Učenje programiranja pomoću aplikacije Swift Playgrounds

Percan, Tomislav

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:105697

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

«Dr. Mijo Mirković»

Tomislav Percan

Učenje programiranja pomoću aplikacije *Swift Playgrounds*

Završni rad

Pula, kolovoz 2019.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

Tomislav Percan

Učenje programiranja pomoću aplikacije *Swift Playgrounds*

Završni rad

JMBAG: 0303046799; redoviti student

Studijski smjer: Poslovna informatika

Kolegij: Osnove programiranja

Mentor: Doc. dr. sc. Tihomir Orehovački

Pula, kolovoz 2019

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student

_____

U Puli, kolovoz, 2019. godine

IZJAVA

o korištenju autorskog djela

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

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

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. Uvod 1
2. Swift 3
2.1. Ključne riječi i interpunkcijski znakovi 4
2.2. Osnovni operatori 4
2.3. Varijable 6
2.4. Konstante 7
2.5. Komentari 8
2.6. *If, if else* i *if else if* naredba 9
2.7. Petlje 11
2.7.1. *For – in* petlja 11
2.7.2. *While* petlja 12
2.7.3. *Repeat while* petlja 13
2.8. Strukture i klase 13
2.9. *Playground* 14
3. *Swift playgrounds* APP 16
3.1. *Learn to code* 1 16
3.2. *Learn to code* 2 18
3.3. *Playground* „Osobni podaci“ 18
3.4. *Lego* 19
3.5. *Sphero* 20
3.6. *Parrot* 21
4. Usporedba C++ i *Swift* programskog jezika 22
4.1. Ključne riječi 22
4.2. Osnovni operatori 23
4.3. Varijable 24
4.4. Konstante 26
4.5. *If, if else, else* 28
4.6. Petlje 28
4.7. Primjer usporedbe 30
5. Zaključak 32
Popis literature 34
Popis slika i tablica 35
Popis slika 35
Popis tablica 36

6. Sažetak 37
7. Summary 38

1. Uvod

Tema ovog završnog rada pod nazivom „Učenje programiranja pomoću aplikacije *Swift playgrounds*“ odabrana je uslijed velikog interesa za Apple proizvode i „IOS“ sučelje.

Većina ljudi 21. stoljeća, kako u privatnom tako i u poslovnom životu, okrenuta je informatizaciji, te programiranju u pravom smislu riječi. Iako vlada mišljenje da je za programiranje potrebno veliko znanje i vještina u savladavanju određenih programskih jezika i kako se to „ne može samo tako naučiti“, zapravo se ne primjećuje da se radi o svakodnevnom „programiranju“. Mnoga se rješenja čine banalnim i posve jednostavnim, ali činjenica je da čitav niz istomišljenika i znanstvenika nije proučavalo i razvijalo te jednostavne sustave, kao što su; programatori za pokretanje navodnjavanja, samoposlužni aparati s *Coca-Colom*, automati za naplatu parkiranja i ostali, bili bi ljudima totalna nepoznanica. U osnovnoškolskim danima susret s programskim jezicima kao što su Terrapin Logo bili su gotovo svakodnevni, a da tada nije postojala svijest o programiranju, već se na to gledalo samo kao na igricu upravljanja kornjačom.

Na Worldwide Developers Conference (WWDC konferenciji) 2014. godine tvrtka Apple po prvi puta javnosti predstavlja objektno orijentiran programski jezik pod nazivom „Swift“ koji je predstavio novu eru programiranja aplikacija. U odnosu na tad već dobro poznatoj aplikaciji *Objective-C,* Swift je moderan, brz i interaktivan što su karakteristike koje ga danas postavljaju na sam vrh među programskim jezicima. Namijenjen je razvoju mobilnih aplikacija u okviru *Cocoa* i *Cocoa Touch* korisničkog sučelja na iOS uređajima, kao što su *iPhone, iPad, watchOS, tvOS*. Godine 2016. na istoimenoj konferenciji, *Apple* ne prestaje oduševljavati svoje korisnike, te po prvi puta predstavlja javnosti *Swift playgrounds* aplikaciju. Aplikacija je konstruirana za studente i učenike, a s ciljem učenja *Swift* programskog jezika na jedan drukčiji način nego što je to do sada bilo uobičajeno. Sama aplikacija je glavna tema ovoga završnog rada, te je rad u cijelosti pisan na jednostavan i razumljiv način kako bi mogao poslužiti početnicima da upoznaju osnovne pojmove programiranja kao što su varijable, konstante, ključne riječi, operatori, petlje i drugi.

Prvo poglavlje pod nazivom *Swift* korišteno je s ciljem upoznavanja s programskim jezikom. Svako potpoglavlje unutar poglavlja *Swift* prikazano je i objašnjeno konkretnim primjerom kako bi čitatelj mogao lakše razumjeti tematiku. Na samome kraju prvoga djela objašnjen je pojam *Playground,* u doslovnome prijevodu „igralište“ u kojem se može vidjeti svaka

promjena tijekom pisanja koda što uvelike olakšava njegovo pisanje i nudi mnogo lakši i jednostavniji način za pronalazak grešaka.

U drugome dijelu rada fokus je stavljen na *Swift playground* aplikaciju, što je i naziv samoga završnog rada. *Swift* aplikaciju moguće je besplatno preuzeti iz trgovine *(APP Store)* samo na *Ipad* uređaje. *Learn to code* nazivi su prvih lekcija s kojima se korisnik susreće te se u njima nastoji upoznati sa svim osnovama *Swift* jezika koji je mnogo prihvatljiviji mlađim generacijama jer kroz igru upravlja likom po imenu *Byte*. *Apple* i u samom nazivu glavnog lika aplikacije ostaje u fokusu računarstva i lik naziva po mjernoj jedinici za količinu podataka. U poglavlju *Playground* koje je nazvano Osobni podaci prikazano je kako je vrlo lako moguće napraviti jednostavan program u koji će se unositi podaci koji će program ispisati na ekranu. Kraj drugoga dijela krase *Lego* kockice, *Parott* i *Sphero* roboti koje je moguće navoditi pomoću preuzetih *playgrounda*.

Treći dio završnog rada predstavlja usporedbu između *Swift-a* i *C++* te donosi prednosti jednog, te nedostatke drugog u određenim segmentima. Usporedba je napravljena s programskim jezikom C++ s koji je u programu prve godine preddiplomskog studija Poslovne informatike.

2. Swift

Swift je *Apple-ov* programski jezik predstavljen javnosti 2014. godine na WWDC[1] konferenciji. Nakon dugog niza godina stvaranja *Swift-a* za koji su najvećim djelom zaslužni Chris Lattner, Doug Gregor, John McCall, Ted Kremenek, Joe Groff i *Apple Inc.*, novi jezik kreiran je za iOS, MacOS, WatchOS, tvOS i Linux sučelje. *Swift* je vrlo siguran, moderan, brz i interaktivan programski jezik koji je načinjen kako bi zadovoljio rad s *Apple Cocoa* i *Cocoa Touch* okvir-om tekako bi funkcionirao i s već postojećim *Objective-C* kodom napisanim za veliki broj *Apple* proizvoda. *Apple* je s *Swift-om* uspio ujediniti *Objective C* i *C* u jednoj aplikaciji. *Swift* koristi LLVM kompajler koji je napisan *C++* kodom te osim *Swifta* ima mogućnost pretvorbe (engl. compile) tj prevođenja raznih programskih jezika kao što su: *C, Objective-C Java, Delphi, Scala, Phython, Ruby, Dylan* i drugi. Brzina je jedan od velikih aduta *Swift*-a, kada govorimo o složenim vrstama objekata, *Swift* je gotovo dvostruko brži od *Objective-C*. Koristi se RC4 enkripcijom koja je kod *Swifta* opet dvostruko brža nego kod *Objective-C* ili čak 220 puta brža nego kod *Pythona.*

Swift pruža svoju verziju svih osnova *C* i *Objective-C* kao što su (int) za broj, (*double* i *float)* za brojeve s većim brojem decimala, (*bool*) za logičke vrijednosti i (*string*) za tekstualne podatke. *Swift* je vrlo jednostavan za korištenje, namijenjen je kako apsolutnim početnicima tako i vrlo iskusnim programerima koji su upoznati s nekom od inačica C jezika. Dizajniran je za najjednostavnije programe kao što je *Hello World,* ali i za najobuhvatnije operativne sustave.

Kada govorimo o *Swiftu* najčešće govorimo o verziji 4.1. koja je uključena u *Xcode 9.2*. Prilikom programiranja u *Xcode-u 9.2*. može se koristiti bilo koja verzija *Swif 4* ili *Swift 3*.

Pisanje *swift-a* u „*playground*[2] jednako je zanimljivo kao i dječja igra u „*playgroundu*“ iz vrlo jednostavnoga razloga što nudi mogućnost testiranja koda kada se to poželi, bez ikakvog pokretanja aplikacije. Velika razlika se može primijetiti kod samog pisanja koda, s obzirom da u jezicima kao što je *C++* na kraju svake linije koda treba staviti „**;**“ kako bi se završila ta linija, a što kod *Swift-a* nije slučaj.

[1] WWDC (Worldwide Developers Conference)
[2] Igralište

2.1. Ključne riječi i interpunkcijski znakovi

Ključne riječi korištene u deklaracijama: *associatedtype, class, deinit, enum, extension, fileprivate, func, import, init, inout, internal, let, open, operator, private, protocol, public, static, struct, subscript, typealias i var.*

Ključne riječi korištene u izjavama: *break, case, continue, default, defer, do, else, fallthrough, for, guard, if, in, repeat, return, switch, where, i while.*

Ključne riječi korištene u izrazima i vrstama: *as, Any, catch, false, is, nil, rethrows, super, self, Self, throw, throws, true, i try.*

Ključne riječi korištene u obrascima: _.

Ključne riječi koje započinjubrojem(#): #available, #colorLiteral, #column, #else, #elseif, #endif, #file, #fileLiteral, #function, #if, #imageLiteral, #line, #selector, i #sourceLocation.

Ključne riječi posebnog konteksta: *associativity, convenience, dynamic, didSet, final, get, infix, indirect, lazy, left, mutating, none, nonmutating, optional, override, postfix, precedence, prefix, Protocol, required, right, set, Type, unowned, weak, i willSet.*

Znakovi rezervirani kao interpunkcijski znakovi: (,), {, }, [,], ., ,, :, ;, =, @, #, & (as a prefix operator), ->, `, ?, and !.

2.2. Osnovni operatori

Operatori su simboli pomoću kojih se dodjeljuju, provjeravaju ili mijenjaju vrijednosti, to su simboli ili znakovi koji nose specifičnu akciju.

