Programski jezik SWIFT

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

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

6. april 2019

Sažetak

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

Sadržaj

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 i postojećeg Objective-C koda pisanog za Apple proizvode. Podržava imperativni, objektno-orijentisani i funkcionalni način programiranja. Napravljen je upotrebom LLVM programskog prevodioca otvorenog koda 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 koda 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 koda 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 izbacio Swift u decembru 2015. godine, kao projekat otvorenog koda 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 Swifta, 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, predstvljen 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.

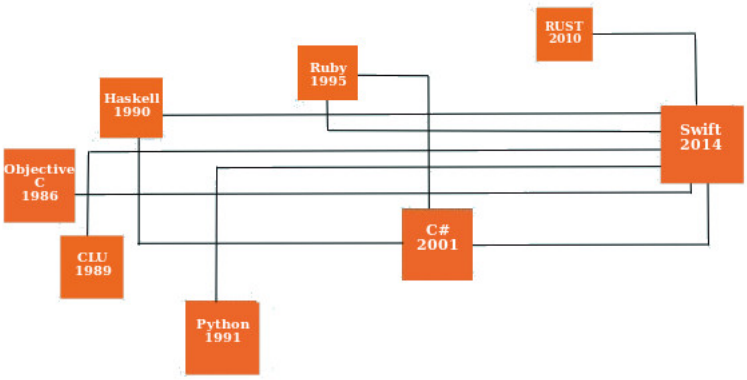
Datum	Verzija
2014-09-09	Swift 1.0
2014-10-22	Swift 1.1
2015-04-08	Swift 1.2
2015-09-21	Swift 2.0
2016-09-13	Swift 3.0
2017-09-19	Swift 4.0
2018-03-29	Swift 4.1
2018-09-17	Swift 4.2
2019-02-28	Swift 4.3
2019-03-25	Swift 5.0

Treća verzija sadrži osnovne promene u samom jeziku i biblioteci Swift standarda, zbog toga nije kompatibilan sa prethodnim verzijama Swift jezika. Jedan od osnovnih ciljeva bio je da on bude kompatibilan na više platformi, to znači da će kod 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 razvojnom stablu i uticaji drugih programskih jezika

Na razvoj Swifta uticali su mnogi programski jezici, od kojih su najznačajniji: Objective-C, Rust, Haskell, Ruby, Python, C#, CLU [**?**]. Mesto programskog jezika Swfit u razvojnom 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	Tuples
Lisp i JavaScript	Closure funkcije
C# i Objective-C	Signed i unsigned int

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 na Linux operativnom sistemu i Raspberry Pi. 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 funcionalnosti IoT aplikacija.

Neka od mogućnosti koje pruža pomoću svojih funkcija [**?**]

- 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.
- Promenljivost kolekcije - Swift nema posebne objekte za promenljive ili nepromenljive kontejnere. Umesto toga, možete da definišete promenljivost definisanjem kontejnera kao konstante ili kao promenljive.
- 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.
- Nabrajanja sa pratećim vrednostima - u Swiftu može da se uradi mnogo više od jednostavnog definisanja grupe srodnih vrednosti pomoću nabrajanja.

Postoji još jedna funkcija, koja tehnički nije funkcija Swifta, već Xcodea 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. Kao što smo već rekli u jezik Swift su uključene i osobine i mnogih drugih jezika. 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 i programer ne treba o tome da brine.
- **Kompatibilnost sa razvojnim okruženjem Cocoa** - porgramski interfejsi aplikacije (API) razvojnog okruženja Cocoa napisani su u jezicima C i Objective-C. Swift je napravljen tako da koristi većinu API interfejsa razvojnog okruženja Cocoa.

4.1 Podržane paradigme

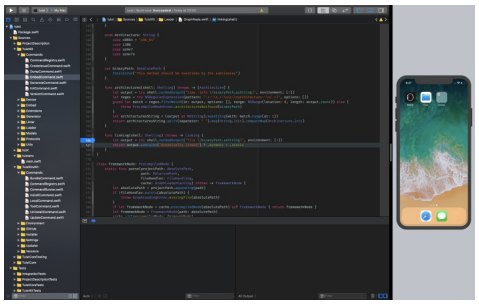
Programski jezik Swift je objektno orijentisani jezik, ali podržava i funkcionalno i imperativno programiranje. Sadrži sve osnovne koncepte objektno orijentisanog programiranja, i objektno orijentisana paradigma je najzastupljenija u ovom jeziku. Međutim, nekoliko osobina Swift jezika, kao što su funkcije 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 (slika 2) 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.



