati i na Android platformama.

Osim što je Swift poznat po razvoju aplikacija za Apple platforme, koristi se i u modernim server aplikacijama. Swift je odličan izbor za server aplikacije koje zahtevaju visoke performanse kompajlera, nizak stepen korišćenja memorije i visok nivo bezbednosti.

Swift je sve popularniji programski jezik za razvoj IoT (eng. *Internet of Things*) aplikacija. Kako bi kompanija Apple postala lider u primeni IoT aplikacija, razvijene su biblioteke i razvojni okviri koje rade najveći deo posla, dok se programeri mogu fokusirati na funkcionalnosti IoT aplikacija.

Swift ima svoju standardnu biblioteku. Biblioteka sadrži osnovne funkcionalnosti za pisanje Swift programa, uključujući tipove podataka, strukture podataka, funkcije, metode, protokole, i mnogo drugih stvari.[?] Neke od mogućnosti koje pruža pomoću svojih funkcija su: [?]

- Automatsko utvrđivanje tipova - Swift može automatski da utvrdi tip promenljive ili konstante na osnovu inicijalne vrednosti.
- Generički tipovi - generički tipovi omogućavaju da se piše kod jednom za izvršenje identičnih zadataka za različite tipove objekata dok se zadržava bezbednost tipa.
- Sintaksa zatvorenog izraza - zatvoreni izrazi su samostalni blokovi funkcionalnosti koji mogu da se proslede i upotrebe u kodu.
- Pseudoklase - pseudoklasa definiše promenljivu koja možda nema vrednost.
- Switch iskaz - Switch iskaz je drastično poboljšan funkcijama, kao što su poklapanje šablona i zaštitni uslovi; zahvaljujući njima, izbegnute su automatske greške.
- Višestruki povratni tipovi - funkcije mogu da imaju višestruke povratne tipove upotrebom torki.

- Preklapanje operatora - klase mogu da obezbede sopstvenu implementaciju postojećih operatora.

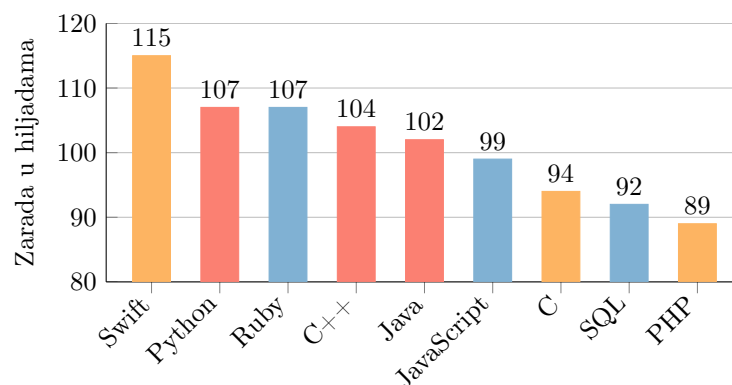
Postoji još jedna funkcija, koja tehnički nije funkcija Swifta, već Xcode-a i kompajlera. To je **Mix and match**. Ona omogućava kreiranje aplikacija koje sadrže Objective-C i Swift fajlove. To omogućava da se sistematski ažuriraju aktuelne Objective-C aplikacije pomoću Swift klasa i upotrebu Objective-C biblioteka/radnih okvira u Swift aplikacijama.

## 4 Osnovne osobine programskog jezika

Swift je objektno orijentisan programski jezik, koji je Apple razvio sa ciljem da se poboljšaju određeni delovi jezika Objective-C, ali da se i iskoriste njegove dobre osobine. Najvažnije osobine programskog jezika Swift, koje ga čine izuzetnim za učenje iOS programiranja su [?]:

