Programski jezik Objective-C

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

Dalma Beara, Denis Aličić, Mateja Marjanović, Anja Miletić
kontakt email prvog, drugog, trećeg, četvrtog autora

6. april 2019

Sažetak

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da ispoštujete!) kao i par tehničkih pomoćnih uputstava. Pročitajte tekst pažljivo jer on sadrži i važne informacije vezane za zahteve obima i karakteristika seminarskog rada.

Sadržaj

1 Uvod

Kada budete predavali seminarski rad, imenujete datoteke tako da sadrže redni broj teme, temu seminarskog rada, kao i prezimena članova grupe. Precizna uputstva na temu imenovnja će biti data na formi za predaju seminarskog rada. Predaja seminarskih radova biće isključivo preko veb forme, a NE slanjem mejla. Link na formu će biti dat u okviru obaveštenja na strani kursa. Vodite računa da prilikom predavanja seminarskog rada predate samo one fajlove koji su neophodni za ponovno generisanje pdf datoteke. To znači da pomoćne fajlove, kao što su .log, .out, .blg, .toc, .aux i slično, **ne treba predavati**.

2 Osnovna uputstva

Vaš seminarski rad mora da sadrži najmanje jednu **sliku**, najmanje jednu **tabelu** i najmanje **sedam referenci** u spisku literature. Najmanje jedna slika treba da bude originalna i da predstavlja neke podatke koje ste Vi osmislili da treba da prezentujete u svom radu. Isto važi i za najmanje jednu tabelu. Od referenci, neophodno je imati bar jednu **knjigu**, bar jedan **naučni članak** iz odgovarajućeg časopisa i bar jednu adekvatnu **veb adresu**.

Dužina seminarskog rada treba da bude od 10 do 12 strana.

Ко жели, може да пише рад ћирилицом. У том случају, неопходно је да су инсталирани одговарајући пакети: texlive-fonts-extra, texlive-latex-extra, texlive-lang-cyrillic, texlive-lang-other.

Nemojte koristiti stari način pisanja slova, tj ovo:

```
\v{s} i \v{c} i \'c ...
```

Koristite direknto naša slova:

```
š i č i ć ...
```

3 Engleski termini i citiranje

Na svakom mestu u tekstu naglasiti odakle tačno potiču informacije. Uz sve novouvedene termine u zagradi naglasiti od koje engleske reči termin potiče.

Naredni primeri ilustruju način uvođenja enlegskih termina kao i citiranje.

Primer 3.1 *Problem zaustavljanja (eng.* halting problem*) je neodlučiv [3].*

Primer 3.2 *Za prevođenje programa napisanih u programskom jeziku C može se koristiti GCC kompajler [1].*

Primer 3.3 *Da bi se ispitivala ispravost softvera, najpre je potrebno precizno definisati njegovo ponašanje [2].*

Reference koje se koriste u ovom tekstu zadate su u datoteci *seminarski.bib*. Prevođenje u pdf format u Linux okruženju može se uraditi na sledeći način:

```
pdflatex TemaImePrezime.tex
bibtex TemaImePrezime.aux
pdflatex TemaImePrezime.tex
pdflatex TemaImePrezime.tex
```

Prvo latexovanje je neophodno da bi se generisao *.aux* fajl. *bibtex* proizvodi odgovarajući *.bbl* fajl koji se koristi za generisanje literature. Potrebna su dva prolaza (dva puta pdflatex) da bi se reference ubacile u tekst (tj da ne bi ostali znakovi pitanja umesto referenci). Dodavanjem novih referenci potrebno je ponoviti ceo postupak.

Broj naslova i podnaslova je proizvoljan. Neophodni su samo Uvod i Zaključak. Na poglavlja unutar teksta referisati se po potrebi.

Primer 3.4 *U odeljku* **??** *precizirani su osnovni pojmovi, dok su zaključci dati u odeljku 13.*

Još jednom da napomenem da nema razloga da pišete:

`\v{s} i \v{c} i \'c ...`

Možete koristiti srpska slova

`š i č i ć ...`

4 Slike i tabele

Slike i tabele treba da budu u svom okruženju, sa odgovarajućim naslovima, obeležene labelom da koje omogućava referenciranje.

Primer 4.1 *Ovako se ubacuje slika. Obratiti pažnju da je dodato i*

`\usepackage{graphicx}`

Slika 1: Pande

Na svaku sliku neophodno je referisati se negde u tekstu. Na primer, na slici 1 prikazane su pande.