Slika 2: Okruženje Xcode

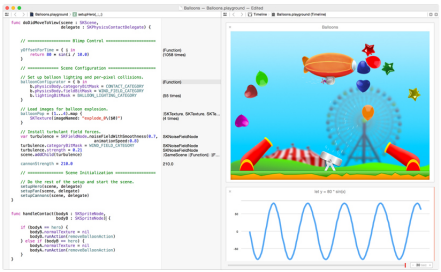
5.2 Playground

Playground (slika 3) je interaktivno radno okruženje koje omogućava da pišemo kod i odmah vidimo rezultate čim su promene izvršene u kodu. 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:

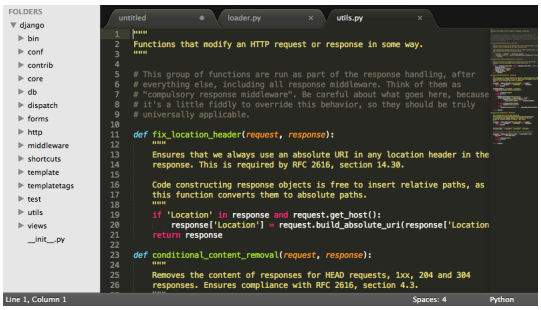
- Prostor za kodiranje.
- Bočna traka za rezultate.
- Prostor za ispravljanje grešaka.

5.3 SublimeText

Sublime Text (slika 4) je verovatno jedan od najrasprostranjenijih okruženja. Poseduje dobar korisnički interfejs i sjajne performanse. On u osnovni podržava mnoge programske jezike a nove funkcionalnosti mogu biti dodate korišćenjem dodataka (plugina), koji je zajednica razvijala pod licencom slobodnog softvera. Vrlo lako ga možemo prilagoditi za razvoj Swift aplikacija dodatkom podrške za Swift pakete.



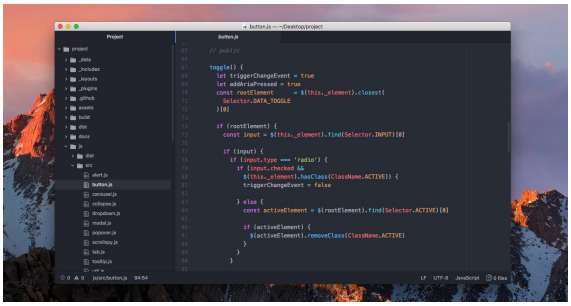
Slika 3: Playground



Slika 4: Sublime

5.4 Atom

Atom (slika 5) je razvijen od strane GitHub-a, visoko modularno okruženje koje omogućava laku instalaciju novih paketa što ga je svrstalo među najpoželjnija okruženja. Dodavanje podrške za Swift programiranje je jednostavno instaliranjem language-swift paketa.



Slika 5: atom

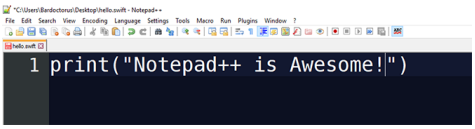
6 Instalacija i uputstvo za pokretanje

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 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++ (slika 6), koji je jednostavan, besplatan i lak za instalaciju.



