Elektrotehnicki fakultet u Beogradu

Katedra za racunarsku tehniku i informatiku

**Projekat**

**– Kompajler za Mikrojavu –**

Marko Hudomal 2015/0112

Opis zadatka

Zadatak projekta je realizacija kompajlera za programski jezik Mikrojava. Kompajler omogućava prevodjenje sintaksno i semantički ispravnih Mikrojava programa u Mikrojava bajtkod koji se izvršava na virtuelnoj mašini za Mikrojavu.

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

1. ***Leksički analizator***prepoznaje jezičke lekseme i vrati skup tokena izdvojenih iz izvornog koda, koji se dalje razmatraju u okviru sintaksne analize.
2. ***Sintaksni analizator***ima zadatak da utvrdi da li izdvojeni tokeni iz izvornog koda programa mogu formiraju gramatički ispravne sentence.
3. ***Semantički analizator***se formira na osnovu apstraktnog sintaksnog stabla koje je nastalo kao rezultat sintaksne analize.
4. ***Generator koda***prevodi sintaksno i semantički ispravne programe u izvršni oblik za odabrano izvršno okruženje Mikrojava VM.

Opis komandi

Pomocu Ant java biblioteke sa definisanim komandama:

1. mjlexer.flex  
<target name="lexerGen" depends="deleteLexer">

<java jar="lib/JFlex.jar" fork="true">

<arg value="-d"/>

<arg value="./src/rs/ac/bg/etf/pp1"/>

<arg value="spec/mjlexer.flex"/>

</java>

</target>

I ukljucivanjem biblioteke JFLex generise se Ylex.class u kome je opisano izdvajanje tokena iz ulaznog fajla. (Koristi sym klasu) Potrebno je pre toga napisati sym klasu

2. mjparser.cup  
<target name="parserGen" depends="deleteParser">

<java jar="lib/cup\_v10k.jar" fork="true">

<arg value="-destdir"/>

<arg value="src/rs/ac/bg/etf/pp1"/>

<arg value="-ast"/>

<arg value="src.rs.ac.bg.etf.pp1.ast"/>

<arg value="-parser"/>

<arg value="MJParser"/>

<arg value="-dump\_states"/>

<arg value="-buildtree"/>

<arg value="spec/mjparser.cup"/>

</java>

</target>  
Generise sve klase vezane za cvorove, u .ast paketu.  
I ukljucivanjem biblioteke cup generise se sym.class i mjparser\_astbuild.cup u kome je opisano izdvajanje tokena iz ulaznog fajla. (Koristi Yles klasu)

3.SemanticAnalyzer.class

<target name="compile" depends="repackage">

<javac srcdir="src/rs/ac/bg/etf/pp1" includeantruntime="false">

<classpath>

<pathelement path="lib/JFlex.jar"/>

<pathelement path="lib/cup\_v10k.jar"/>

<pathelement path="lib/log4j-1.2.17.jar"/>

<pathelement path="lib/symboltable-1-1.jar"/>

<pathelement path="lib/mj-runtime-1.1.jar"/>

</classpath>

</javac>

</target>

Generise sve klase vezane za cvorove, u .ast paketu

4. CodeGenerator.class  
<target name="disasm">

<java classname="rs.etf.pp1.mj.runtime.disasm">

<arg value="test/program.obj"/>

<classpath>

<pathelement location="lib/mj-runtime-1.1.jar"/>

</classpath>

</java>

</target>

<target name="runObj" depends="disasm" >

<java classname="rs.etf.pp1.mj.runtime.Run">

<redirector input="test/input.txt"></redirector>

<arg value="test/program.obj"/>

<classpath>

<pathelement location="lib/mj-runtime-1.1.jar"/>

</classpath>

</java>

</target>

Koristi se biblioteka mj-runtime koju je potrebno prethodno referisati

**Kompajliranje i pokratenje programa:**

Pokretanjem main metode klase Compiler kod cija je putanja prosledjena kao prvi argument programa kompajlira se u fajl koji je prosledjen kao drugi argument funkcije.

Program se moze pokrenuti preko ant-a sto je definisano sa target-om *runObj* ili pokretanjem klase Run iz definisane biblioteke.

Opis test primera

testLex.mj – provera svih mogucih tokena definisanih u projektu

testParserSintaksa.mj – provera sintaksne ispravnosti projekta

testParserErrors – oporavak i sve moguce greske

testSemantika1.mj – razlicite kombinacije programa pogodne za semanticku analizu

testSemantika2.mj – razlicite kombinacije programa pogodne za semanticku analizu

test301.mj – zvanicni test kod za generisanje koda

test302.mj – zvanicni test kod za generisanje koda

testUltimate.mj – test koji kompletno pokriva projektne uslove. Na kraju treba da prijavi error jer se pristupa null referenci.

Opis novouvedenih klasa

Faza 1 (pored *mjlexer.flex* fajla)

MJTest.java – klasa za testiranje i obradu leksike koda

Ylex.java – generisana klasa u fazi 1. (Objasnjeno u drugom poglavlju)

Faza 2 (pored *mjparser.cup* fajla)

MJParserTest.java – klasa za testiranje i obradu sintakse koda

RuleVisitor.java – klasa za obilazak generisanog stabla u sintaksnoj analizi.

sym.java – generisana klasa u fazi 2. (Objasnjeno u drugom poglavlju)

MJParser.java – klasa koju generise biblioteka parsera i sadrzi sve informacije vezane za ostale generisane klase u .ast paketu

Faza 3

Compiler.java – klasa za pokretanje kompajlirana, kasnije nadogradjena u fazi 4. I za generisanje koda

SemanticAnalyzer.java – klasa za obilazak generisanog stabla u semantickoj analizi. Nadogradjuje generisano stablo i proverava semantiku istog.

Faza 4

CodeGenerator.java – klasa za obilazak generisanog stabla kod generisanja koda. Koristi polja i vrednosti i generise kod.

ConuterVisitor.java – Broji ponavljanja odredjeni cvorova. U nasem slucaju koristili za proj parametara i lokalnih promenljivih u okviru meetoda/funkcija.