

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

2. april 2019

Sažetak

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada

Sadržaj

1	Uvod	3
2	Nastanak i istorijski razvoj, uticaji drugih programskih jezika	3
2.1	Swift 1	4
2.2	Swift 2	4
2.3	Swift 3	4
2.4	Swift 4	4
3	Osnovna namena, svrha i mogućnosti	4
4	Osnovne osobine ovog programskog jezika, podržane paradigme i koncepti	5
4.1	Veza sa Objective-C	6
4.2	Osnovne osobine	6
4.3	Paradigma	6
5	Najpoznatija okruženja (framework) za korišćenje ovog jezika i njihove karakteristike	7
6	Instalacija i uputstvo za pokretanje na Linux/Windows operativnim sistemima	7
7	Primer jednostavnog koda i njegovo objašnjenje	8
8	Sve ono što je specifično i važno za sam taj programski jezik	10
9	Zaključak	10
	Literatura	10

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.

2 Nastanak i istorijski razvoj, uticaji drugih programskih jezika

Razvoj programskog jezika Swift je započeo 2010. godine Chirs 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 posato glavni fokus grupe Apple Developer Tools.

Swift je predstavljen na međunarodnoj konferenciji programera (WWDC-Worldwide Developers Conference) 2014. godine, uz integrisano razvojno okruženje Xcode 6 i OS 8. U decembru 2015. godine Apple je zvanično izdao Swift kao projekat otvorenog koda i pokrenuo je veb sajt <http://swift.org>, koji je posvećen zajednici Swift. Swift skladište se nalazi 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 se može pronaći lista predloženih promena koje su prihvaćene i onih koje su odbijene. Swift 3 sadrži nekoliko poboljšanja koje je preporučila zajednica programera. Na razvoj Swifta uticali su mnogi programski jezici, od kojih su najznačajniji: Objective-C, Ruby, Haskell, C#, Python. U tabeli 1 se nalaze sve do sada izbačene verzije programskog jezika Swift, u hronološkom redosledu.

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
...	Swift 5.0

2.1 Swift 1

Prvu verziju karakteriše REPL alat koji omogućava izvršavanje manjih fragmenata Swift koda i njegovo testiranje sa komandne linije. U Swift 1.2 verziji uvedena je nova struktura podataka - skup. 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.

2.2 Swift 2

Glavne funkcionalnosti koje su ugrađene u programski jezik su:

- programiranje orjentisano na protokole,
- model za obradu grešaka,
- odlaganje izvršavanja naredbe pomoću ključne reči defer,
- provera da li je funkcija dostupna na trenutnoj verziji uređaja i platforme koja pokreće našu aplikaciju pomoću ključne reči available.

U drugoj verziji, kao deo novog projekta, predstavljen je Swift paket menadžer za upravljanje Swift bibliotekama. Kao priprema za naredne verzije dodata je provera verzije izvršnog okruženja.

2.3 Swift 3

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 autora Swifta 3 je da on bude kompatibilan na više platformi, tako da kod koji se napiše za jednu platformu bude kompatibilan na svim drugim platformama. To znači da će kod koji se napiše za MAC OS funkcionisati i na Linuxu.

2.4 Swift 4

Četvrta verzija je kompatibilna sa trećom. Nove karakteristike koje je podržala četvrta verzija su:

- parsiranje fajlova u json i xml formatu,
- jednostrano definisani opsezi,
- poboljšanje funkcionalnosti struktura podataka: rečnik i skup,
- kombinovanje klasa i protokola,
- pisanje generičkih podskripti.

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 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.

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 poboljšao 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.
- Nabranjanja sa pratećim vrednostima - u Swiftu može da se uradi mnogo više od jednostavnog definisanja grupe srodnih vrednosti pomoću nabranjanja.

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 ovog programskog jezika, podržane paradigme i koncepti

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. Pored Objective-C, u jezik Swift su uključene i osobine i mnogih drugih jezika, kao što su Python, Java, Ruby, Haskell.

