



FAKULTET INŽENJERSKIH NAUKA UNIVERZITET U KRAGUJEVCU

Tema: Programiranje aplikacija IOS sistema
Student: Nevena Stašić
Profesor: Nenad Grujović

Kragujevac, 2019. godine

Sadržaj

- ◉ String
- ◉ Empty String
- ◉ Interpolacija (umetanje) Stringa
- ◉ Spajanje Stringa
- ◉ Dužina Stringa
- ◉ Upoređivanje Stringova
- ◉ Iteracija (razdvajanje) karaktera
- ◉ Metode vezane za Stringove
- ◉ Karakteri
- ◉ Spajanje stringa i karaktera
- ◉ Nizovi
- ◉ Pristupanje nizovima
- ◉ Modifikovanje nizova
- ◉ Sabiranje dva niza
- ◉ Rečnici
- ◉ Pristupanje rečnicima

String

- Stringovi u Swift 4 programskom jeziku su uređena kolekcija znakova, kao što su “Hello World” i predstavljeni su u Swiftu tipom String, koji zauzvrat predstavlja skup vrednosti tipa Charactera.
- Sintaksa kreiranja različitih vrsta stringova:

```
// String creation using String literal
var stringA = "Hello, Swift 4!"
print( stringA )

// String creation using String instance
var stringB = String("Hello, Swift 4!")
print( stringB )

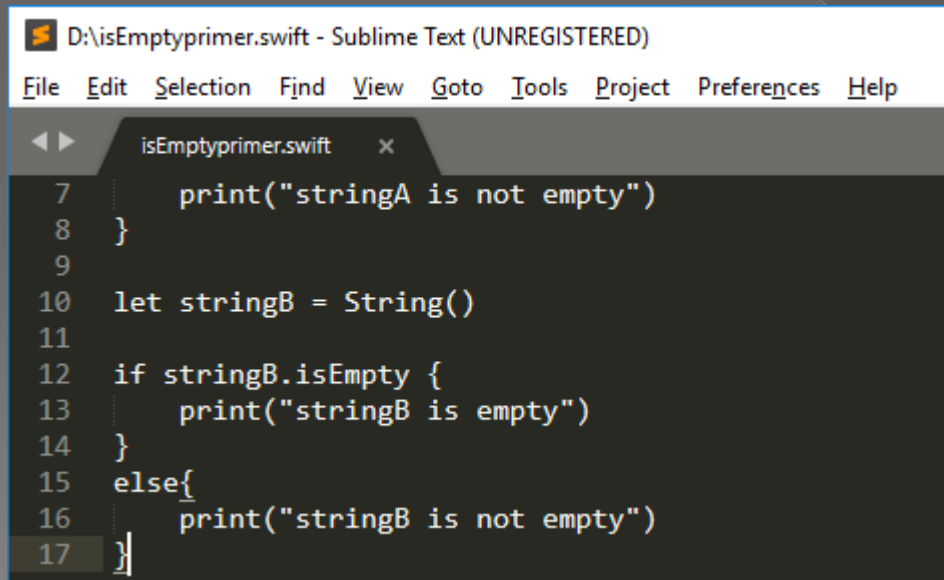
//Multiple line string

let stringC = """
Hey this is a
example of multiple Line
string by tutorialsPoint
"""
print(stringC)
```

Slika 1 – Sintaksa kreiranja stringova

Empty String

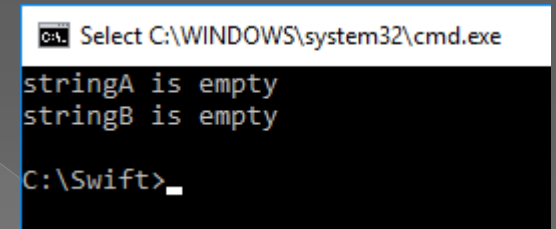
- Možemo da kreiramo prazan string ili korišćenjem praznog stringa ili kreiranjem instance String klase kao što je prikazano u primeru ispod. Takođe, možemo proveriti da li je String prazan ili ne koristeći komandu isEmpty.



```
D:\isEmptyprimer.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

isEmptyprimer.swift x
7   print("stringA is not empty")
8   }
9
10  let stringB = String()
11
12  if stringB.isEmpty {
13      print("stringB is empty")
14  }
15  else{
16      print("stringB is not empty")
17  }
```