Tabica 1. Aritmetički operatori (internetski izvor)

Aritmetički operatori	
+	Operator zbrajanja
-	Operator oduzimanja
*	Operator množenja
/	Operator dijeljenja
%	Operator modularnog dijeljenja (rezultat je cjelobrojni ostatak dijeljenja dvaju brojeva)

Tablica 1. prikazuje aritmetičke operatore koji se koriste kako bi se jednoj vrijednosti pridodala, oduzela, umnožila ili podijelila vrijednost. Na primjer 1 + 1 = 2 ili 1 * 1 = 1

Tablica 2. Operatori dodjeljivanja (internetski izvor)

Operatori dodjeljivanja	
=	Jednostavni operator za dodjelu vrijednosti
+=	Operator za zbrajanje i dodjelu vrijednosti
-=	Operator za oduzimanje i dodjelu vrijednosti
*=	Operator za množenje i dodjelu vrijednosti
/=	Operator za dijeljenje i dodjelu vrijednosti
%=	Operator modulo i dodjela vrijednosti
<<=	Operator manje ili jednako
>>=	Operator veće ili jednako
&=	Operator I i dodjela vrijednosti
^=	Isključujući operator ILI i dodjela vrijednosti
|=	Uključujući operator ILI i dodjela vrijednosti

Operatori dodjeljivanja služe, kako i sam naziv govori, da se određenom atributu doda neka vrijednost.

Slika 1. Dodjeljivanje vrijednosti (vlastita izrada)

Slika 1. predstavlja primjer iz kojeg je vidljivo da na početku varijabla „a“ ima vrijednost 5, a kasnije se taj „a“ opet uvećao za 5 što je u konačnici rezultiralo rezultatom a = 10.

Tablica 3. Logički operatori (internetski izvor)

Logički operatori	
||	Logički ILI
&&	Logički I

Logički operator ILI koristi se kada ishod nekog zadatka može biti rješiv na dva načina, odnosno jedan od dva ponuđena rješenja mora imati vrijednost „true“ (točno). Logički I operator se koristi kod rješavanja problema gdje oba uvjeta moraju biti zadovoljena, odnosno oba moraju biti „true“, kako bi rješenje bilo potvrdno. Ukoliko jedan od dva rješenja ima „false“ (krivo), sveukupno rješenje biti će netočno.

Operatori usporedbe	
= =	Jednako
!=	Nije jednako
>	Veće
<	Manje
>=	Veće ili jednako
<=	Manje ili jednako

Tablica 4. Operatori usporedbe (internetski izvor)

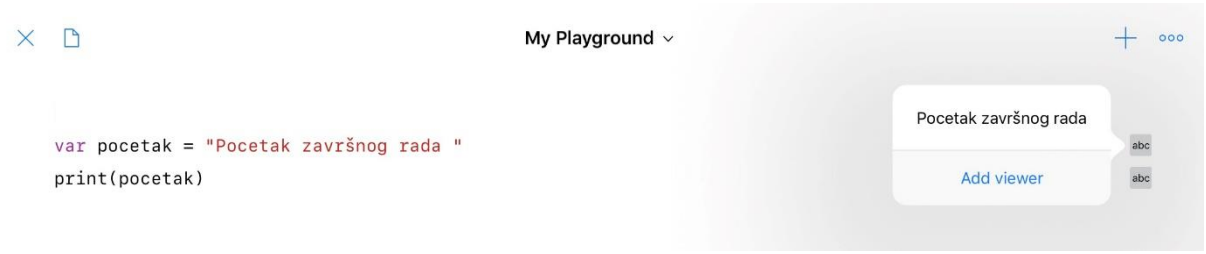
Operatori usporedbe vrlo se intenzivno koriste u procesu pisanja koda. Kako bi se mogli učiniti programi koji će funkcionirati na temelju usporedbe, kao npr. ispiši sve brojeve koji su > (veći) od 20 i < (manji) od 150. U konkretnom primjeru program će ispisati sve brojeve veće od 20 i manje od 150, dakle tu će biti ispisani brojevi 21,22,23….149, no da se uvjet postavio na način: >=20 i <=150 tu bi bili ispisani svi brojevi kao i prije, uključujući brojeve 20 i 150.

2.3. Varijable

Naziv varijabla dolazi iz engleskog jezika, *variable* što u prijevodu znači promjenjivo. Varijablu se može definirati i kao skladište podataka te se unutar tog skladišta mogu uskladištiti razni podaci koji se mogu mijenjati tijekom pisanja koda.

Varijable se deklariraju kako bi se mogle koristiti, a to činimo pomoću ključne riječi „var“, imena varijable te vrijednosti same varijable. Varijable su promjenjive te ih tokom pisanja koda možemo mijenjati, uvećavati, smanjivati ili totalno zamijeniti nekom drugom varijablom. Svaka varijabla može imati i različit tip kako bi se moglo u nju uskladištiti različite vrijednosti koje mogu biti; slova, brojevi, brojevi s pomičnim zarezom, logičke vrijednosti.

var ime_varijable = vrijednost_varijable



Slika 2. Deklaracija varijable (vlastita izrada)

Kao što je vidljivo iz slike 2. kada se koriste varijable nije nužno definirati tip varijable, no to se može učiniti na način koji prikazuje slika 3.

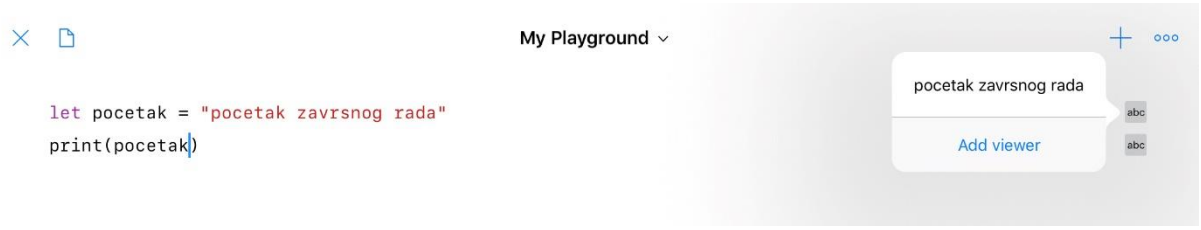
Slika 3. Određivanje tipa varijabli (vlastita izrada)

Slika 3. prikazuje u konkretnom primjeru kako se koristi varijabla koja nije bila definirana, i u drugome slučaju koja je bila definirana s *floating point-om „float*“, odnosno brojem koji ne mora biti cijeli. Prvu varijablu se također moglo definirati s i*ntegerom* (int) s obzirom da broj učenika može biti isključivo cijeli broj.

2.4. Konstante

Kako i sama riječ konstanta govori, radi se o stalnim i nepromjenjivim vrijednostima. . Jednom deklarirana konstanta ostaje jednaka tijekom pisanja cijeloga koda. Deklaracija konstanti u *Swift-u* odvija se pomoću ključne riječi „let“. Pri korištenju konstante najčešće se misli na brojevne konstante koje moraju biti takve kakve jesu, ne mijenjaju se i uvijek su u tome obliku, a jedna od njih svakako je π s uvijek istom vrijednošću 3.14.

let ime_konstante = vrijednost_konstante



Slika 4. Deklaracija konstanti (vlastita izrada)

Istim postupkom kao i varijable definiraju se i konstante.

U primjeru koji slijedi (Slika 5.) definiran je tip konstante. U konkretnim primjerima koristio se *floating point „float*“ ali mogao se koristiti i „double“ koji ima jednaku namjenu, a jedina razlika među njima je način na koji oni skladište podatke.

Slika 5. Konstante (mora biti!)

Vrijednosti varijabli i konstanti ispisuje se funkcijom Ispiši (engl. *print*) jer se koristi *Swiftplayground* aplikacija, no u slučaju korištenja *Xcode-a* ta bi naredba glasila *println*.

2.5. Komentari

Jednako kao i kod programskog jezika *C* ili *C++* komentar se može upisati među linije koda što služi kao podsjetnik na upisani kod, ali se koristi i kao obrazloženje drugomu programeru kako bi vidio u određenoj liniji koda što je u kodu prethodno napisano. Svaki program ili Swift kompajler ignorira bilo koji komentar, odnosno sve što je zapisano u tome komentaru prilikom pokretanja programa nije vidljivo. Kod korištenja više linijskog komentara treba obratiti pažnju je li komentar na kraju zatvoren, jer ako nije, cijeli će ostatak koda biti zapisan kao komentar.

Komentari mogu biti jednolinijski:

```
1   //Ovo je moj prvi jednolinijski komentar!
2   //Ovo je moj drugi jednolinijski komentar!
```

Slika 6. Jednolinijski komentar (vlastita izrada)

Komentari mogu biti i višelinijski:

```
1   /* Ovo je jedan više
2   linijski komentar,
3   koji se proteže u 3 reda. */
```

Slika7. Višelinijski komentar (vlastita izrada)

Za razliku od *C* i *C++* kod *Swifta* može se koristiti komentar unutar komentara.

```
1   /* Ovo je početak prvog
2   višelinijskog komentara.
3
4   /* Ovo je drugi
5   višelinijski komentar unutar prvog komentara*/
6
7   /* Ovo je kraj prvog
8   višelinijskog komentara*/
9
```

Slika 8. Komentar unutar komentara (vlastita izrada)

2.6. *If, if else* i *if else if* naredba

If naredba omogućuje izvršavanje koda samo i jedino ako je uvjet zadovoljen, ukoliko nije, program neće izvršiti nikakvu radnju. Ako se od korisnika zahtjeva da napiše kod gdje postoje dvije varijante ishoda funkcije programa, koristi se *if else* naredba. *If else* naredba vrlo je korisna ukoliko je nužno na određeno pitanje dobiti konačan odgovor.

If else if naredba je proširena *if else* naredba pomoću koje je moguće ispitati veći broj uvjeta.

If

```
1.  let broj = 5
2.  if broj = 5 {
3.          print("Upisan je točan broj!")
4.  }
```

If naredba koristi se za provjeru točno određene vrijednosti. Prilikom pokretanja programa, na ekranu bi se pojavio tekst: „Upisan je točan broj!“ Razlog takvog rješenja skriva se u prvoj i drugoj liniji koda gdje je upisana vrijednost 5, a uvjet je da mora biti upisan točno taj broj 5.

If else

```
let broj = 5
if broj > 0 {
        print("Broj je pozitivan!")
} else {
        print("Broj je negativan! ")
}
```

If else naredba pruža mogućnost provjere valjanosti tvrdnje. Program ispisan ovim kodom ispisao bi: „Broj je pozitivan!“, jer je na početku zadana vrijednost broj 5 koji označava pozitivnu vrijednost.

If else if

```
1.  let broj = 0
2.  if broj > 0 {
3.          print("Broj je pozitivan!")
4.  }
5.  else if (broj < 0) {
6.          print("Broj je negativan!")
7.  }
8.  else {
9.          print("Broj je 0!")
10. }
```

If else if primjer prikazuje naredbe s moguće tri varijante ishoda određenog koda. Program bi, u slučaju kada je napisana još neka linija koda, poslao upit da se upiše određeni broj, na temelju kojeg bi potom ispisao jednu od mogućih varijanti; „Broj je pozitivan“, „Broj je negativan“ ili „Broj je nula“. Kako je na početku koda upisana konstanta let broj = 0; program će u ovom slučaju ispisati: „Broj je 0!“.

