***MikroJava kompajler***

Programski prevodioci 1

Elektrotehnički fakultet Univerziteta u Beogradu

Nastavnik: ***dr Dragan Bojić, redovni profesor***

Asistenti: ***mast. inž. Maja Vukasović***,  
***mast. inž. Mihajlo Ogrizović***,  
***dipl. inž. Luka Hrvačević***

Student: ***Predrag Pešić 2020/0023***

***Kratak opis postavke zadatka***

Cilj projektnog zadatka je realizacija kompajlera za programski jezik MikroJavu. Kompajler omogućava prevodjenje sintaksno i semantički ispravnih Mikrojava programa u MikroJava bajtkod koji se izvršava na virtuelnoj mašini za MikroJavu. Sintaksno i semantički ispravni MikroJava programi su definisani specifikacijom [MJ].

Programski prevodilac za Mikrojavu ima četiri osnovne funkcionalnosti: leksičku analizu, sintaksnu analizu, semantičku analizu i generisanje koda.

Leksička analiza podrazumeva prepoznavanje i izdvajanje tokena programa, na osnovu teksta napisanog u okviru MikroJava programa.

Sintaksna analiza podrazumeva zamenu prepoznatih tokena odgovarajućim terminalima i neterminalima (tj smenama), u cilju provere da li je moguće dobiti početnu smenu, tj u cilju provere da li je program sintaksno ispravno napisan.

Semantička analiza podrazumeva popunjavanje tabele simbole i istovremene provere semantičke ispravnosti programa, koja između ostalog podrazumeva proveru dodele tipova, da li je upotreba nekog imena prethodno deklarisana, da li postoje višestruke deklaracije itd. Samo semantički ispravni programi idu na sledeću fazu.

Generisanje koda podrazumeva generisanje odgovarajućih instrukcija koje napisani tekst u MikroJava programu pretvaraju u MikroJava bajtkod koji se izvršava na MikroJava virtuelnoj mašini.

***Opis komandi***

*Generisanje java koda (prevodjenje koda kompajlerom)*

* Desni klik na build.xml
* Run as
* 4 Build Project.xml (Ant build)

Alternativno

* Desni klik na build.xml
* Run as
* 2 Ant Build..
* Izabrati target: compile [default]

*Pokretanje resenja*

* Ekspandovati test folder
* Ekspandovati rs.ac.bg.etf.pp1 folder
* Otvoriti klasu MJCodeGenTest.java
* Kao fajl za sourceCode priloziti putanju do zeljenog testa
* Desni klik
* Run as
* 1 Java Application

*Testiranje resenja*

* Desni klik na build.xml
* Run as
* 5 Disassamble.xml (Ant Build) [obično pokretanje]
* 6 Debug.xml (Ant Build) [debug pokretanje]

Alternativno

* Desni klik na build.xml
* Run as
* 2 Ant Build…
* Izabrati target: runObj [za obično pokretanje]
* Izabrati target: debug [za debug pokretanje]

***Opis prilozenih test primera***

1. program\_lexer.mj – najosnovnije testiranje leksera i prepoznavanje bazicnih tokena
2. program\_parser.mj – najosnovnije testiranje parsera i sintaksne ispravnosti programa
3. program\_analyzer\_0.mj – printovi i izrazi
4. program\_analyzer\_1.mj – prostori imena, promenljive, konstante i funkcije
5. program\_analyzer\_2.mj – kompleksniji izrazi i dodele vrednosti elementima niza
6. program\_analyzer\_3.mj – klase
7. program\_analyzer\_4.mj – kreirenje objekata klasa
8. program\_analyzer\_5.mj – nasledjivanje i polimorfizam
9. program\_analyzer\_6.mj – staticki inicijalizatori, raspakivanje niza, kontrolne strukture
10. program\_analyzer\_7.mj – range, print sa nizovima, list comprehension
11. program\_analyzer\_8.mj – primer sajta
12. program\_codegen\_0.mj – printovi i izrazi
13. program\_codegen\_1.mj – prostori imena, promenljive, konstante i funkcije
14. program\_codegen\_2.mj – kompleksniji izrazi i dodele vrednosti elementima niza
15. program\_codegen\_3.mj – klase
16. program\_codegen\_4.mj – kreirenje objekata klasa
17. program\_codegen\_5.mj – nasledjivanje i polimorfizam
18. program\_codegen\_6.mj – staticki inicijalizatori, raspakivanje niza, kontrolne strukture
19. program\_codegen\_7.mj – range, print sa nizovima, list comprehension
20. program\_codegen\_8.mj – primer sajta

***Opis novouvedenih klasa***

1. Codee – pisanje nekih slozenijih instrukcija
2. CodeGenerator – implementacija cetvrte faze
3. NewVisitor – visitor sa podrskom za bool i rekurzivno ispisivanje klasa
4. Objj – dodavanje prostora imena, stat promenljivih i labela kao novih tipova objekata
5. SemanticAnalyzer – implementacija trece faze
6. Tabb – pisanje slozenijih provera za tabelu simbola, kao i slozenija umetanja
7. Runn – klasa koja omogucava peekovanje expr stacka; s obzirom na to da bi ovo verovatno bio hack, ova klasa uopste nije koriscena
8. Mjlexer.flex – implementacija prve faze
9. Mjparser.cup – implementacija druge faze
10. MJAnalyzerTest – pokretanje testova za trecu fazu
11. MJCodeGenTest – pokretanje testova za cetvrtu fazu
12. MJLexerTest – pokretanje testova za prvu fazu
13. MJParserTest – pokretanje testova za drugu fazu