Slika 2 – Primeri Empty String

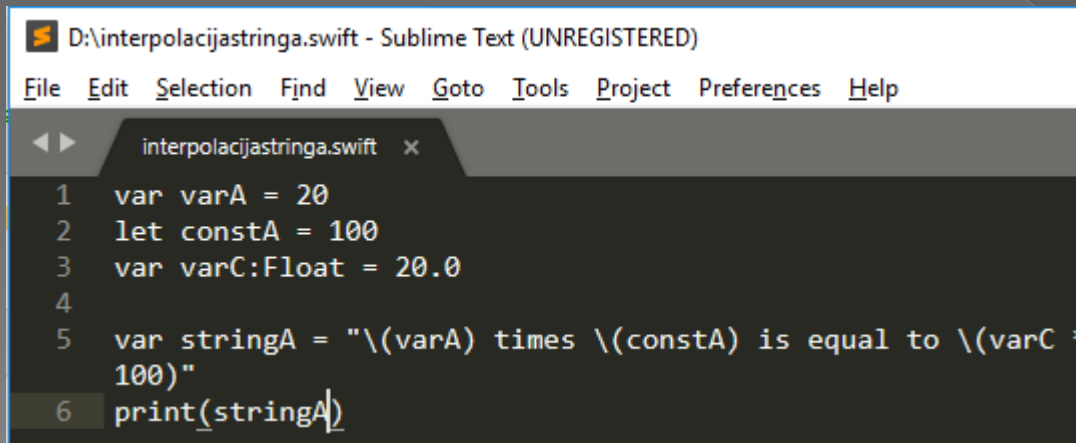


```
C:\> Select C:\WINDOWS\system32\cmd.exe
stringA is empty
stringB is empty
C:\Swift>_
```

Slika 3 – Startovan program

Interpolacija(umetanje) Stringa

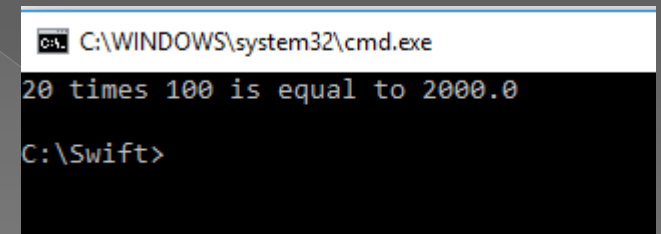
- Interpolacija stringa je način da se konstruiše nova vrednost Stringa iz mešavine konstanti, promenljivih, literala i izraza tako što će se njihove vrednosti uključiti u string.
- Svaka stavka(promenljiva ili konstanta) koju unesete u string je omotana u par zagrada i prefiksom obrnute crte. Na slici ispod je prikazan jednostavan primer umetanja promenljivih i konstanti u string.



```
D:\interpolacijastringa.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

interpolacijastringa.swift x
1 var varA = 20
2 let constA = 100
3 var varC:Float = 20.0
4
5 var stringA = "\(varA) times \(constA) is equal to \(varC *
6 print(stringA)
```