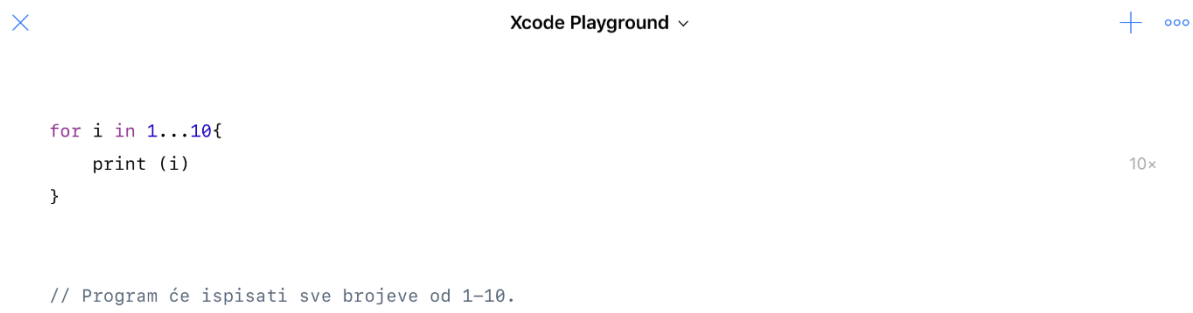
2.7. Petlje

Ponekad se dijelovi programa trebaju izvršiti više puta (iterirati). Struktura koja to omogućuje naziva se programska petlja. Programskom petljom dijelovi programa mogu se ponavljati unaprijed zadani broj puta, ili, sve dok je određeni uvjet ispunjen.[3] Programske petlje nazivaju se još i naredbama za ponavljanje.

Petlje se ostvaruju naredbama: *for-in, while i repat while.*

2.7.1. *For – in* petlja

[3] https://sites.google.com/site/sandasutalo/osnove-programiranja/programska-petlja



Slika 9. *For-in* petlja (vlastita izrada)

Slika 9. prikazuje jedan jednostavan primjer *for-in* petlje. Kao što je vidljivo iz primjera varijablu „i“ nije bilo potrebno deklarirati jer se pretpostavlja da je to niz cijelih brojeva odnosno engl. *integer*.

2.7.2. *While* petlja

Slika 10. *While* petlja (vlastita izrada)

While petlja koristi se kada je potrebno da se neka radnja odvija sve dok uvjet nije zadovoljen. U konkretnom primjeru koji prikazuje slika 10., *While* petlja će se odvijati sve dok ne ispiše sve brojeve od 1-10.

2.7.3. *Repeat while* petlja

Slika 11. *Repeat while* petlja (vlastita izrada)

Repeat while ili ponavljajuća *while* petlja ponavljati će se onoliki broj puta dok uvjet *while* < *10* ne bude zadovoljen, odnosno dok program ne ispiše sve brojeve od 1-9 uključujući i broj 9. Da je uvjet postavljen kao u prethodnome primjeru odnosno *while* <= *10* program bi ispisivao sve brojeve uključujući i broj 10.

2.8. Strukture i klase

Strukture i klase imaju sličnu sintaksu, strukture se obilježavaju s ključnom riječi *struct* dok se klase označavaju s ključnom riječi *class*. Cijela definicija odnosno naredba se stavlja u par vitičastih zagrada. Među strukturama i klasama kao i u drugim programskim jezicima postoje razlike, a jedna od najvažnijih razlika između struktura i klasa je da se strukture uvijek kopiraju kada se prosljeđuju u kodu, a klase se prosljeđuju prema referenci.“[4]

Klase i strukture u *Swiftu* imaju mnogo toga zajedničkog:[5]

- Definiranje svojstva za pohranu vrijednosti
- Definiranje metode za pružanje funkcionalnosti
- Definiranje indexa kako bi se omogućio pristup njihovim vrijednostima pomoću sintakse indeks
- Definiranje inicijalizatora za postavljanje početnog stanja
- Mogu biti prošireni kako bi proširili svoju funkcionalnost izvan zadane implementacije
- U skladu su s protokolima za pružanje standardne funkcionalnosti određene vrste

[4] Swift programski jezik (Swift 4.1), Apple INC, str 40.
[5] Swift programski jezik (Swift 4.1), Apple INC, str 364.

```
1   struct Resolution {
2       var width = 0
3       var height = 0
4   }
5   class VideoMode {
6       var resolution = Resolution()
7       var interlaced = false
8       var frameRate = 0.0
9       var name: String?
10  }
```

Slika 13. Strukture i klase (*Apple* inc. 367 str)[6]

Slika 13. prikazuje strukturu koja se naziva *Resolution* te ima pohranjena dva svojstva, a to su: *width* i *height*. U istome primjeru definirana je i klasa *Video Mode* koja ima 4 varijable: *resolution, interlaced, frameRate, name.*

2.9. Playground

Playground je alat koji se po prvi puta pojavljuje u verziji *Xcode-a* 6. *Playground* pojednostavljuje pisanje koda za razvoj aplikacija i programa te se u samom tijeku pisanja koda dobiva na uvid i rezultat izvršavanja tog koda. Prilikom pisanja koda u desnom dijelu ekrana odvija se radnja toga koda, a sve trenutne promjene u kodu rezultiraju automatskom promjenom i u desnom dijelu ekrana gdje se odvija radnja toga koda. Programeru se tijekom pisanja koda nudi i mogućnost *Quick look-a* kojom je omogućen pregled vrijednosti te ima mogućnost prikazivanja boja, nizova, slika, klasa, struktura…

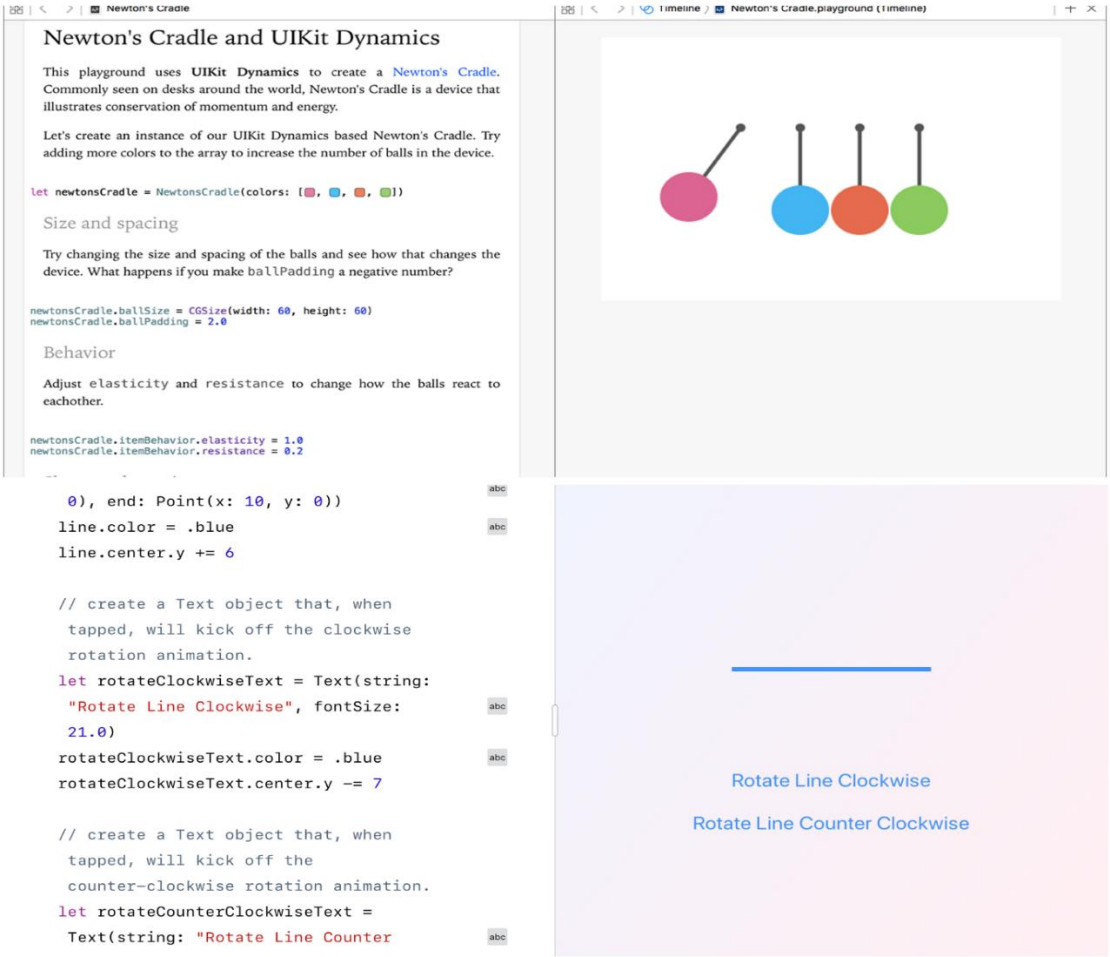
Tim se načinom pisanja koda uvelike smanjuje mogućnost pogreške. Neke od mogućnosti u *Playgroundu* kojima se olakšava pisanje koda su:

- ***XCPShowView*** - prikaz vremenskih prikaza uživo.
- ***XCPSetExecutionShouldContinueIndefinitely***. - omogućuje nastavak izvođenja čak i nakon dostizanja koda najviše razine u playground-u.
- ***XCPCaptureValue*** - ručno snimanje vrijednosti.
- ***XCPSharedDataDirectoryPath*** - dohvaća putanju direktorija koja sadrži dijeljene podatke između svih playground-a.

[6] Swift programski jezik (Swift 4.1), Apple INC, str 367.

Iako *Playground* ima mnoge pozitivne strane, dakako da nije svemoguć, te su ovo neke od stvari koje ne podržava:

- ne može se koristiti za testiranje izvedbe,
- ne podržava korisničku interakciju,
- ne podržava izvršavanje na uređaju.



Slika 14. *Xcode playground – Swift playground* app (vlastita izrada)

Slika 14. prikazuje dvije verzije *playgrounda*, u prvom djelu prikazan je *playground* dostupan u *X-codeu* na *Mac* uređajima, a drugi dio slike prikazuje *playground* koji je korišten pri izradi rada te je dostupan u *Swift playgrounds* aplikaciji.

3. Swift playgrounds APP

Swift playground je *Apple*-ova aplikacija namijenjena *iPad* korisnicima čija dob može biti od +2 do 99+ godina. Aplikacija je vrlo interaktivna i zabavna, ne zahtjeva ranije poznavanje koda niti bilo čega što je vezano uz kod i programiranje.

Swift playground integriran je u *X-code* te je kao takav prvi puta predstavljen javnosti 2014. godine na WWDC konferenciji. Chris Lattner jedan od izumitelja *Swift* programskog jezika kao i izumitelj same aplikacije. Osim što je izumitelj, Chriss Lattner je direktor te arhitekt na odjelu za razvojne alate.