Primer 4.2 *I tabele treba da budu u svom okruženju, i na njih je neophodno referisati se u tekstu. Na primer, u tabeli 1 su prikazana različita poravnanja u tabelama.*

Tabela 1: Razlčita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

5 Kôd i paket listings

Za ubacivanje koda koristite paket **listings**: https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings

Primer ubacivanja koda

```
# This program adds up integers in the command line
import sys
try:
    total = sum(int(arg) for arg in sys.argv[1:])
    print 'sum =', total
```

```
except ValueError:
    print 'Please supply integer arguments'
```

6 Nastanak i istorijski razvoj, mesto u razvojnom stablu, uticaji drugih programskih jezika

6.1 Nastanak i istorijski razvoj

Iako se pojam jezika Objective-C primarno vezuje za proizvode kompanije Epl (engl. Apple) – MAC OS X, iPhone itd, on je zapravo nastao mnogo pre njih. Njegove temelje postavili su Bred Koks i Tom Lav 1981. godine, težeći da nađu način za povećavanje produktivnosti programera. Te godine pojavio se novi, revolucionarni jezik Smalltalk, koji je tada unapredio koncept objektno-orijentisanog programiranja. U osnovi tog jezika bilo je posmatranje programa kao skupova objekata koji su mogli da komuniciraju jedni sa drugima dinamički pozivajući metode. To je omogućilo da se stanje programa menja pod uticajem korisnika. Koks je u ovoj ideji video mogućnost da se vrtoglavo ubrza pisanje programa, pošto su se u njemu mogle praviti biblioteke objekata i posle koristiti u drugim programima bez izmene. Međutim, Smalltalk je bio veoma spor i zahtevao je da se svi programi pišu i pokreću u posebnom okruženju. Tada se javila potreba za spajanjem objektno-orijentisanih ideja Smalltalk-a i brzog jezika C. Koks je 1983. objavio naučni rad u kom je predstavio objektno-orijentisani prekompilator (eng. *Object-Oriented Precompiler*). Da bi ovu svoju ideju izbacili na tržište, njih dvojica su osnovali kompaniju Stepstoun (eng. *Stepstone*), izmenili kompilator za OOPC i preimenovali jezik u Objective-C. Iako je glavna ideja ovog projekta bila pisanje i prodavanje pomenutih biblioteka koje su mogle ponovo da se koriste, korisnici su imali dosta zamerki, te je jezik sve više poprimao karakteristike objektno-orijentisanih jezika kakve ih danas znamo, kao što su sakupljač otpada i interpreter, te je kompanija uskoro propala. Stiv Džobs je 1988. kupio licencu, a nekoliko godina kasnije i sva prava za Objective-C za potrebe svoje kompanije NeXT Computer koja se 1996. pripojila Apple-u. Sa tom tranzicijom, sam jezik je pretrpeo razne promene, a najvažnije su uvođenje dveju novih „komponenti": kategorija (eng. *categories*), danas poznatih kao ekstenzije (eng. *extensions*) i protokola (eng. *protocols*). Kategorije su služile da programerima omoguće da sami dodaju funkcije u objekte iz već postojećih biblioteka, a protokoli su omogućavali objektima da komunuciraju jedni s drugima. Oni su danas podržani u jeziku Java i poznati su nam kao interfejsi (eng. *interfaces*).

6.2 Mesto u razvojnom stablu

6.3 Uticaj drugih programskih jezika

Kao što je već navedeno, Objective-C je nastao kao kombinacija koncepata na kojima počiva Smalltalk utočenih u sintaksu jezika C. Isprva je čak funkcionisao tako što je kod prevođen na C i onda izvršavan. S druge strane, on je uticao na jezik Swift koji je razvio Epl. Swift se često naziva i „Objective-C bez C-a".

7 Osnovna namena programskog jezika, svrha i mogućnosti

Krajem 80-ih godina prošlog veka, popularnost Objective-C jezika je rasla sa razvojem NeXT sistema. Najviše se koristi za razvoj softvera za Apple iOS operativni sistem. Uticaj Objective-C jezika je vidljiv u Java programskom jeziku. Objective-C definiše mali ali moćan skup ekstenzija ANSI C programskog jezika koje omogućavaju objektno orijentisano programiranje. Cocoa, Apple-ov API za macOS je napisan uz pomoć Objective-C jezika, takodje veliki deo aplikacija je napisan koristeći taj jezik.

8 Osnovne osobine ovog programskog jezika, podržane paradigme i koncepti

Objective-C je nadskup C programskog jezika koji pruža objektno orijentisane sposobnosti i dinamičko izvršno okruzenje (eng. *runtime*). Objective-C nasledjuje sintaksu, primitivne tipove i naredbe za tok izvršvanja od jezika C, a dodaje sintaksu za definisanje klasa i metoda. Takodje omogućava dinamičko tipiziranje i povezivanje.

8.1 Osnovni koncepti

8.1.1 Klasteri klasa

Klasteri klasa (eng. *class clusters*) su programskišsablon koji se često koristi u pisanju programa na Objective-C jeziku. Klasteri klasa grupišu odredjeni broj privatnih podklasa unutar javne apstraktne nadklase. Ovakvo grupisanje pojednostavljuje javno vidljivu arhitekturu objektno orijentisanih biblioteka i radnih okvira, bez smanjenja njihove funkcionalnosti. Apstraktna nadklasa mora da deklariše metode za kreiranje instanci svojih privatnih podklasa.