Slika 4 – Interpolacija Stringa



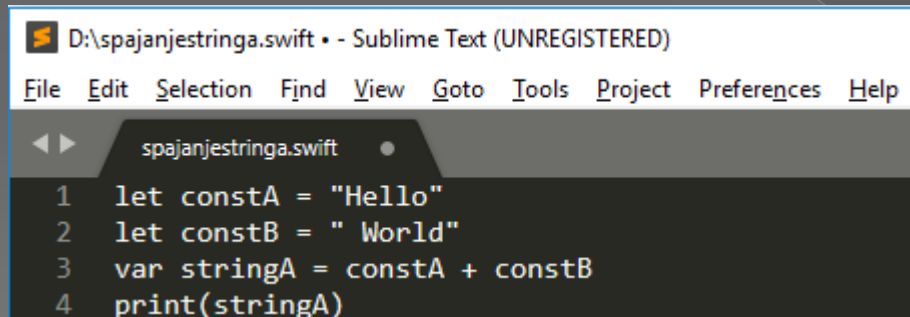
```
C:\WINDOWS\system32\cmd.exe
20 times 100 is equal to 2000.0

C:\Swift>
```

Slika 5 – Startovanje programa

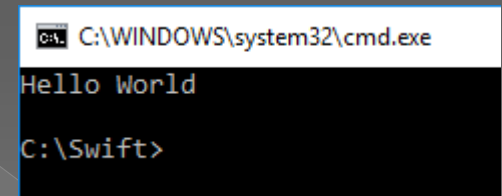
Spajanje stringa

- Možemo koristiti operator + da spojimo dva stringa ili string i karakter, ili dva karaktera. Evo jednostavnog primera:



```
D:\spajanjestringa.swift • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
spajanjestringa.swift
1 let constA = "Hello"
2 let constB = " World"
3 var stringA = constA + constB
4 print(stringA)
```

Slika 6 – Spajanje stringa

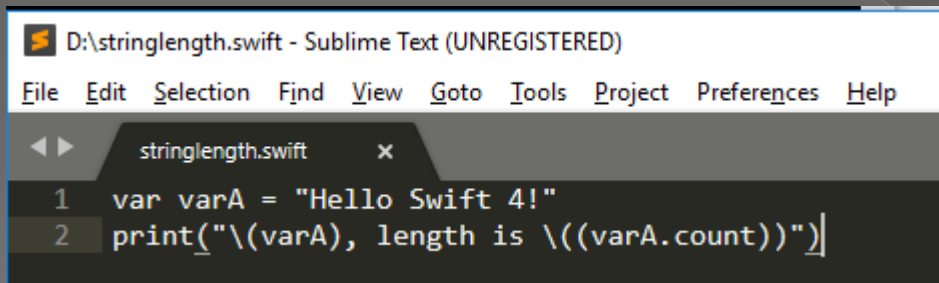


```
C:\WINDOWS\system32\cmd.exe
Hello World
C:\Swift>
```

Slika 7 – Startovan program

Dužina stringa

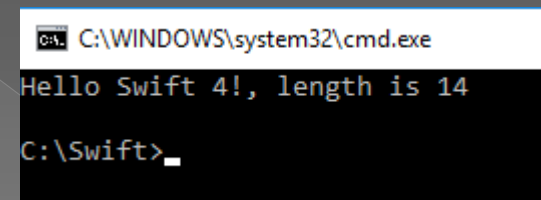
- Swift 4 stringovi nemaju operaciju `length()`, ali možemo koristiti globalnu `count()` funkciju za brojanje znakova u nizu. Jednostavan primer za to je prikazan na slici ispod:



```
D:\stringlength.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

stringlength.swift x
1 var varA = "Hello Swift 4!"
2 print("\(varA), length is \(varA.count)")
```