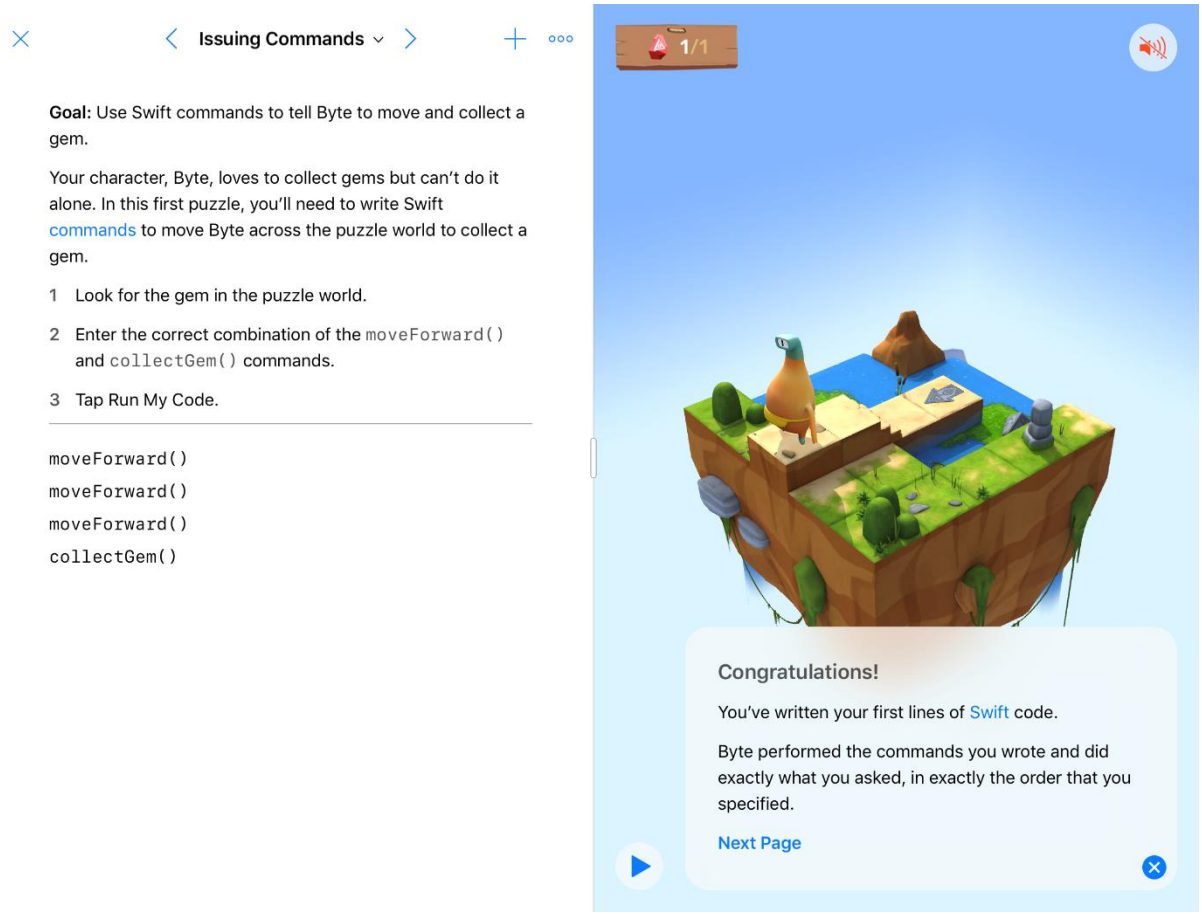
13. rujna 2016. godine po prvi puta se *Apple*-ovi korisnici susreću sa *Swift Playgrounds* aplikacijom namijenjenoj za pokretanje na iPadu. Aplikacija je predstavljena kao nastavni alat za učenike i studente uvodeći koncepte kodiranja na vrlo interaktivan način na tabletu, dostupna je na većem broju jezika kao što su engleski, njemački, talijanski, portugalski, korejski, kineski što dokazuje da *Apple* misli i djeluje globalno.

Veliki *korak* je napravljen time što se aplikacija nalazi na tablet uređaju kojeg korisnici imaju mogućnost koristiti u ugođaju svojeg doma, školskoj klupi, autobusu, na odmoru i sl. Korištenjem aplikacije korisnici mogu preuzeti datoteke lekcija kreiranih od *Apple-a* ili nekog drugog poslužitelja. Lekcije se kreću od iznimno lakih do ozbiljnih i vrlo zahtjevnih. Korisnik se na samome početku susreće s prvom lekcijom pod nazivom *Learn to code 1* te savladavanjem lekcije postepeno može rasti i preuzimati sve zahtjevnije i sve obuhvatnije lekcije. Preporuča se učenje pomoću prve dvije lekcije *Learn to code 1,2* kako bi korisnik bio upućen u sve što se koristi u ostalim zadacima. Učenjem *Learn to code* korisnik se susreće s varijablama, konstantama, petljama, nizovima, komentarima… Nakon šte se korisnik poduči u dovoljnoj mjeri o *swiftu* može početi koristiti *Explore challenges* gdje će preuzimati redizajnirane *playgrounde* i pokušati riješiti problem.

U aplikaciji iBooks svaki korisnik može pronaći i preuzeti pisane verzije vodiča kroz aplikaciju.

3.1. *Learn to code* 1

Naziv prve lekcije je *Learn to code* i kao što sam naziv govori kroz tu lekciju uče se osnovne stvari. Na slici koja slijedi prikazano je kako bi budući programer morao razmišljati logički ali i jednostavno. Zadatak je da lik po imenu Byte skuplja „dragulje“ i tako u svakoj sljedećoj lekciji što najprije izgleda vrlo jednostavno no svaka sljedeća lekcija postaje sve zahtjevnija.



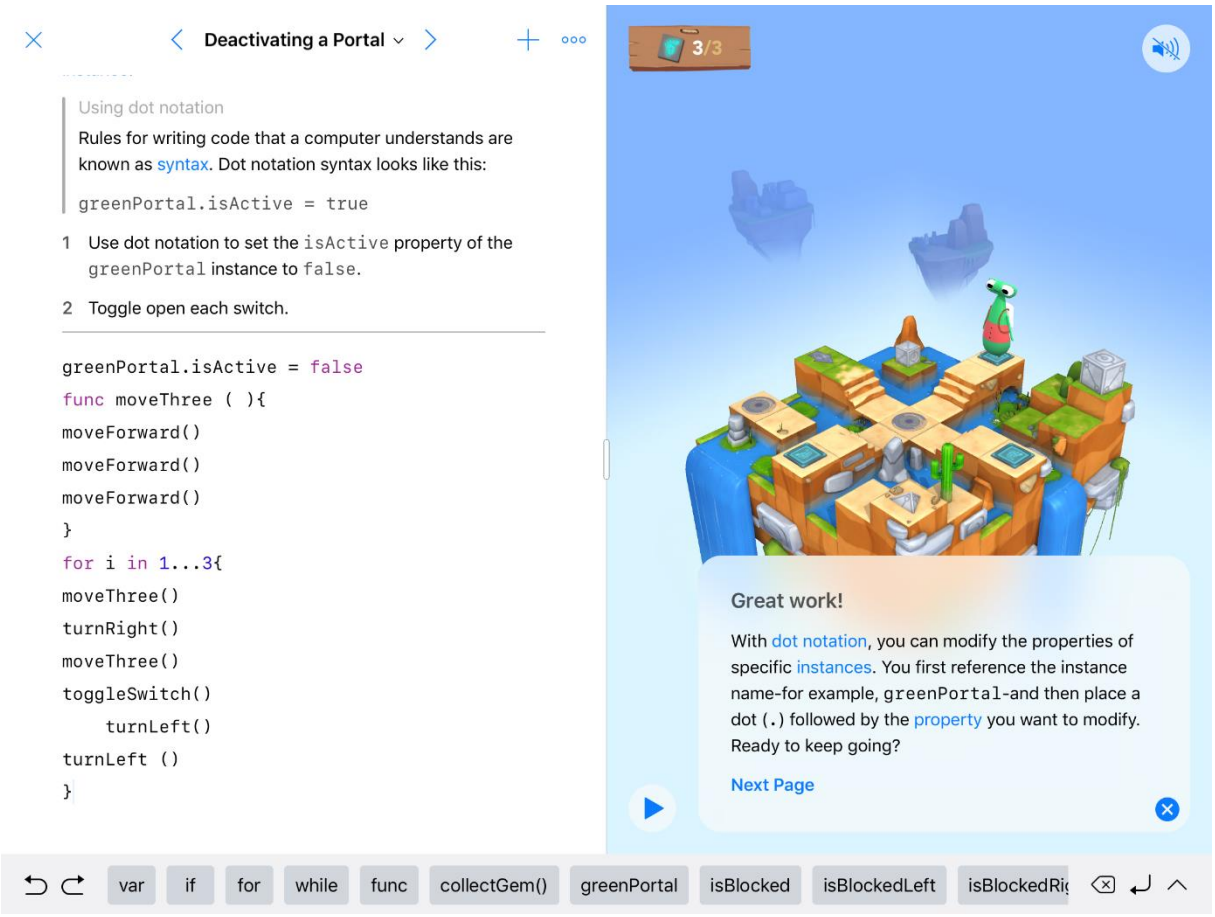
Slika 15. *Learn to code* 1 (vlastita izrada)

Savladavanjem dijelova lekcije prelazi se na sljedeću, no prije svake lekcije slikovno je prikazano što će se u narednom periodu naučiti. Tako se kroz *Learn to code* koji je podijeljen u 7 dijelova korisnik susreće s *Commands* – Komandama, *Functions* – Funkcijama, *For loops* – For petljama, *Conditional code*, *Logical operators* – Logičkim operatorima, *While loops* – While petljama, *Algorithms* – Algoritmima

Korištenjem *Learn to code* priručnika pojednostavljuje se pisanje koda, na svakom koraku korisnik ima mogućnost sam pisati kod ili se koristiti već gotovim naredbama, kao što su *moveForward, lurnLeft, collectGem*. Vrlo korisna stvar koju su *Apple*-ovi rezvojni programeri implementirali u aplikaciju, zasigurno potaknuti problemima koji se događaju u njihovom svakodnevnom radu, jest pogreška u pisanju koda. Pogreške mogu biti od krajnje jednostavnih kao što je krivo upisano određeno slovo do krivo pozvanih funkcija, te aplikacija nudi razne mogućnosti popravka koda. Ukoliko korisnik nije svjestan pogreške otvara mu se mogućnost korištenja izbornika gdje korisnik klikne na naredbu *Fix* te aplikacija po svojoj logici koju je dobila iz dijelova napisanog koda rješava nedostatke u kodu dok se kod u potpunosti ne ispravi.