- **Objektno orijentisan** - moderan objektno orijentisan jezik.
- **Funkcionalan** - sadrži osobine zbog kojih je pogodan za pisanje funkcionalnih programa.
- **Jasan** - lako se čita i lako piše, ima minimalne sintaksne ukrase i samo nekoliko skrivenih prečica. Njegova sintaksa je jasna, dosledna i očigledna.
- **Bezbedan** - zahteva jake tipove kako bi obezbedio da u svakom trenutku i on i programer znaju na šta se sve tipovi objekata pozivaju.
- **Ekonomičan** - mali jezik koji nudi samo neke osnovne tipove podataka i funkcionalnosti. Preostalo mora da bude dato kodom programera ili bibliotekama (razvojno okruženje Cocoa).
- **Upravlja memorijom** - automatski upravlja memorijom, programer ne treba o tome da brine.
- **Kompatibilnost sa razvojnim okruženjem Cocoa** - Swift je napravljen tako da koristi većinu API interfejsa razvojnog okruženja Cocoa.

Zbog svih ovih osobina, 2016. godine, programski jezik Swift je bio najplaćeniji jezik u Americi. Na sledećem grafikonu (slika 2) prikazano je 10 najplaćenijih programskih jezika:



Slika 2: Najplaćeniji programski jezici u Americi 2016. godine

## 4.1 Podržane paradigme

Programski jezik Swift je objektno orijentisani jezik, ali podržava i funkcionalno i imperativno programiranje. Sadrži sve osnovne koncepte objektno orijentisanog programiranja, i objektno orijentisana paradigma je najzastupljenija u ovom jeziku. Međutim, nekoliko osobina Swift jezika, kao što su funkcije prvog reda, sofisticirani sistem tipizacije, lambda izrazi, korišćenje Karijevih funkcija i parcijalna aplikacija, čine jezik posebno pogodnim za pisanje funkcionalnih programa[?]. Swift, kao funkcionalni jezik, može se u svakodnevnoj praksi koristiti za pisanje kraćih, elegantnijih i sigurnijih programa, koji su lakši za održavanje, nadogradnju i testiranje.

## 5 Okruženja i njihove karakteristike

Programski jezik Swift se može pisati u različitim okruženjima. Najpoznatije okruženje je **Xcode**, a pored njega, koriste se i AppCode, Atom, CLion, SublimeText.

### 5.1 Xcode

Xcode je integrisano razvojno okruženje koje je napravila kompanija Apple. Koristi se za razvoj iOS i macOS aplikacija. U ovom okruženju mogu se pisati kodovi u mnogim programskim jezicima, kao što su C, C++, Objective-C, Java, AppleScript, Python, Ruby i Swift. Xcode sadrži i okvire, za programski jezik Swift su najvažniji **Playground** i **Cocoa Touch**. Postoji 9 verzija ovog okruženja, a od verzije 6, obuhvata i programski jezik Swift.

### 5.2 Playground

Playground je interaktivno radno okruženje koje omogućava da pišemo kôd i odmah vidimo rezultate čim su promene izvršene u kôdu. Kako bismo mogli da koristimo ovaj okvir potrebno je prvo da pokrenemo Xcode i zatim izaberemo opciju Get started with a playground. Sastoji se od nekoliko delova, među kojima su najvažniji:

- Prostor za kodiranje.
- Bočna traka za rezultate.
- Prostor za ispravljanje grešaka.

### 5.3 Cocoa Touch

Cocoa Touch je okruženje prvenstveno napravljeno za razvoj programa koji su namenjeni uređajima koji koriste iOS operativni sistem. Ovo okruženje je napisano u ObjectiveC-u. Omogućava upotrebu hardvera i karakteristika koje nisu implementirani na macOS računarima, već je jedinstvena za iOS asortiman uređaja. Sadrži različite skupove alata za kontrolu grafičkih elemenata. Alati za razvoj aplikacija ovog okruženja uključeni su u iOS SDK.

### 5.4 Sublime Text i Atom

Sublime Text i Atom, iako nisu prvenstveno namenjeni za razvijanje Swift programa, verovatno su jedni od najrasprostranjenijih i najpoželj-

njihijih okruženja za rad. Poseduju dobar korisnički interfejs i sjajne performanse. U osnovni podržavaju mnoge jezike, a nove funkcionalnosti mogu biti dodate korišćenjem dodataka (plugina). Vrlo lako ih možemo prilagoditi za razvoj Swift aplikacija dodatkom podrške za Swift pakete.