Slika 8 – Primer za dužinu stringa

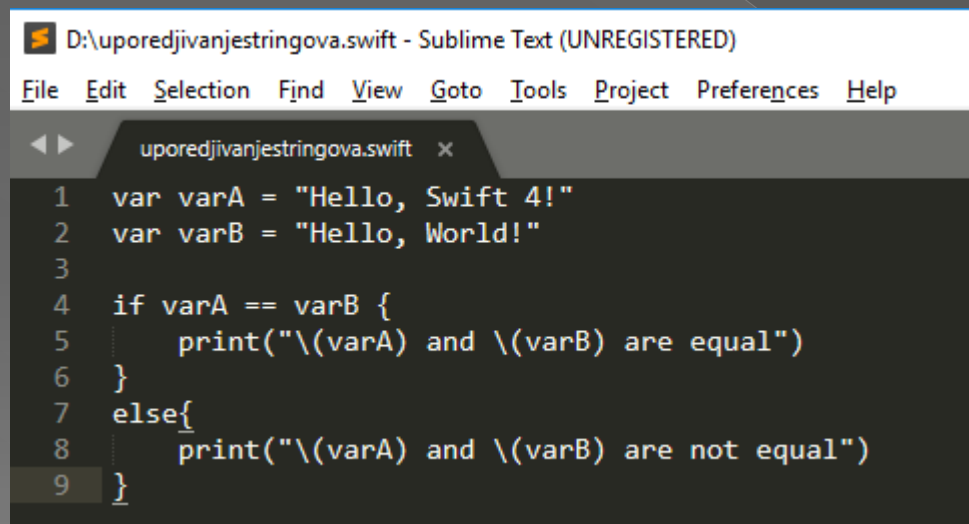


```
C:\WINDOWS\system32\cmd.exe
Hello Swift 4!, length is 14
C:\Swift>
```

Slika 9 – Startovan program

Upoređivanje stringova

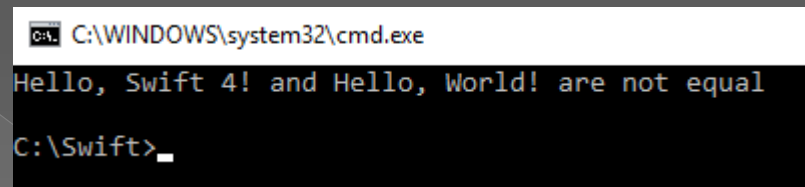
- Možemo koristiti operator `==` da bismo uporedili dve promenljive ili konstante. Primer za ovo se nalazi na slici ispod:



```
D:\uporedjivanjestringova.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

uporedjivanjestringova.swift x
1 var varA = "Hello, Swift 4!"
2 var varB = "Hello, World!"
3
4 if varA == varB {
5     print("\(varA) and \(varB) are equal")
6 }
7 else{
8     print("\(varA) and \(varB) are not equal")
9 }
```

Slika 10 – Primer za upoređivanje stringova

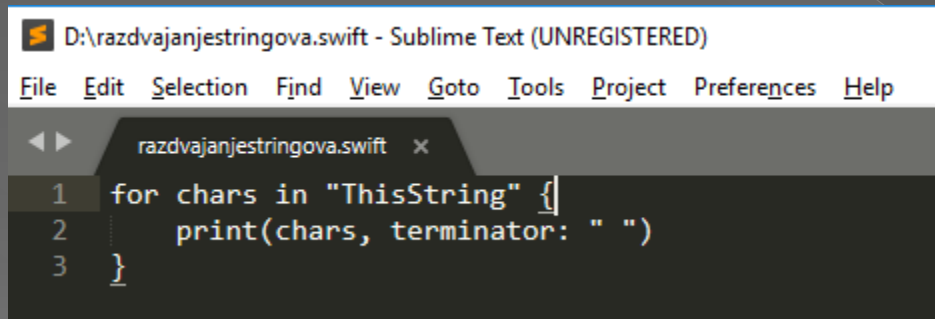


```
C:\WINDOWS\system32\cmd.exe
Hello, Swift 4! and Hello, World! are not equal
C:\Swift>
```

Slika 11 – Startovan program

Iteracija(razdvajanje) karaktera u stringu

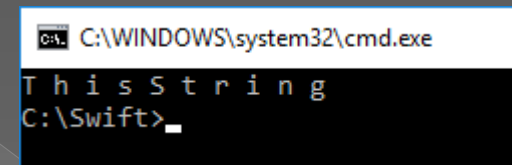
- Na slici ispod je prikazan primer gde je slovo po slovo iz stringa razdvojeno praznim mestom.



```
D:\razdvajanjestringova.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

razdvajanjestringova.swift x
1 for chars in "ThisString" {
2     print(chars, terminator: " ")
3 }
```