3.2. *Learn to code* 2

Drugi dio ili *Learn to code* 2 podijeljen je u 6 dijelova. Korisnik se susreće sa varijablama i njihovom deklaracijom, tipovima, inicijalizacijom, parametrima, *World Building* i nizovima. Kao i u *Learn to code* 1 korisnik ima zadatak pisanjem koda ili odabirom već ponuđenih ključnih riječi dovesti *Byte-a* do kraja svake staze. Ukoliko korisnik nije savladao neki od prethodnih dijelova, ili mu nije nešto potpuno jasno, u izborniku ima mogućnost odabira jednog od već gotovih ponuđenih rješenja, te tako završiti određenu razinu.

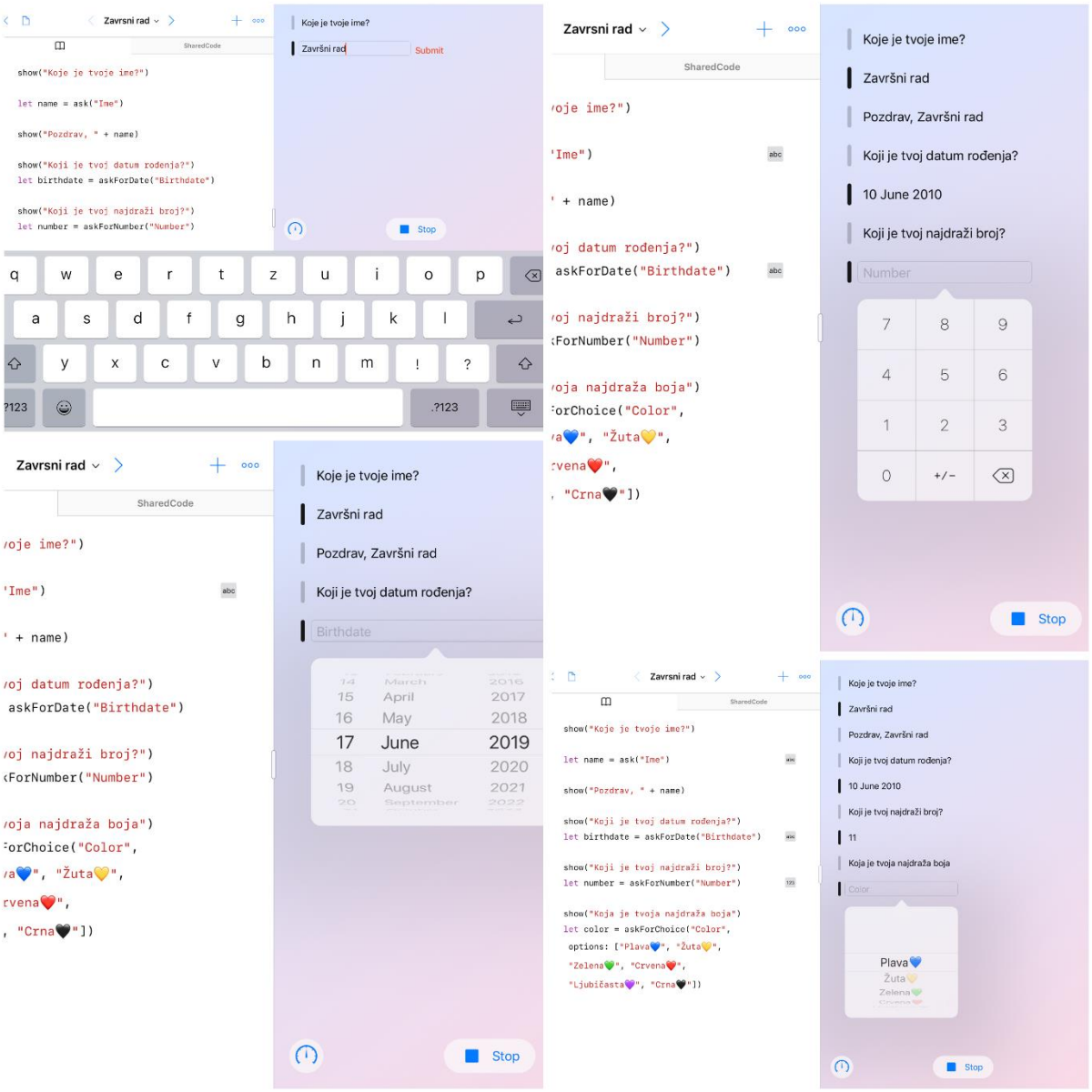


Slika 16. *Learn to code* 2 (vlastita izrada)

3.3. *Playground* „Osobni podaci“

*Playground „*Osobni podaci“ osmišljen je na temelju gotovog *playgrounda* koji je izmijenjen te kodiran na način kako bi mogao zadovoljiti željeno. Cilj je bio stvoriti kod koji traži od korisnika da upisuje podatke kao što su ime, datum rođenja, najdraži broj i najdraža boja. Kako je *playground* osmišljen za uporabu na *iPad-u* tako prilikom pokretanja odnosno

prilikom odgovaranja na pitanja, program nam dodjeljuje tipkovnicu kako bismo mogli unijeti ime, nudi nam kalendar pomoću kojeg možemo unijeti datum rođenja i na kraju nam daje izbor od više mogućnosti kako bi korisnik mogao odabrati jednu od ponuđenih boja. Slika 15. prikazuje korak po korak te sam izgled *playgrounda*.



Slika 17. Osobni podaci (vlastita izrada)

3.4. Lego

Da je Apple shvatio važnost učenja programiranja najmlađih generacija što želi popularizirati i potaknuti u najvećem mogućem smislu, dokazuje i suradnja s jednom od najpoznatijih dječjih igračka *Lego-m.* Danas *Lego* nema više cilj samo razviti kreativnost stvaranjem personaliziranog lika, dvoraca, automobila, već potaknuti pripadnike najmlađe populacije na

razmišljanje da njihov lik može biti i upravljiv i koristan. *Apple* i *Lego* osmislili su način kojim će ujediniti zajednička iskustva i vrijednosti i stvoriti nešto inovativno, korisno i dosad neviđeno. Prilikom korištenja *Lego Mindstorms* likova korisnik ima mogućnost složiti svoj lik te upravljati njime pomoću *iPad* uređaja korištenjem aplikacije *Swift playgrounds*. U samoj aplikaciji moguće je upravljati likom osnovnim naredbama kao što je naprijed, natrag, lijevo i desno no moguće je i napisati linije koda koje će se ponavljati, izvršavati pomoću već naučenih *while, repeat while* petlji. Na samom liku nalazi se i ekran na kojem je moguće prikazivati slike i tekst, ugrađeni zvučnik može reproducirati zvukove, dok ugrađeni senzori mogu razlikovati boje, svijetlost i objekte. Specifikacije LEGO MINDSTORMS EV3 INTELLIGENT BRICK su:[7]

- *A faster ARM9 microcontroller*
- *16 MB flash memory*
- *64 MB RAM*
- *A SD memory slot allowing to expand RAM memory*
- *An open-source Linux operating system*
- *A 2.0 USB interface to allow Wifi connection*
- *4 input ports and 4 output ports allowing up to 4 NXT bricks to be connected together*
- *A matrix display screen*
- *A speaker*
- *Bluetooth ® v2.1*
- *iOS and Android compatibility*

3.5. Sphero

Sphero je robot koji koristi *Bluetooth* kako bi *Swift* kod učinio živim. Nudi mogućnosti personalizacije, odnosno mijenjanja boja na led zaslonu, te nudi mogućnosti upravljanja u svim smjerovima. Koristeći gotov *playground* moguće je zadati mu zadatke (napisane kodom) da prati smjer koji je kodom zadan kako bi svojim kretnjama napravio određeni geometrijski lik. Tako je moguće zadati mu da se kreće određenom brzinom određenu udaljenost te nakon prijeđene udaljenosti da skrene u lijevo i to toliko puta kako bi napravio neki geometrijski lik, npr. kvadrat. Osim što ima mogućnost spoznaje prijeđenog puta može ga se i kodirati da

[7]Generation robots, dostupno na: https://www.generationrobots.com/en/401485-lego-mindstorms-ev3-intelligent-brick.html, pristupljeno 20.06.2019.

skreće pod određenim kutovima. *Sphero Bolt* jednostavan je primjer zabavne, interaktivne i obrazovne igračke za djecu i odrasle.

Slika 18. *Sphero Bolt*

3.6. Parrot

Parrot drone kao što i naziv govori da se radi se o dronovima marke Parrot koje je moguće povezati putem *Bluetootha* na IPad uređaj. Namijenjeni su svim generacijama i u sve svrhe za razliku od *Lega* i *Sphero bolt* lopte koji su namijenjeni učenju kroz igru ili samo za igru. Dronovi se danas koriste za pretrage izgubljenih stvari i objekata, a jednako tako dobro je poznato kako su danas slike iz zraka vrlo tražene na web stranicama raznih apartmanskih kompleksa ili hotela. U *Swift playground* aplikaciji moguće je preuzeti *Parrot Education* i putem njega naučiti sve o samom dronu, njegovim mogućnostima te o samom navođenju.
Dronom se može upravljati u svim smjerovima, okretati ga oko svoje osi te mu namještati nagib. Moguće je povezati više vrsta dronova kao što su: *Rolling Spider, Airborne Cargo, Airborne Night i Mambo.*

TECHNICAL SPECIFICATIONS

STABILIZATION SENSORS
- Inertial Measurement Unit to evaluate speed, tilt and obstacle
- Contact (3-axis accelerometer and 3-axis gyroscope)
- Vertical stabilization: Ultrasound sensor (measures below 13 ft / 4 m altitude) & Pressure sensor
- Horizontal stabilization: Camera sensor

ENERGY
- 550 mAh LiPo Battery
- 8 min battery life with bumpers
- 9 min battery life with bumpers
- 30 min charging time with a 2.1 A charger

AERONAUTICAL DESIGN
- Stabilization and flight control software
- Optimized weight/speed ratio
- Polyamide and Polypropylene structure

SPEED MEASUREMENT & CAMERA
- 60 FPS vertical camera
- Camera: 300,000 pixels

RANGE
- 65 ft / 20 m with smartphone
- 200 ft / 60 m with Parrot Flypad

WEIGHT, DIMENSIONS & OS COMPATIBILITY
- 2.22 oz / 63 g without bumpers or accessories
- 7.1 x 7.1 in. / 18 x 18 cm with bumpers
- iOS 7 and up / Android 4.3 and up
- OS Linux. SDK available on developer.parrot.com

6 MONTHS TYNKER SUBSCRIPTION
- 3 Introductory Courses
- Drones and Robotics Courses
- 6 Game Design Courses
- 4 Minecraft Courses
- Private Minecraft Server
- Daily Missions

Slika 19. Tehničke karakteristike (internetski izvor)

Slika 19. prikazuje tehničke karakteristike *Parrot Mambo* uređaja, te sve što se nalazi u kutiji prilikom kupnje uređaja. 6 *Months Tynker Subscription* je *Parrot-ov* vlastiti korisnički priručnik koji je nalik *Swift playground* app, te na jednaki način putem tableta s *Tynker* aplikacijom korisnik može naučiti upravljati uređajem.