## 6 Instalacija i uputstvo za pokretanje

Programski jezik Swift se može koristiti na različitim operativnim sistemima. Ukoliko koristimo MAC operativni sistem, dovoljno je da se preuzme i instalira Xcode razvojno okruženje, jer Xcode uključuje izdanje Swift-a koje podržava Apple. Za Linux i Windows instalacija i pokretanje su složeniji, i biće prikazano u ovom poglavlju.

### 6.1 Swift na Windows-u

Za instalaciju programskog jezika Swift na Windows-u potrebno je prvo da ga preuzmemo sa ovog [linka](#). Nakon toga nam se pojavljuje prozor za instalaciju, gde pratimo dalja uputstva i tako instaliramo Swift i kompajler za ovaj jezik. Nakon završetka instalacije, potreban nam je editor teksta u kojem želimo da kodiramo. Možemo koristiti bilo koje od gore navedenih razvojnih okruženja u kojem nam je udobno da radimo. U ovom primeru koristićemo [Notepad++](#), koji je jednostavan, besplatan i lak za instalaciju.

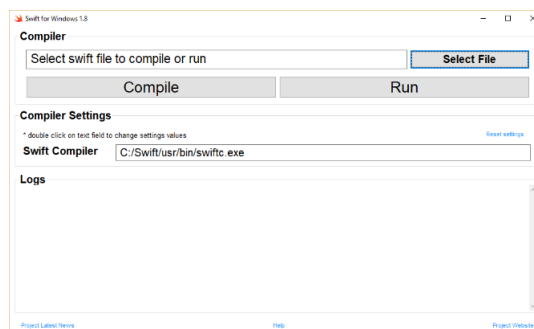
Nakon instalacije okruženja, napisaćemo jednostavan program koji ćemo kasnije pokrenuti pomoću Windows komandne linije. Prvo je potrebno da otvorimo novi Notepad++ fajl, i u njega unosimo komandu za ispis.

```
1000 print ("Hello world!")
```

Listing 1: Primer komande

Da bi sačuvali ovaj kôd, koristićemo File > Save As i izabrati Swift file iz Save As Type menija. Ako u meniju nedostaje tip ovog fajla izabraćemo all files, i dodati .swift fajl ekstenziju nakon što smo dali odgovarajuće ime fajlu.

Sada kada imamo program, želimo da ga kompajliramo i pokrenemo. Za kompajliranje i pokretanje koristimo korisnički interfejs Swift-a koji je prikazan na slici 3.



Slika 3: Korisničko okruženje za Windows

Nakon pritiska na Select File izabraćemo naš program koji smo prethodno napisali i pritisnuti Compile. Nakon što sačekamo da se kompilacija završi dobićemo poruku "Successfully compiled". Jednom kompajliran program možemo pokrenuti neograničen broj puta.

## 6.2 Swift na Linux-u

Kao i u prethodnom primeru biće nam potreban tekst editor gde ćemo napisati jednostavan kôd. Možemo koristiti bilo koji integrisani editor koji Linux poseduje. Na isti način ćemo iskodirati poruku za ispis "Hello World", kao u prethodnom primeru, i taj fajl ćemo sačuvati sa ekstenzijom .swift.