Slika 12 – Primer razdvajanja karaktera u stringu



```
C:\WINDOWS\system32\cmd.exe
T h i s S t r i n g
C:\Swift>
```

Slika 13 – Startovan program

Metode vezane za stringove

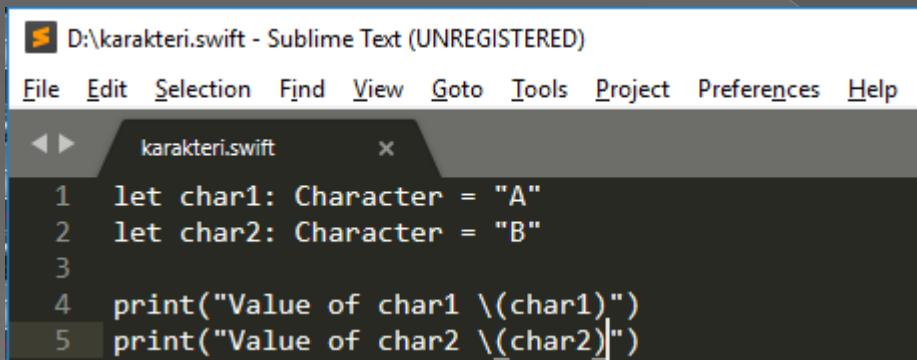
- Swift 4 podržava širok spektar metoda i operatora vezanih za Stringove. Ispod će biti prikazani neki od njih.

Sr.No	Functions/Operators & Purpose
1	isEmpty A Boolean value that determines whether a string is empty or not.
2	hasPrefix(prefix: String) Function to check whether a given parameter string exists as a prefix of the string or not.
3	hasSuffix(suffix: String) Function to check whether a given parameter string exists as a suffix of the string or not.
4	toInt() Function to convert numeric String value into Integer.
5	count() Global function to count the number of Characters in a string.
6	utf8 Property to return a UTF-8 representation of a string.

7	utf16 Property to return a UTF-16 representation of a string.
8	unicodeScalars Property to return a Unicode Scalar representation of a string.
9	+ Operator to concatenate two strings, or a string and a character, or two characters.
10	+= Operator to append a string or character to an existing string.
11	== Operator to determine the equality of two strings.
12	< Operator to perform a lexicographical comparison to determine whether one string evaluates as less than another.
13	startIndex To get the value at starting index of string.

Karakteri

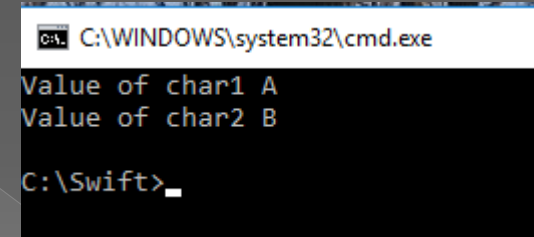
- Karakter u Swiftu 4 je pojedinačni karakter Stringa. U nastavku je prikazana sintaksa definisanja karaktera.



The screenshot shows a Sublime Text editor window titled "D:\karakter.swift - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The editor displays the following Swift code in a file named "karakter.swift":

```
1 let char1: Character = "A"
2 let char2: Character = "B"
3
4 print("Value of char1 \(char1)")
5 print("Value of char2 \(char2)")
```

Slika 16 – Sintaksa definisanja char-a



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The output of the Swift program is displayed as follows:

```
Value of char1 A
Value of char2 B

C:\Swift>
```