4. Usporedba C++ i *Swift* programskog jezika

Usporedba općepoznatog C++ programskog jezika koji je za većinu studenata preddiplomskih informatičkih i računarskih fakulteta baza za budućnost s modernim i „mladim“ *Swiftom,* koji je osnova mnogih aplikacija namijenjena iOS uređajima, jedna je vrlo zanimljiva tema koja predstavlja duel između „starog“ dobro prihvaćenog C++ i modernog vrlo raširenog *Swifta*. Vrlo je teško usporedbu svesti na nešto posve jednostavno odnosno dobiti odgovor na pitanje koji je od ponuđenih jezika bolji? Svakom programeru svidjet će se određeni jezik, te će vrlo vjerojatno „hvaliti“ onog koji je sam savladao. *Swift* je programski jezik koji je nastao na osnovama *Objective-C* te riješio nedostatke modernih programskih jezika koji se u njemu nisu pojavljivali. Sintaksa kod *Swifta* je mnogo jednostavnija te mnogo lakša za savladati. Sa *Swiftom* se nameće jedan veliki problem, a to je nemogućnost korištenja na *Windows* i *Linux* operativnim sustavima. Uporaba *Swifta* nameće uporabu *Xcode-a,*a *Xcode* nameće uporabu *Mac* računala. Uz samo korištenje *Mac* računala vrlo je velika vjerojatnost i nabavke *iPhone* uređaja na kojem će biti moguće testirati načinjene aplikacije.

4.1. Ključne riječi

C++ , *Swift* kao i drugi programski jezici koriste ključne riječi te se većina njih ponavlja u oba jezika. Kod C++ pojavljuju se neka ograničenja koja kod *Swifta* ne postoje te *Swift* u tim segmentima prednjači i pojednostavljuje stvari. Pisanjem koda C++ jezikom ne smiju se koristiti ključne riječi u nazivima varijabli dok je to kod *Swifta* moguće uporabom navodnih znakova. Dakle, *Swift* ima tu mogućnost da se rezervirane ključne riječi mogu koristiti za nazive varijabli, konstanti, klasa.

To je moguće učiniti na sljedeći način*:* `let `var` = "Pozdrav"` te ovaj primjer prikazuje konstantu imena „var“ tipa String „Pozdrav“.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Slika 20. Swift ključne riječi (internetski izvor)

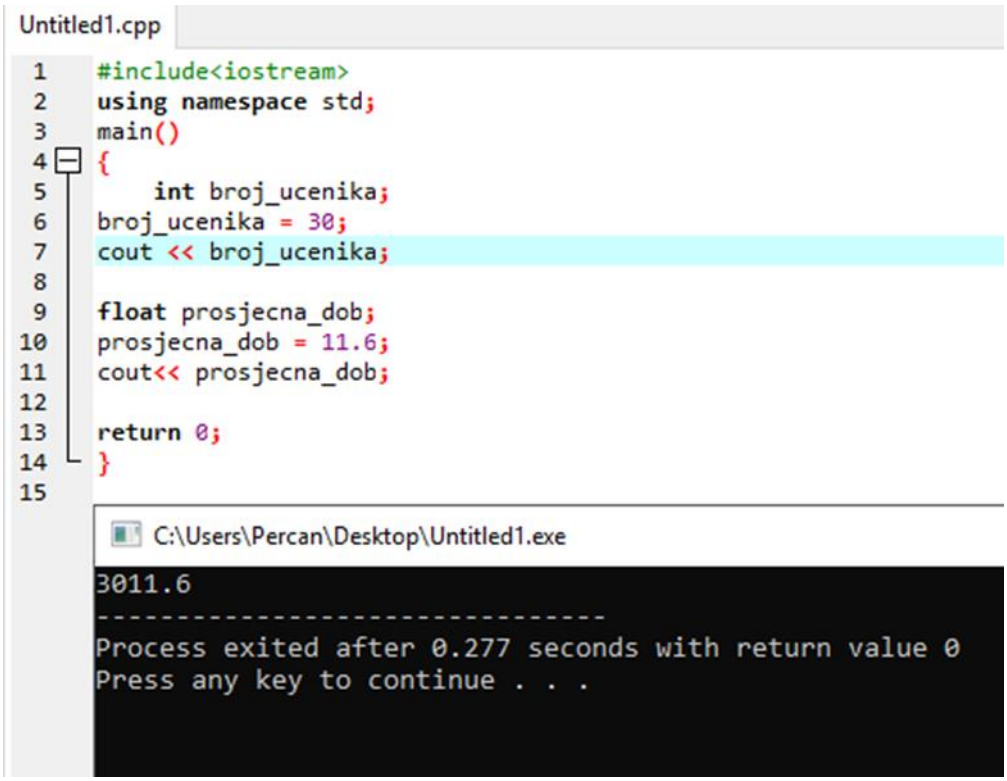
4.2. Osnovni operatori

Osnovni operatori se na jednaki način koriste u oba programska jezika no postoji razlika s aritmetičkim operatorima koje Swift detektira i ne dopušta da se u pojedini mehanizam sakupljanja pohrani vrijednost koja premašuje zadani raspon. Swift podržava operatore koji u C-u ne postoje a to su: a..<b i a...b koji predstavljaju kraticu za izražavanje raspona vrijednosti.

4.3. Varijable

Swift u nazivima varijabli može posjedovati bilo koji znak: slovo, brojka, emotikondok kod C++ to nije slučaj.

Swift omogućuje da se napiše veći broj varijabla u jednome retku koda odvojenih samo zarezom s jednom oznakom tipa varijable na kraju retka odvojenih : (dvotočkom) te to izgleda ovako: var red, green, blue: Double[8]



Slika 21. Primjer varijable u C++ (vlastita izrada)

[8] SWIFT, The Swfift programming language , Swift 5.1, dostupno na: https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html pristupljeno (23.06.2019)

Untitled1.cpp

```
1   #include<iostream>
2   using namespace std;
3   main()
4   {
5       int broj_ucenika;
6   broj_ucenika = 30;
7   cout << broj_ucenika<<endl;
8
9   float prosjecna_dob;
10  prosjecna_dob = 11.6;
11  cout<< prosjecna_dob;
12
13  return 0;
14  }
15
```

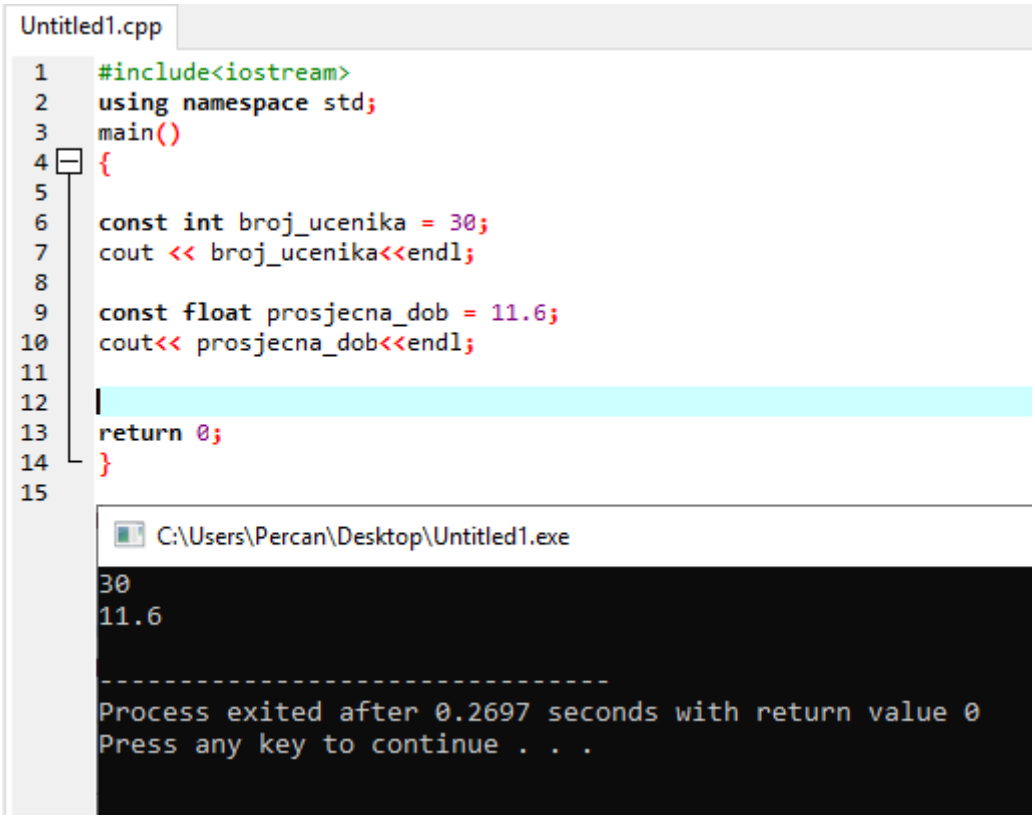
```
C:\Users\Percan\Desktop\Untitled1.exe
30
11.6
--------------------------------
Process exited after 0.2835 seconds with return value 0
Press any key to continue . . .
```

Slika 22. Prikaz C++ varijable s endl; u kodu (vlastita izrada)

U prethodnim primjerima prikazana su dva gotovo ista koda s jednom razlikom u 7. liniji koda. Pisanje Swift koda prikazuje mnogo jednostavniji način naspram C++ u ovome segmentu. Naime Swift svaku liniju koda koju korisnik napiše prikazuje na ekranu kao rezultat na onaj način na koji je korisnik i napisao kod. U konkretnim primjerima usporedbe Swifta i C++, bez uporabe naredbe *end*l; kao kratice za kraj retka C++ ispisuju se kao rezultat dva broja u istome retku što je budućeg korisnika mogao dovesti u zabludu radi li se tu o broju 3011.6 ili o dva broja 30 i 11.6.

4.4. Konstante

Konstante korištene u oba jezika imaju jednu vrijednost od početka do kraja, točnije nepromjenjive su. Konstante se u ovim programskim jezicima deklariraju s različitom riječju, kod C++ to je riječ const dok je kod *Swifta* to let.



23. Prikaz C++ konstante (vlastita izrada)

U ovim se primjerima prikazuje korištenje konstanti te njihova implementacija. Budući da broj učenika i prosječna dob u ovim slučajevima nije promjenjiva, korištene su konstante. Slika koja slijedi u nastavku prikazuje pokušaj promjene vrijednosti konstante u kodu, te program prilikom te promjene izbacuje grešku jer njezina vrijednost tijekom jednog koda nije promjenjiva, odnosno ne može odstupati od one zadane početne vrijednosti.

Slika 24. C++ konstanta i pokušaj promjene vrijednosti (vlastita izrada)

4.5. If, if else, else

If, if else i *else* gotovo jednako se koriste u oba programska jezika te im je zadaća ispitivanje uvjeta, odnosno hoće li se neka linija kod izvršiti ovisno o tome kakav je uvjet postavljen. Predstavljeni primjer ispisuje na ekran Broj je 0!, što je rezultat 6. linije koda gdje je konstanti broj zadana vrijednost 0.