Da bismo koristili Swift na Linux-u moramo ga prvo instalirati. U terminalu kucamo sledeće komande:

```
1000 wget https://swift.org/builds/ubuntu1510/swift-2.2-SNAPSHOT-2015-12-10-a/swift-2.2-SNAPSHOT-2015-12-10-a-ubuntu15.10.tar.gz
```

Listing 2: Instaliranje Swift-a

Nakon preuzimanja, pozicioniraćemo se u folder Downloads i tamo raspakovati arhivu u kojoj se nalazi Swift instalacija.

```
1000 cd ~/Downloads
tar -xvzf swift-2.2-SNAPSHOT*
```

Listing 3: Raspakovanje Swift instalacije

Kada raspakujemo fajl potrebno je podesiti putanju do BIN-a kako bismo mogli da izvršavamo programe.

```
1000 cd ~/Downloads/swift-2.2-SNAPSHOT*
      cd usr/bin
1002 pwd
```

Listing 4: Podešavanje putanje do BIN-a

Kao rezultat komande pwd dobićemo tačnu lokaciju koju ćemo koristiti. Kopiramo i zamenimo je na sledeći način.

```
1000 export PATH=path_to_swift_usr_bin:$PATH
```

Listing 5: Kopiranje putanje

Zatim moramo instalirati jos par biblioteka kako bi omogućili da Swift nesmetano funkcioniše na Linux-u.

```
1000 sudo apt-get install clang libicu-dev
      swift -version
```

Listing 6: Instalacija biblioteka

Na kraju, da bismo kompajlirali i pokrenuli prethodno napisan program potrebno je da ukucamo sledeće komande:

```
1000 swift imeprograma.swift
      ./imeprograma
```

Listing 7: Komanda za kompajliranje



## 7 Primer jednostavnog koda i njegovo objašnjenje

U narednim primerima biće prikazane i objašnjene osnovne funkcionalnosti jezika Swift.

**Primer 7.1** *Ispis teksta se vrši pomoću funkcije `print()`. Tačka-zarez su opcioni na kraju svakog reda.*

```
1000 print("Hello world!")           // Hello World!  
    print("Hello world!");         // Hello World!
```

Listing 8: Ispis teksta

**Primer 7.2** *Stringovi se takodje dodeljuju pomoću operatora dodele. Konkatencija stringova se vrši pomoću specijalnih karaktera `\` (string)' ili jednostavnim navodjenjem u naredbi `'print'`, gde se stringovi razdvajaju zarezima.*

```
1000 var ime = "Swift"  
    var jezik = "programski jezik"  
1002  
    var poruka = " je najbolji "  
1004 var poruka1 = "\ (ime) je najbolji \ (jezik) !"  
1006 print(ime, poruka, jezik, "!")           // Swift je  
    najbolji progmski jezik  
    print(poruka1)                          // Swift je najbolji  
    progmski jezik
```

Listing 9: Stringovi i konkatencija stringova

**Primer 7.3** *Listu stringova koji su razdvojeni određenim separatorom, pravimo pomoću naredbe `print`, gde se prvo navode stringovi koji čine tu listu, a nakon toga separator i terminator. Možemo koristiti još jedan parametar u funkciji `print()`, pod nazivom `toStream`. Pomoću njega preusmeravamo ispis funkcije `print()`. Konkretno u ovom primeru preusmeravamo ispis u promenljivu `ime4`.*

```
1000 var ime1 = "Swift"  
    var ime2 = "Java"  
1002 var ime3 = "Python"  
    var ime4 = ""  
1004  
    print(ime1, ime2, ime3, separator: ", ", terminator: "")  
    // Swift, Java, Python  
1006 print(ime1, ime2, ime3, separator: ", ", terminator: "", to:&  
    ime4)
```

Listing 10: Lista stringova

**Primer 7.4** *U ovom primeru je pokazana funkcionalnost `for` i `while` petlje. Takođe jos jednom i mogućnost konkatencije pomoću operatora `+`. Nakon `for` petlje u konzoli će biti ispisani brojevi od 0 do 10. U `while` petlji će se svakog puta dodavati po jedno slovo `a`, i tako 5 puta.*

```

1000 var x:Int = 0
1001 for x in 0...10 {
1002     print(x)
1003 }
1004
1005 var y:String = ""
1006 while y != "aaaaa" {
1007     print(y)
1008     y = y + "a"
1009 }