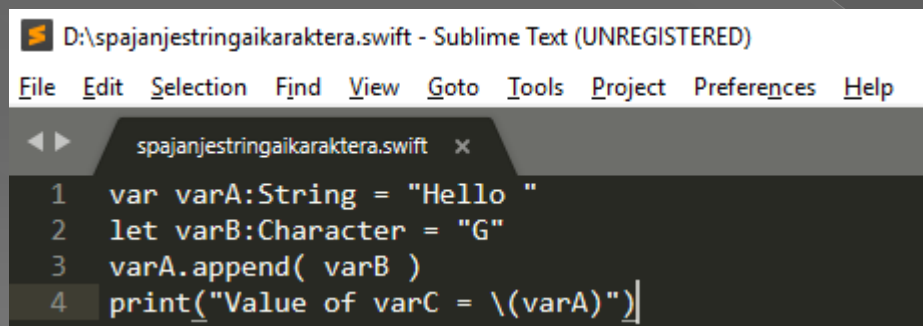
Slika 17 – Startovan program

Karakteristi

- ⦿ Ako pokušamo da sačuvamo više od jednog znaka u promenljivoj tipa char ili konstanti tipa char, Swift 4 to neće dozvoliti!
- ⦿ Takođe, nije moguće kreirati praznu promenljivu char ili konstantu tipa char koja će imati praznu vrednost!

Spajanje stringa i karaktera

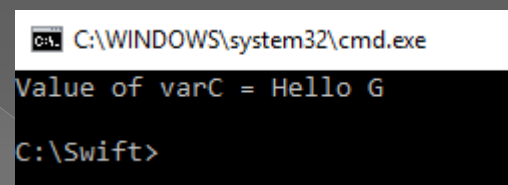
- Sledeći primer pokazuje kako se Swift 4 karakter može spojiti sa Swift 4 stringom:



```
D:\spajanjestringaikaraktera.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

spajanjestringaikaraktera.swift x
1 var varA:String = "Hello "
2 let varB:Character = "G"
3 varA.append( varB )
4 print("Value of varC = \(varA)")
```

Slika 18 – Primer spajanja stringa i karaktera



```
C:\WINDOWS\system32\cmd.exe
Value of varC = Hello G

C:\Swift>
```

Slika 19 – Startovan program

Nizovi

- Swift 4 nizovi se koriste za skladištenje poređanih lista vrednosti istog tipa. Swift 4 stavlja strogu proveru koja nam ne dozvoljava da unesemo pogrešan tip u polje, čak ni greškom.
- Ako dodelite kreirani niz promenljivoj, ona je uvek promenljiva, što znači da je možete promeniti dodavanjem, uklanjanjem ili menjanjem članova, ali ako dodelite niz konstanti, taj niz je nepromenljiv i njegova veličina i sadržaj se ne mogu menjati.
- Možemo kreirati prazan niz određenog tipa, koristeći sledeću sintaksu:

```
var someArray = [SomeType]()
```

Slika 20 – Kreiranje praznog niza

Nizovi

Sintaksa za kreiranje niza date veličine a i inicijalizacija vrednosti je prikazana na slici ispod:

```
var someArray = [SomeType](count: NumbeOfElements, repeatedValue: InitialValue)
```

Slika 21 – Sintaksa za kreiranje niza

Možemo koristiti sledeći izraz da bismo kreirali prazan niz tipa Int, koji ima 3 elementa i početnu vrednost kao nulu:

```
var someInts = [Int](count: 3, repeatedValue: 0)
```

Slika 22 – Sintaksa za kreiranje niza određene veličine

Sledi još jedan primer za kreiranje niza od tri elementa sa određenim vrednostima:

```
var someInts:[Int] = [10, 20, 30]
```

Slika 23 – Kreiranje niza sa određenim elementima

Pristupanje nizovima

- Možemo da izvučemo vrednost iz niza koristeći indeks, prosleđujući indek vrednosti koju želimo da preuzmemo u uglastim zagradama odmah nakon imena niza na sledeći način:

```
var someVar = someArray[index]
```