Untitled1.cpp

```
1   #include<iostream>
2   using namespace std;
3   main()
4   {
5
6   const int broj = 0;
7
8   if ( broj > 0 )
9   {
10      cout<<"Broj je pozitivan!"<<endl;
11
12  }
13  else if  ( broj < 0 )
14  {
15      cout<<"Broj je negativan!"<<endl;
16  }
17
18  else
19  {
20      cout<<"Broj je 0!"<<endl;
21  }
22  return 0;
23  }
24
```

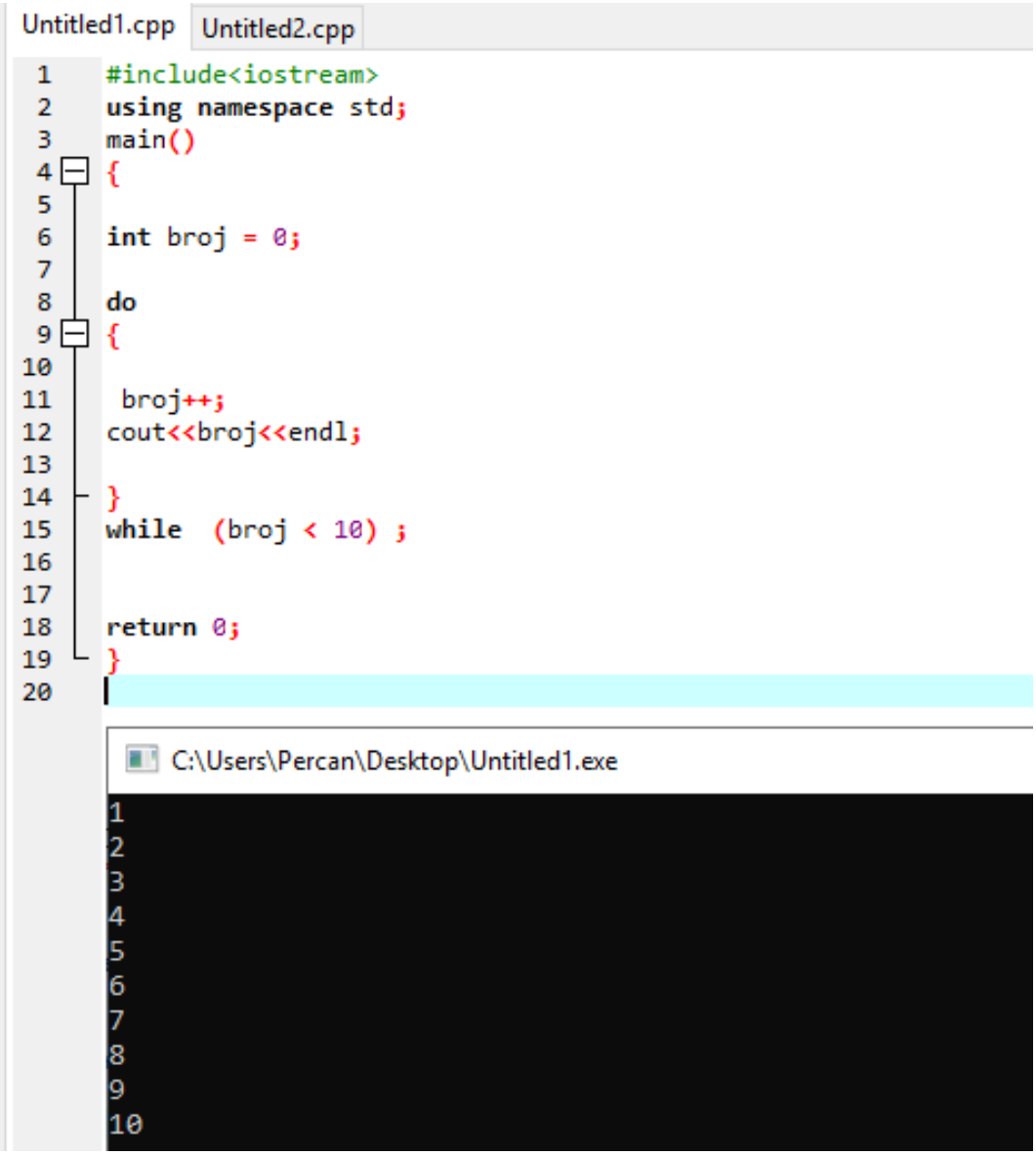
```
C:\Users\Percan\Desktop\Untitled1.exe
Broj je 0!

--------------------------------
Process exited after 0.2864 seconds with return value 0
Press any key to continue . . .
```

Slika 25. If, else if, else naredba u C++ (vlastita izrada)

4.6. Petlje

Korištenje programskih petlji karakteristika je svakog programskog jezika kako se ne bi pisao velik broj linija koda već se olakšalo određenim uvjetima. U oba programska jezika koriste se gotovo identične petlje, jedina iznimka su *for* i *for in* petlja te *repeat while* i *do while* petlja. U primjeru koji slijedi prikazana je *do while* petlja u C++ jeziku. Načinjena na način da program ispiše sve brojeve u zadanome rasponu te da program uveća svaki prethodni broj za 1 sve dok

ne dođe do kraja uvjeta. Ispisan je i broj 10 iz razloga što je program uvećavao svaki prethodni broji za 1 sve dok nije dosegnuo broj koji je jednak broju u zadanome uvjetu. Slika 11. *Repeat while* petlja gotovo je ista kao i primjer koji slijedi te prikazuje *do while* petlju.



Slika 26. Do while petlja u C++ (vlastita izrada)

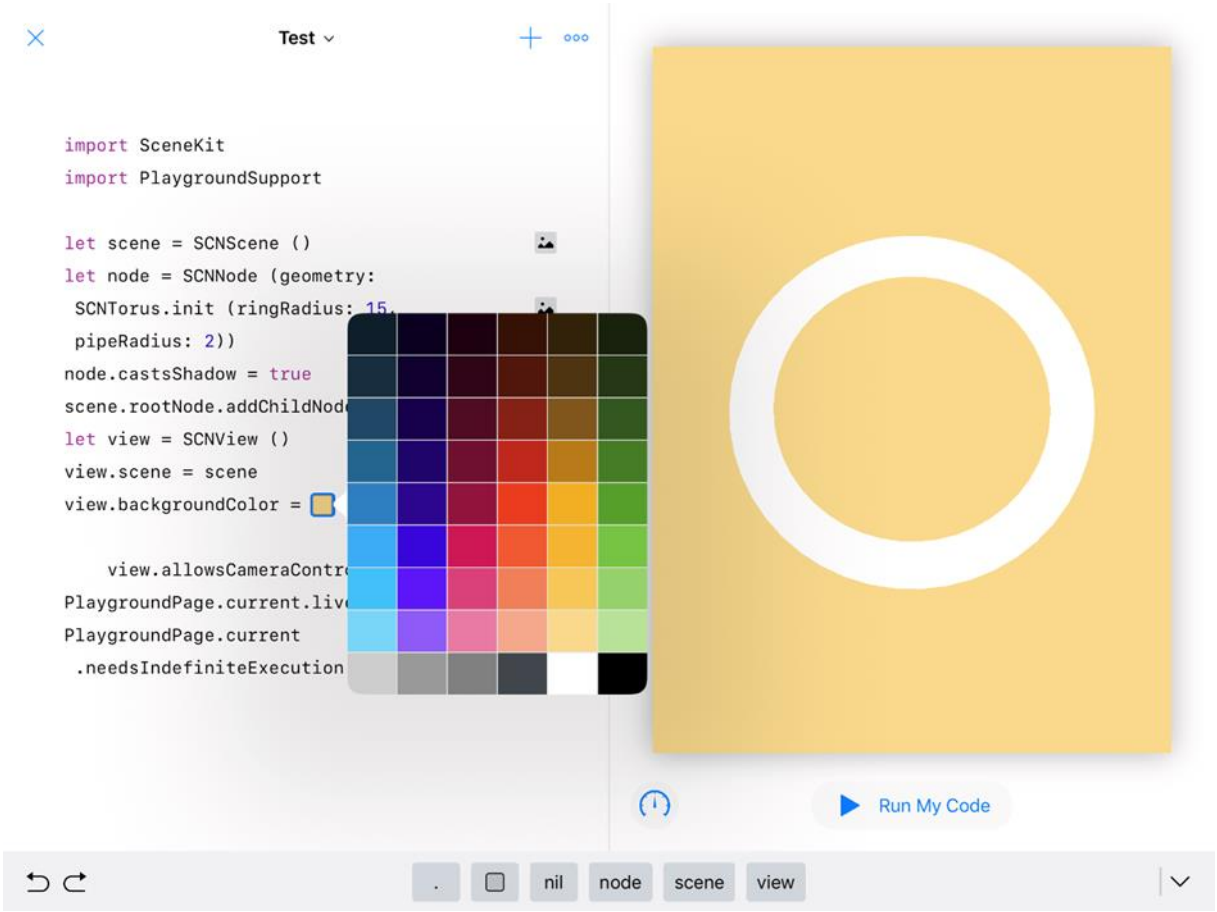
4.7. Primjer usporedbe

Untitled1.cpp

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long pretvorba(int);
int main()
{
    int broj1, broj2, srednji_broj, binarni_broj;

    cout << "Upisi prvi broj: ";
    cin >> broj1;
    cout<<"upisi drugi broj: ";
    cin>> broj2;
    srednji_broj = (broj1 + broj2) / 2;

    cout<<"Srednji broj iznosi: "<< srednji_broj <<endl;

    binarni_broj = pretvorba(srednji_broj);
    cout<<"BINARNI BROJ IZNOSI: "<<binarni_broj<<endl;

    return 0;
}
long long pretvorba(int srednji_broj)
{

    long long binarni_broj = 0;
    int zapamti, i = 1;
    while (srednji_broj!=0)
    {
        zapamti = srednji_broj%2;

        srednji_broj /= 2;
        binarni_broj += zapamti*i;
        i *= 10;
    }
    return binarni_broj;

}
```

```
C:\Users\Percan\Desktop\ZavrÜni\Untitled1.exe
Upisi prvi broj: 10
upisi drugi broj: 40
Srednji broj iznosi: 25
BINARNI BROJ IZNOSI: 11001