8.1.2 Introspekcija

Introspekcija je mocno svojstvo objektno orijentisanih jezika i okruzenja. Ono predstavlja mogucnost objekta da predstavi detalje o sopstvenoj implementaciji u toku izvršavanja programa. Ti detalji mogu biti njegovo mesto u drvetu nasledjivanja, da li implementira odredjeni protokol i da li odgovara na neku poruku. Primer:

```
while ( id anObject = [objectEnumerator nextObject] ) {
    if ( [self class] == [anObject superclass] ) {
        // do something appropriate...
    }
}
```

8.1.3 Alokacija Objekata

Prilikom alokacije objekata, dodeljuje se zahtevana memorija iz regiona virtualne memorije dostupne programu. Da bi se izračunalo koliko memorije treba alocirati, uzimaju se u obzir instancirane promenjlive tog

objekta - uključujući njihove tipove i redosled - kao sto je navedeno u klasi objekta.

8.1.4 MVC

MVC (eng. *Model-View-Controller*) programski šablon je šablon visokog nivoa koji predstavlja globalnu arhitekturu aplikacije i razvrstava objekte prema ulozi koju imaju u toj aplikaciji. Objektno orijentisano programiranje ima više koristi od ovog modela; povećava se modularnost korišćenih objekata, njihovi interfejsi su bolje definisani. Sami programi se bolje adaptiraju promeni zahteva. MVC smatra da postoje tri vrste objekata: model, pogled, i kontroler objekti. MVC šablon definiše uloge koji ovi objekti imaju u aplikaciji i način na koji komuniciraju. Pri dizajniranju aplikacije, veliki korak je biranje - ili pravljenje - klasa za objekte koji pripadaju jednoj od ovih uloga.

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

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

10.1 Instalacija neophodnih paketa

Da bismo mogli da kompajliramo programe napisane u Objective-C, potrebno je prvo da instaliramo par paketa. Za kompajliranje Objective-C programa koristi se gcc kompajler[1], uz jos par dodataka. Prvo što treba da uradimo je da instaliramo gcc podršku za Objective-C.

```
1000 $ sudo apt-get install gobjc
```

Zatim treba da instaliramo radno okruženje (eng. *framework*) na kom se moderni Objective-C zasniva i bez kog ne bi imalo puno smisla raditi (iako je moguće). To okruženje se zove GNUstep i instalira se komandom

```
1000 $ sudo apt-get install gnustep
     $ sudo apt-get install gnustep-devel
```

10.2 Kompajliranje

Svaki put kada pokrenemo novu sesiju, moramo da pokrenemo jedan skript, da bismo mogli da kompajliramo Objective-C programe.

```
1000 $ chmod +x /usr/share/GNUstep/Makefiles/GNUstep.sh
     $ /usr/share/GNUstep/Makefiles/GNUstep.sh
```

Da ne bismo svaki put morali da izvršavamo GNUstep.sh, možemo da uradimo sledeće:

```
$ echo /usr/share/GNUstep/Makefiles/GNUstep.sh >> ~/.
    bashrc
```

I na taj način, svaki put kad se pokrene nova sesija, bice izvršen GNUstep.sh i nećemo morati da brinemo.

Kao što smo već naveli, Objective-C koristi GNU-ov C kompajler, uz dodatnu podršku. Sem standardnih argumenata, gcc-u su potrebni i dodatni argumenti (eng. *flags*), kao što su -lobjc, -lgnustep-base...

Dakle ako želimo da prevedemo hello.m program, moramo da izvršimo sledeće:

```
$ gcc hello.m -o hello `gnustep-config --objc-flags` -
    lgnustep-base -lobjc
```

Izvršavanje programa se radi uobičajeno:

```
$ ./hello
```

11 Primer jednostavnog koda i njegovo objašnjenje

Zatim, navedimo primer zastarelog (od verzije MAC 10.5), ali zanimljivog koda koji omogućava „imitiranje"(engl. posing).

```
#import <Foundation/Foundation.h>

@interface MyString : NSString

@end

@implementation MyString

- (NSString *)stringByReplacingOccurrencesOfString:(
    NSString *)target
withString:(NSString *)replacement {
    NSLog(@"The Target string is %@",target);
    NSLog(@"The Replacement string is %@",replacement);
}

@end

int main() {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc]
     init];
    [MyString poseAsClass:[NSString class]];
    NSString *string = @"Test";
    [string stringByReplacingOccurrencesOfString:@"a"
     withString:@"c"];

    [pool drain];
    return 0;
}
```

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

Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst.

12.1 ... podnaslov

Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst.

12.2 ... podnaslov

Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst. Ovde pišem tekst.

13 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.

Literatura

[1] Free Software Foundation. GNU gcc, 2013. on-line at: http://gcc.gnu.org/.

[2] J. Laski and W. Stanley. *Software Verification and Analysis*. Springer-Verlag, London, 2009.

[3] A. M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

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.