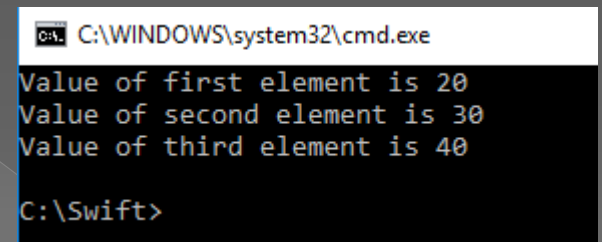
Slika 24 – Sintaksa pristupanju niza

Modifikovanje nizova

Možemo koristiti `append()` metodu ili operator dodavanja (`+=`) da bismo dodali novu stavku na kraju niza. Pogledajmo sledeći primer. Ovde, u početku, kreiramo prazan niz i zatim dodamo nove elemente u isti niz.

```
1 var someInts = [Int]()
2
3 someInts.append(20)
4 someInts.append(30)
5 someInts += [40]
6
7 var someVar = someInts[0]
8
9 print( "Value of first element is \(someVar)" )
10 print( "Value of second element is \(someInts[1])" )
11 print( "Value of third element is \(someInts[2])" )
```

Slika 25 – Primer modifikovanja niza

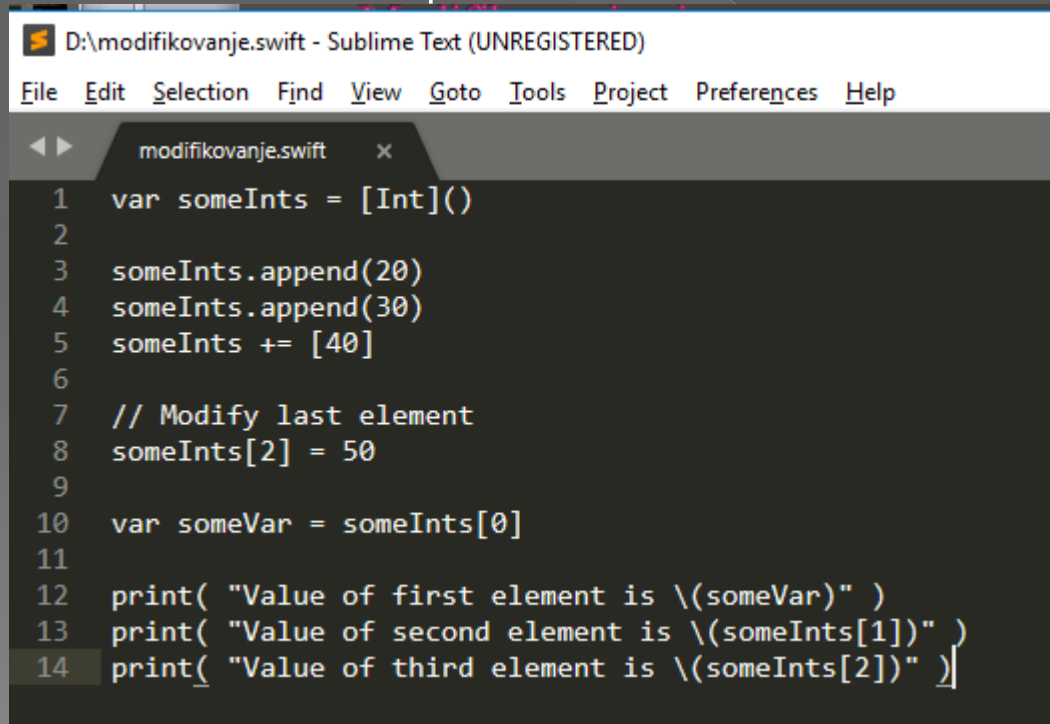
A screenshot of a terminal window titled "C:\WINDOWS\system32\cmd.exe". The window shows the output of a Swift program: "Value of first element is 20", "Value of second element is 30", and "Value of third element is 40". The prompt "C:\Swift>" is visible at the bottom.

```
C:\WINDOWS\system32\cmd.exe
Value of first element is 20
Value of second element is 30
Value of third element is 40
C:\Swift>
```

