Univerzitet u Beogradu
Elektrotehnički fakultet
Katedra za računarsku tehniku i informatiku

Programski prevodioci 1

- domaći zadatak -

Autor:
Uroš Petrović 2018/0674

Sadržaj

Opis zadatka

Potrebno je napraviti kompajler za programski jezik Mikrojava. Kompajler treba da pruži mogućnost prevodjenja sintaksno i semantički ispravnih Mikrojava (u daljem tekstu MJ) programa i generisanje MJ bajtkoda koji se izvršava na virtuelnoj mašini za MJ.

Alati koji su dozvoljeni prilikom implementacije rešenja:

- Biblioteka JFlex.jar pomoću koje se .flex specifikacija leksičkog analizatora transformiše u implementaciju leksera na jeziku Java;
- Biblioteka cup_v10k.jar koja predstavlja AST-CUP generator, odnosno generator sintaksnih analizatora;
- Biblioteka symboltable-1-1.jar koja predstavlja implementaciju tabele simbola na jeziku Java;
- Biblioteka mj-runtime-1.1.jar koja sadrži implementaciju alata Code, disasm i Run koji su potrebni prilikom generisanja bajtkoda i pokretanje tako generisanog koda na virtuelnoj mašini za MJ.

Opis test primera

1. Provera oporavka od greške: definicija globalne promenljive ili globalne konstante;
2. Provera oporavka od greške: konstrukcija iskaza dodele;
3. Provera semantičke greške: deklarisanje promenljive koja je već deklarisana;
4. Provera semantičke greške: korišćenje imena u programu koje nije deklarisano;
5. Provera semantičke greške: dodela vrednosti globalnoj konstanti koja se ne poklapa sa njenim navedenim tipom;
6. Provera semantičke greške: korišćenje nepostojećeg tipa podatka;
7. Provera semantičke greške: dodeljivanje vrednosti, inkrementiranje, dekrementiranje, new operator, pristup element niza, nizovska dodela, aritmetički izrazi;
8. Sintaksno i semantički ispravan MJ program koji pokriva sve smene iz gramatike (nadograđena verzija javnog testa).
9. Javni test

Opis komandi za pokretanje i testiranje rešenja

Prevođenje koda kompajlerom se vrši pomoću build.xml skripte. Iz razvojnog okruženja je potrebno odabrati Run as Ant Build (kompajliranje je podrazumevano pravilo, naravno može se izabrati bilo koje pravilo ponaosob radi testiranja funkcionalnosti).

Generisanje koda može da se pokrene iz razvojnog okruženja tako što se pokrene izvršavanje klase Compiler sa odgovarajućim argumentima ili može da se pokrene preko komandne linije tako što se pozicionira u folder projekta i unese sledeća komanda:
java -cp ./src;./config;lib/cup_v10k.jar;lib/mj-runtime-1.1.jar;lib/symboltable-1-1.jar;lib/log4j-1.2.17.jar;lib/JFlex.jar rs.ac.bg.etf.pp1.Compiler test/nameOfSrcFile.mj test/nameOfSrcFile.obj >test/izlazNameOfSrcFile.out 2>test/izlazNameOfSrcFile.err

Izvršavanje bajtkoda u MJ virtuelnoj mašini može da se pokrene pomoću build.xml skripte ili iz komandne linije. Preporuka je koristiti komandnu liniju jer u build.xml skripti mora stalno da se menja naziv objektnog fajla koji predstavlja ulaz. Pozicionira se u folder projekta i unese se sledeća komanda:
java -cp lib/mj-runtime-1.1.jar rs.etf.pp1.mj.runtime.Run putanjaDoFajla

Postoji i opcija za debagovanje, koja prikazuje stanje steka nakon svake instrukcije. Potrebno je u gore navedenu komandu za pokretanje objektnog fajla dodati fleg -debug.

Ispis generisanih instrukcija može da se uradi pomoću disasm alata. Pozicionira se u folder projekta i unese se sledeća komanda:
java -cp lib/mj-runtime-1.1.jar rs.etf.pp1.mj.runtime.disasm putanjaDoFajla

Kratak opis klasa

- **sym.java** – automatski generisana klasa pomoću CUP alata koja sadrži konstante simbola;
- **Yylex.java** – automatski generisana klasa pomoću alata JFlex koja predstavlja skener ulaznog fajla sa izvornim kodom;
- **VisitorCounter.java** – pomoćna klasa koja broji koliko ima argumenata funkcije i lokalnih promenljivih;
- **CodeGenerator.java –** klasa pomoću koje se generiše kod koji je izvršiv na MJ virtuelnoj mašini;
- **Compiler.java –** klasa u kojoj se nalazi main metoda i koja je zadužena za pozivanje svih pomoćnih klasa za leksičku, sintaksnu i semantičku analizu kao i za generisanje koda;
- **MJParser.java –** automatski generisana klasa pomoću CUP alata koja predstavlja parser;
- **SemanticAnalyzer.java –** klasa koja proverava semantičku ispravnost izvornog koda;
- **SymTable.java –** izvedena klasa iz klase Tab koja se nalazi u biblioteci symboltable-1-1.jar (dodata podrška za bool tip podatka, kao i podrška za ispisivanje bool tipa podatka);
- **EditedDumpSymbolTableVisitor.java –** pomoćna klasa koja samo dodaje podršku za ispis bool tipa podatka iz tabele simbola;
- **Log4JUtils.java** – pomoćna klasa koja pravi logove za svako novo pokretanje;
- **MJTest.java –** pomoćna klasa za testiranje koja isključivo služi za testiranje leksičke analize;
- **MJParserTest.java –** pomoćna klasa koja služi za testiranje leksičke, sintaksne i semantičke analize kao i za generisanje koda.