--------------------------------
Process exited after 4.407 seconds with return value 0
```

Slika 27. Program aritmetičke sredine i binarnoga zapisa (vlastita izrada)



Slika 28. Pokretljiva kružnica (vlastita izrada)

Primjeri prikazuju razliku programskih jezika C++ i Swift. Slike 28. i 29. prikazuju razliku među kodovima pa je temeljem tih primjera moguće vidjeti koji je programski jezik korisniji za određenu vrstu sadržaja.

Programski jezik Swift pruža mnogo više u razvoju mobilnih aplikacija, 3D dizajniranju geometrijskih likova, izradi Dugmadi (engl. Button) za aplikacije, prikladan je i za izradu animacije, a jednostavnim naredbama nudi vrlo zanimljive mogućnosti, npr. odabir boje pozadine.

C++ kod, prikazan na slici 28., prikladniji je za logičke operacije kao i za izradu programa koji će temeljem upisanih brojeva pronaći njihovu aritmetičku sredinu te za taj aritmetički broj ispisati binarni kod. Iako sintaksa koda C++-a ima više linija, kod djeluje mnogo urednije i jednostavnije, što posljedično rezultira manjim brojem mogućnosti.

5. Zaključak

U mnoštvu poznatih programskih jezika kojima je moguće programirati, danas sve raširenije mobilne aplikacije, obrađen je jezik *Swift* na temu „Učenje programiranja uz pomoć *Swift playgrounds* aplikacije“. *Swift playground* aplikacija nudi mogućnost jednostavnijeg i oku ugodnijeg načina učenja programiranja. Velika većina ljudi danas određene stvari želi steći na jednostavan i brz način, a *Swift,* poznavajući potrebe i želje ljudi, jedini nudi mogućnost učenja programiranja preuzimanjem samo jedne aplikacije. Jedna aplikacija ujedinjuje mnoštvo sadržaja, a osnovne su pisanje koda, upravljanje raznim modelima (opisani u ovom završnom radu), korisnički priručnik i zabavni način učenja.

Apple nažalost, kao i mnogo puta do sada, svoje znanje i svoj kvalitetno razvijeni sustav ograničava. Iako je ponuđena vrhunska aplikacija i svima potpuno besplatno dostupan vrhunski programski jezik, za njezino je korištenje potrebno u najmanju ruku biti njihov obožavatelj, te posjedovati minimalno *iPad* ili *Mac* uređaj. Mnogim programerima ili osobama koja se žele početi baviti programiranjem cijena uređaja je velika, a ponekad i nepremostiva prepreka što onemogućava aplikaciji *Swift* da se probije na sam vrh najtraženijih i najčešće korištenih programskih jezika.

Premda je aplikacija *Swift playgrounds* vrlo lijepo osmišljena i vrlo lijepo stilizirana, prijateljski (engl. *user friendly)* nastrojena, postavlja se pitanje kome je ona zapravo namijenjena? Sam doticaj s aplikacijom, a potom promišljanje o njezinim karakteristikama i mogućnostima, dovodi u pitanje jednostavnost koja se nameće kao prvi dojam o aplikaciji. Preuzeta aplikacija na korisnika djeluje kao vrlo jednostavna igra gdje se gotovim kombinacijama riječi navodi lik po imenu „Byte“ kroz određene staze. Budući da je aplikacija namijenjena edukaciji djece od najranije dobi, vrlo je logično očekivati da će prve staze biti vrlo jednostavne, a zahtjevnost samoga koda će postupno rasti.

Korištenjem aplikacije može se zaključiti da aplikacija ipak pristupačna djeci predškolske dobi, jer dok ne usvoje vještine čitanje i pisanje djeca ne mogu savladati obrazloženja koja nudi aplikacija. Engleski, talijanski, njemački, francuski i ostali jezici na kojima je moguće koristiti aplikaciju jednako tako ograničavaju njezinu uporabu korisnicima koji nisu savladali neki od navedenih jezika ili im nisu materinji. Korisnici odrasle dobi također će se susretati s problemima i s tematikom same aplikacije. Vrlo vjerojatno im upravljanje određenim likom neće biti presudan detalj radi kojeg bi koristili aplikaciju.

Postavlja se pitanje hoće li se netko tko je odrastao u nekom drugom vremenu bez računala i mobitela u svakodnevnom životu, i bez ikakvog predznanja o programiranju, moći okušati u

tome i savladati određena znanja. Na to pitanje nema argumentiranog odgovora, no pretpostavka je da neće.

Nakon svega navedenog, može se zaključiti da je aplikacija primarno namijenjena osobama koja pohađaju minimalno osnovnu školu te se dobro služe barem jednim od ponuđenih stranih jezika. Do sličnog zaključka može se doći već i nakon kraćeg korištenja aplikacije. Korisnik ove aplikacije u određenim segmentima ne uspijeva razumjeti sve što se od njega traži čak i ako poznaje određene strane jezike i ako je upoznat s programskim jezicima kao što su C, C++ i Java.

Razradom *Swift playgrounds* kompatibilnih robota i dronova nameće se novo pitanje, mogu li se ti isti roboti i dronovi spojiti na neki drugi uređaj koji nije načinjen na *iOs* platformi, odnosno je li ga moguće povezati s android uređajima? *Lego, Sphero i Parrot* razvili su i vlastite aplikacije u kojima je moguće na gotovo isti način upravljati robotima i dronovima, i to bez ograničavanja korisnika na korištenje određenih uređaja.

Uzevši u obzir sve napisano govori se o vrlo kvalitetnoj aplikaciji, ali bez dobre definicije kome je namijenjena. Ekspanziju aplikacije sprječava Apple-ove zaštićenost prema drugim uređajima sličnih operativnih sustava. Aplikacija je namijenjena edukaciji, no istovremeno je vrlo zahtjevna - i očekuje korisnika s vrlo dobrim poznavanjem *Swift* programskog jezika kako bi korisniku uspjela protumačiti sve pojedinosti.

Popis literature

Knjige:

1. Apple Inc. (2017), The Swift Programming Language, Dostupno na iBooks, (pristupljeno 12.svibnja 2019.)
2. Jesse Feiler (2017), Exploring *Swift Playgrounds*

Internet izvori:

1. Tutorialspoint, dostupno na: https://www.tutorialspoint.com/swift/swift_variables.htm, (pristupljeno: 21.5.2019.)
2. Apple inc, dostupno na: https://www.apple.com/cz/newsroom/2017/06/swift-playgrounds-expands-coding-education-to-robots-drones-and-musical-instruments/, (pristupljeno: 26.5.2019.)
3. SWIFT, The swift programming language, Swift 5.1, dostupno na: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/, (pristupljeno: 21.5.2019.)
4. Macworld, dostupno na: https://www.macworld.co.uk/how-to/iosapps/how-use-swift-playgrounds-learn-code-with-apples-swift-training-app-3642835/, (pristupljeno: 29.5.2019.)
5. GitHub, dostupno na: https://github.com/uraimo/Awesome-Swift-Playgrounds, (pristupljeno: 1.6.2019.)
6. Apple Inc, dostupno na: https://www.apple.com/ca/education/docs/Swift_Playgrounds_Resource_CAEN_Mar2017.pdf, (pristupljeno: 1.6.2019.)
7. Apple Inc: dostupno na: http://images.apple.com/education/docs/hourofcode_guide.pdf, (pristupljeno: 7.6.2019.)
8. Apple Inc,Swift, dostupno na: https://swift.org, (pristupljeno: 11.6.2019.)
9. W3Schools, dostupno na: https://www.w3schools.in/cplusplus-tutorial/keywords/ (pristupljeno: 12.07.2019.)
10. Generation robots, dostupno na: https://www.generationrobots.com/en/401485-lego-mindstorms-ev3-intelligent-brick.html (pristupljeno: 20.06.2019.)

Popis slika i tablica
Popis slika

Slika 1.Dodjeljivanje vrijednosti (vlastita izrada)

Slika 2.Deklaracija varijable (vlastita izrada)

Slika 3.Određivanje tipa varijabli (vlastita izrada)

Slika 4.Deklaracija konstanti (vlastita izrada)

Slika 5. Konstante

Slika 6. Jednolinijski komentar (vlastita izrada)

Slika 7. Više linijski komentar (vlastita izrada)

Slika 8. Komentar unutar komentara (vlastita izrada)

Slika 9. *For-in* petlja (vlastita izrada)

Slika 10. *While* petlja (vlastita izrada)

Slika 11. *Repeat while petlja* (vlastita izrada)

Slika 12. Strukture i klase (Apple inc. 367 str)

Slika 13. Strukture i klase (vlastita izrada)

Slika 14. *Xcode playground – Swift playground app* (vlastita izrada)

Slika 15. *Learn to code* 1 (vlastita izrada)

Slika 16. *Learn to code 2* (vlastita izrada)

Slika 17. Osobni podaci (vlastita izrada)

Slika 18. *Sphero bolt* (vlastita izrada)

Slika 19. Tehničke karakteristike (https://www.generationrobots.com/en/401485-lego-mindstorms-ev3-intelligent-brick.html, pristupljeno: 22.6.2019.)

Slika 20. C++ ključne riječi

(W3Schools, dostupno na: https://www.w3schools.in/cplusplus-tutorial/keywords/, pristupljeno 12.07.2019)

Slika 21. Primjeri varijable C++ (vlastita izrada)

Slika 22. Prikaz varijable s *endl*; u kodu (vlastita izrada)

Slika 23. Prikaz C++ konstante (vlastita izrada)

Slika 24. C++ konstanta i pokušaj promjene vrijednosti (vlastita izrada)

Slika 25. *If, if else, else* naredba u C++ (vlastita izrada)

Slika 26. *Do while* petlja u C++ (vlastita izrada)

Slika 27. Program aritmetičke sredine i binarnoga zapisa (vlastita izrada)

Slika 28. Pokretljiva kružnica (vlastita izrada)

Popis tablica

Tablica 1. Aritmetički operatori

Tablica 2. Operatori dodjeljivanja

Tablica 3. Logički operatori

Tablica 4. Operatori usporedbe

6. Sažetak

U ovom je završnom radu obrađena tema „Učenje programiranja uz pomoć aplikacije *Swift playgrounds*“. Nekadašnji uvriježeni načini učenja počeli su se zamjenjivati novijim, vizualnijim, jednostavnijim i u konačnici zanimljivijim. . Kako je u današnje vrijeme važno da je sve nadohvat ruke, tako je tvrtka Apple razvila aplikaciju za učenje programiranja koju je moguće preuzeti na Ipad uređaje. Cilj aplikacije je razviti svijest ljudi i potaknuti ih na logičko razmišljanje te omogućiti daljnju informatizaciju društva. U radu se obrađuje *Swift* programski jezik na kojem se zasniva cijela aplikacija te jednostavni kodovi. Cijeli rad je koncipiran na način priručnika te potkrijepljen slikama koje se mogu koristiti za upoznavanje *Swifta* i *Swift playgrounds* aplikacije.

Ključne riječi: Programiranje, učenje, *Swift, Swift playgrounds*

7. Summary

In this final paper, the topic of "Learning programming using the Swift playgrounds aplication" was discussed. Former modes of learning have begun to be replaced with newer, more visual, simpler and ultimately more interesting ways. Since today it is very important that everything is within reach, Apple has developed an application that can be downloaded on Ipad devices and learning programming. The aim of the application is to develop awareness and to encourage people to logical thinking and to inform the society themselves. Throughout the work, the Swift programming language is based on which the entire application is based and familiar with simple codes. The whole work is conceived based on a manual that is substantiated by pictures that the user can use to overcome and meet Swift and Swift playgrounds app.

Key words: Programming, Learning, Swift, Swift playgrounds