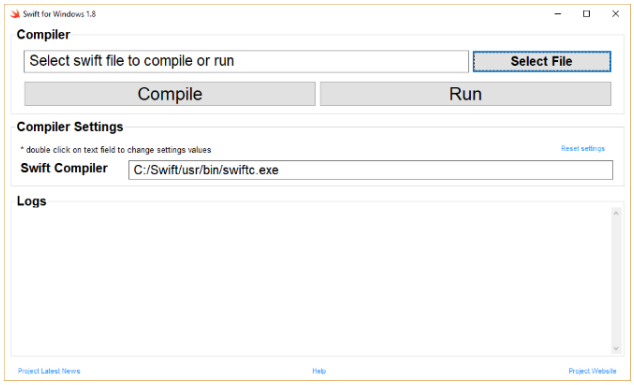
Slika 6: Notpad

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!")
```

Da bi sačuvali ovaj kod, 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 7.



Slika 7: 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 kod. 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
```

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*
```

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
```

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
```

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
```

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

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

7 Primer jednostavnog koda i njegovo objašnjene

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!")
1002 print("Hello world!");
```

Listing 1: Ispis teksta

Primer 7.2 *Nije strogo tipiziran (deklarišemo promenljive pomocu - var), a dodeljujemo vrednost pomoću operatora dodele. Zagrade kod provere uslova su opcione. Preporučujemo da se zagrade ne koriste, osim u slučaju kada imate više uslovnih iskaza u istoj liniji. Velike (vitičaste) zagrade moraju se koristiti kod if-grane i petlji, u suprotnom će doći do greške. I za uslovne iskaze (if i while) i za iskaze dodele (=) razmaci su opcioni.*

```
1000 var x = 5
var y = 3
1002
if x == 5 {
1004   print("x je 5");
}
1006
if (x==3) {
1008   print("x je 5");
}
1010
if (x == 5)          //Syntax error
1012   print("x je 5");
```

Listing 2: Dodeljivanje vrednosti i provera uslova

Primer 7.3 *Stringovi se takođe dodeljuju pomoću operatora dodele. Konkatenacija stringova se vrši pomoću specijalnih karaktera '\ (string)' ili jednostavnim navođenjem u naredbi 'print', gde se stringovi razdvajaju zarezima.*

```
1000
var ime = "Swift"
1002 var jezik = "programski jezik"

1004 var poruka = " je najbolji "
var poruka1 = "\ (ime) je najbolji \ (jezik) !"
1006
print(ime,poruka,jezik,"!")
1008 print(poruka1)
```

Listing 3: Stringovi i konkatenacija stringova

Primer 7.4 *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 jos 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"
1002 var ime2 = "Java"
var ime3 = "Python"
1004 var ime3 = ""

1006 print(ime1,ime2,ime3,separator:", ", terminator:"")
print(ime1,ime2,ime3,separator:", ", terminator:"", to:&
    ime4)
```

Listing 4: Lista stringova

Primer 7.5 *U ovom primeru je pokazana funkcionalnost for i while petlje. Takođe jos jednom i mogućnost konkatenacije 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
1002 for x in 0...10 {
  print(x)
1004 }

1006 var y:String = ""
while y != "aaaaa" {
1008   print(y)
  y = y + "a"
1010 }
```

Listing 5: Petlje

Primer 7.6 *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
1002 var trenutno_stanje:Float = 0

1004 func dodaj_poene_i_novac(poeni:Int, novac:Float){
  skor = skor + poeni
1006   trenutno_stanje = trenutno_stanje + novac
}
1008
dodaj_poene_i_novac(points: 30, novac: 1.45)
1010 dodaj_poene_i_novac(points: 60, novac: 2.86)

1012 print(skor)
print(trenutno_stanje)
```

Listing 6: Funkcije bez povratne vrednosti

Primer 7.7 *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,
    max: Int, suma: Int) {
1002
    if a > b {
1004        return (b, a, a + b)
    } else {
1006        return (a, b, a + b)
    }
1008}

1010let statistika = izracunajMinMaxSuma(5, b: 19)
let (min2, max2, suma2) = izracunajMinMaxSuma(5, b: 19)
1012print(suma2)
print(statistika.sum)
1014print(statistika.2)
```

Listing 7: Funkcije koje imaju povratnu vrednost

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:

- Promenjive su uvek inicijalizovane pre upotrebe.
- Nizovi su provereni za out of bounds greškom.
- Integer-i su provereni za overflow.
- Opcione promenjive 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. Sintaksa i standardna biblioteka su bazirane na konceptu da je očigledan način za pisanje koda takav da se on izvršava najbolje. 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, a ovaj programski jezik je o tome najbolji. 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 jezik, a takodje, i da ga motiviše da nastavi sa učenjem i usavršavanjem ovog jezika.