Slika 26 – Startovan program

Modifikovanje nizova

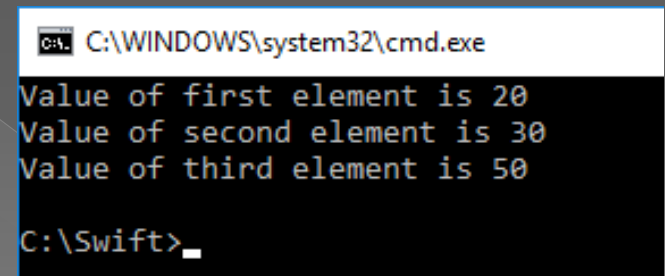
- Postojeći element niza možemo modifikovati dodeljivanjem nove vrednosti na datom indeksu, kao što je prikazano u sledećem primeru:



```
D:\modifikovanje.swift - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

modifikovanje.swift x
1 var someInts = [Int]()
2
3 someInts.append(20)
4 someInts.append(30)
5 someInts += [40]
6
7 // Modify last element
8 someInts[2] = 50
9
10 var someVar = someInts[0]
11
12 print( "Value of first element is \(someVar)" )
13 print( "Value of second element is \(someInts[1])" )
14 print( "Value of third element is \(someInts[2])" )
```

Slika 27 – Menjanje postojećeg elementa



```
C:\WINDOWS\system32\cmd.exe
Value of first element is 20
Value of second element is 30
Value of third element is 50
C:\Swift>
```

Slika 28 – Startovan program

Sabiranje dva niza

- Da saberemo dva niza istog tipa koji će dati novi niz sa kombinacijom vrednosti iz dva niza, možemo koristiti operator dodavanja (+). Način na koji se to radi je prikazan kroz primer na slici ispod:

```
var intsA = [Int](count:2, repeatedValue: 2)
var intsB = [Int](count:3, repeatedValue: 1)

var intsC = intsA + intsB
for item in intsC {
    print(item)
}
```

Slika 29 – Sintaksa sabiranja dva niza

Rečnici

- Swift 4 rečnici se koriste za skladištenje neuređenih lista vrednosti istog tipa. Swift 4 stavlja strogu proveru koja nam ne dozvoljava da unesemo pogrešan tip u rečniku čak ni greškom.
- Swift 4 rečnici koriste jedinstveni identifikator poznat kao ključ za čuvanje vrednosti koja se kasnije može referencirati i pogledati kroz isti ključ. Za razliku od elemenata niza, elementi u rečniku nemaju određeni redosled. Možemo koristiti rečnik kada je potrebno da potražimo vrednosti na osnovu njihovih ključeva.
- Ključ rečnika može biti ili celi broj ili niz bez ograničenja, ali bi trebao biti jedinstven u rečniku.
- Ako dodelimo kreirani rečnik promenljivoj, ona je uvek promenljiva, što znači da možemo da je promenimo dodavanjem, uklanjanjem ili menjanjem stavki. Ali ako dodelimo rečnik konstanti, taj rečnik je nepromenljiv, a njegova veličina i sadržaj se ne mogu menjati.

Rečnici

Možemo kreirati prazan rečnik određenog tipa, koristeći sledeću sintaksu:

```
var someDict = [KeyType: ValueType]()
```

Slika 30 – Sintaksa kreiranja rečnika

Možemo koristiti sledeću jednostavnu sintaksu da bismo kreirali prazan rečnik čiji će ključ biti tipa Int, a pridružene vrednosti će biti nizovi.

```
var someDict = [Int: String]()
```

Slika 31 – Sintaksa za kreiranje rečnika određenog tipa

Rečnici

- ◉ Evo primera za kreiranje rečnika iz skupa datih vrednosti:

```
var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]
```

Slika 32 – Sintaksa kreiranja rečnika

Pristupanje rečnicima

- Možemo doći do vrednosti iz rečnika, prosleđujući ključ vrednosti koju želimo da preuzmemo u uglastim zagradama odmah nakon imena rečnika na sledeći način:

```
var someVar = someDict[key]
```

Slika 33 – Sintaksa pristupanja rečniku