Programski prevodioci 1

Domaći zadatak

Izrada kompajlera za “Microjava” programski jezik

Profesor: Dragan Bojić

Asistenti: Maja Vukasović, Mihajlo Ogrizović

Student: Jovana Križan 2016/0557 Datum: 16.2.2022., U Beogradu

# Opis projekta

Cilj projekta je izrada kompajlera za programski jezik MicroJava. Realizovan je A nivo postavke zadatka, koji obuhvata osnovne iskaze, aritmetičke iskaze i rad sa nizovima prostih tipova.

## Pregled postavke zadatka za A nivo

* Definicija programa
* Definicija globalnih promenljivih, koje mogu biti tipova int, char i bool
* Definicija lokalnih promenljivih, koje mogu biti tipova int, char i bool
* Definicija globalnih konstanti
* Naredbe dodele vrednosti
* Naredbe print i read na/sa standardnog izlaza/ulaza
* Naredba goto
* Definicija labele
* Definicija operacija za sabiranje, oduzimanje, množenje i deljenje

## Korišćeni alati prilikom realizacije

Kompajler je realizovan korišćenjem pomoćnih alata, pomoću kojih su realizovane leksička analiza, sintaksna analiza i tabela simbola, i za kraj i generisanje koda.

* Za realizaciju leksičkog analizatora korišćen je alat Jflex. Ovaj alat na osnovu flex specifikacije generiše Yylex klasu koja vrši leksičku analizu.
* Za realizaciju sintaksnog analizatora korišćena je CUP biblioteka. Ovaj alat na osnovu cup specifikacije generiše klasu MJParser koja vrši sintaksnu analizu, klasu sym u kojoj kodira sve dobijene lekseme od leksičkog analizatora i generiše folder ast u kome generiše klase pomoću kojih se kreira apstraktno sintaksno stablo. Takođe, u folderu ast generišu se interfejs Visitor i klasa VisitorAdaptor, pomoću kojih je moguće vršiti obilazak stabla, pri čemu se visit metode izvršavaju u postorder redosledu.
* Za realizaciju semantičke analize i generisanja koda korišćena je implementacija biblioteke symboltable. Ova biblioteka sadrži realizaciju tabele simbola i definiše brojne konstante i tipove koji olakšavaju proveru semantičke ispravnosti programa.
* Za generisanje koda i njegovo pokretanje, korišćena je i biblioteka mj-runtime.
* U cilju lakšeg testiranja korišćena je biblioteka log4j.

## Klase korišćene za realizaciju

* Pored generisanih i bibliotečnih klasa, uvedene su dodatne klase za realizaciju semantičke analize i generisanja koda.
* Klasa SemanticAnalyzer nasleđuje klasu VisitorAdaptor i implementira visit metode za čvorove za koje je to neophodno, kako bi se proverila semantička ispravnost ulaznog programa.
* Klasa CodeGenerator takođe vrši obilazak apstraktnog sintaksnog stabla realizacijom klase VisitorAdaptor. U njenim visit metodama vrši se generisanje binarnog koda.
* Klasa SymbolTablePomClass je pomoćna klasa za tabelu simbola, koja sadrži funckije poput insert, find, dump i tako dalje.
* Klasa LabelaPom je pomoćna klasa za smeštanje labela, i služi za goto statement-e.
* U folderu test nalazi se klasa MJParserTest pomoću koje je moguće pokrenuti kompajler. Ova klasa očekuje dva argumenta komandne linije, putanje do ulaznog i izlaznog fajla.

## Pokretanje

Generisanje parsera:

* Iz ant-a pokrenuti target parserGen

Pokretanje kompajliranja:

* Iz ant-a pokrenuti target compiler

Pokretanje koda:

* Iz ant-a pokrenuti target runObj

Build.xml skripta:

* Program ćemo pokretati uz pomoć ove skripte. Target disasm omogućava da se vidi dobijeni kod u tekstualnom obliku.

## Testiranje

U folderu test nalaze se i paketi test primerima za svaki deo kompajlera(leksička analiza, semantička analiza i generisanje koda). Testovi obuhvataju veliki broj ispravnih i neispravnih test primera za svaki deo kompajlera.

* Testovi lexTest i lexTest2 su namenjeni za testiranje leksičke analize.
* Testovi u folderu sintaksnaAnaliza, od test10 do test14 su testovi sa ispravnom sintaksom, dok su testovi test21 i test22 sa greskama.
* Testovi u folderu semantickaAnaliza , od test30 do test33 su testovi sa greskama, i vidimo kako ih prepoznaje.
* Test u folderu generisanjeKoda je javni test sa sajta Programskih prevodioca 1.
* I jos par nekih testova koriscenih pri pravljenju ovog projekta.