4.1 Veza sa Objective-C

Jezik Swift predstavlja unapređenu, intuitivniju i pregledniju verziju jezika Objective-C. Nedostaci programskog jezika Objective-C u odnosu na Swift su:

- Delimično objektno orijentisan - Objective-C objektno orijentisane osobine je preuzeo od jezika C. Sadrži i objekte i skalarne tipove podataka, sa tim što se objekti nalaze unutar posebnog C tipa podataka (pokazivača).
- Sintaksa može da bude složena i teska za razumevanje.
- Provera tipova često može da se isključi, i često je isključena.
- Koristi ručno upravljanje memorijom.

Glavna osobina jezika Objective-C je kompatibilnost sa razvojnim okruženjem Cocoa. Cocoa API interfejsi su ugrađene komande, sa kojima kod programera mora da sarađuje kako bi se bilo šta dogodilo na nekom iOS uređaju. Cocoa Api interfejs je napisan u jezicima Objective-C i C. Swift je napravljen tako da može koristiti većinu API interfejsa razvojnog okruženja Cocoa, i to mu omogućava pisanje iOS aplikacija.

4.2 Osnovne osobine

Najvažnije osobine programskog jezika Swift 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**

Ove osobine čine Swift izuzetnim jezikom za učenje iOS programiranja.

4.3 Paradigma

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, nadgradnju i testiranje.

5 Najpoznatija okruženja (framework) za korišćenje ovog jezika i njihove karakteristike

Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo

6 Instalacija i uputstvo za pokretanje na Linux/Windows operativnim sistemima

Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo
Igorov deo

7 Primer jednostavnog koda i njegovo objašnjenje

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

```
1000 print("Hello world!")
1002 print("Hello world!");
```

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

```
1000 var x = 5
1002 var y = 3
1004 if x == 5 {
1005     print("x je 5");
1006 }
1008 if (x==3) {
1009     print("x je 5");
1010 }
1012 if (x == 5) //Syntax error
1013     print("x je 5");
```

Nije strogo tipiziran (deklariramo promenljive pomoću `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 ime = "Swift"
1002 var jezik = "programski jezik"
1004 var poruka = " je najbolji "
1005 var poruka1 = "\ (ime) je najbolji \ (jezik) !"
1006
1007 print(ime, poruka, jezik, "!")
1008 print(poruka1)
```

Stringovi se takodje dodeljuju pomoću operatora dodele.

Konkatenacija stringova se vrši pomoću specijalnih karaktera `'\ (string)'` ili jednostavnim navodjenjem u naredbi `'print'`, gde se stringovi razdvajaju zarezima.

```
1000 var ime1 = "Swift"
1002 var ime2 = "Java"
1003 var ime3 = "Python"
1004 var ime3 = ""
1006
1007 print(ime1, ime2, ime3, separator: ", ", terminator: "")
1008 print(ime1, ime2, ime3, separator: ", ", terminator: "", to:&
    ime4)
```


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 x:Int = 0
1002 for x in 0...10 {
1004     print(x)
1006 }
1008 var y:String = ""
1010 while y != "aaaaa" {
1012     print(y)
1014     y = y + "a"
1016 }
```

U ovom primeru je pokazana funkcionalnost `for` i `while` petlje. Takođe još 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 score:Int = 0
1002 var currency:Float = 0
1004 func addToScore(points:Int , money:Float){
1006     score = score + points
1008     currency = currency + money
1010 }
1012 addToScore(points: 30, money: 1.45)
1014 addToScore(points: 60, money: 2.86)
1016 print(score)
1018 print(currency)
```

Pozivanje funkcije sa parametrima koja nema povratnu vrednost. Nakon dva poziva, u kojima moramo proslediti dva argumenta funkcije, bice ispisana dva broja (90-score i 4.31-currency).

```
1000 var x:Int = 1
1002 func function() -> Bool {
1004     if(x == 0){
1006         return false
1008     } else {
1010         return true
1012     }
1014 }
1016 var y:Bool = function()
```

Pozivanje funkcije koja ima povratnu vrednost. Nakon poziva povratna vrednost funkcije u ovom slučaju je `true`, koju ujedno dobija i promenljiva `y`.

8 Sve ono što je specifično i važno za sam taj programski jezik

Moj deo
Moj deo
Moj deo
Moj deo
Moj deo
Moj deo
Moj deo

9 Zaključak

Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.
Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.