```

Listing 11: Petlje

**Primer 7.5** Pozivanje funkcije sa parametrima koja nema povratnu vrednost. Sintaksa za deklaraciju funkcija se sastoji iz ključne reči `func`, naziva i liste parametara. Parametri su oblika `naziv ':' tip promenljive`.

```

1000 var skor:Int = 0
1001 var trenutno_stanje:Float = 0
1002
1003 func dodaj_poene_i_novac(poeni:Int , novac:Float){
1004     skor = skor + poeni
1005     trenutno_stanje = trenutno_stanje + novac
1006 }
1007
1008 dodaj_poene_i_novac(poeni: 30, novac: 1.45)
1009 dodaj_poene_i_novac(poeni: 60, novac: 2.86)
1010
1011 print(skor) // 90
1012 print(trenutno_stanje) // 4.31

```

Listing 12: Funkcije bez povratne vrednosti

**Primer 7.6** Funkcije u programskom jeziku Swift dozvoljavaju da vraćamo jednu ili više vrednosti istovremeno. Sintaksa za povratnu vrednost je `'return(niz vrednosti koje želimo da vratimo)'`.

```

1000 func izracunajMinMaxSuma(a: Int, b: Int) -> (min: Int,
1001     max: Int, suma: Int) {
1002
1003     if a > b {
1004         return (b, a, a + b)
1005     } else {
1006         return (a, b, a + b)
1007     }
1008 }
1009
1010 let statistika = izracunajMinMaxSuma(5, b: 19)
1011 let (min2, max2, suma2) = izracunajMinMaxSuma(5, b: 19)
1012 print(suma2) // 24
1013 print(statistika.sum) // 24
1014 print(statistika.2) // 5

```

Listing 13: Funkcije koje imaju povratnu vrednost

**Primer 7.7** Klase u programskom jeziku Swift se definišu sa `'class'`, nakon čega sledi ime klase. Klase sadrže polja, konstruktor i metode. Poljima klase se pristupa pomoću `'self'`, dok se konstruktor definiše pomoću ključne reči `'init'`,

```

1000 class Osoba {
1002     var ime: String
1002     var godine: Int
1004
1004     init(ime: String, godine: Int) {
1006         self.ime = ime
1006         self.godine = godine
1008     }
1008     func getIme() -> String {
1010         return "Tvoje ime je \(self.ime)"
1010     }
1012 }
1012 var osoba1 = Osoba(ime: "Daca", godine: 22)
1012 print(osoba1.getIme()) // Daca

```

Listing 14: Klasa

## 8 Specifičnosti

Swift je veoma pristupačan novim programerima, to je industrijski kvalitetan programski jezik koji je veoma detaljan i pogodan kao skriptni jezik. Swift se dobro štiti od najzastupljenih programskih grešaka usvajanjem modernih paterna programiranja:

- Promenljive su uvek inicijalizovane pre upotrebe.
- Obrađena je greška za pristupanje nepostojećem elementu niza (eng. *out of bounds*).
- Integer-i su provereni za prekoračenje memorije (eng. *overflow*).
- Opcione promenljive zahtevaju eksplicitno rukovanje.
- Memorijom se upravlja automatski.
- Rukovanje greškama omogućava kontrolisani oporavak od neočekivanih prekida (crash).

Swift kod je kompajliran i optimizovan da izvuče najviše iz modernog hardvera. Ova kombinacija sigurnosti i brzine čini Swift odličnim izborom za sve, od komande "Hello world!", do celog operativnog sistema.

## 9 Zaključak

Swift je mlad jezik, koji iz godine u godinu napreduje i postaje popularniji u iOS programiranju. Očekuje se njegova ekspanzija u narednom periodu, pre svega zbog sve većeg korišćenja iOS aplikacija. Godinama se radilo na Swift-u i on nastavlja da evoluira sa novim funkcionalnostima, a cilj koji ima je veoma ambiciozan. Ovaj seminarski rad ima za cilj da uvede čitaoca u ovaj programski jezik, a takođe, i da ga motiviše da nastavi sa učenjem i usavršavanjem ovog jezika.