Elektrotehnički fakultet u Beogradu

Katedra za računarsku tehniku i informatiku

Predmet: Programski prevodioci 1

Školska: 2022/2023.

Ispitni rok: Julski ispitni rok

**Projekat**

**- Kompajler za Mikrojavu -**

Vuk Milic 2017/0132

# -Opis 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.

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

**Leksički analizator** je implementiran kroz mjlexer.lex fajl. On prima tekstualni fajl (MicroJava kod) koji deli na tokene, i te tokene prosledjuje dalje sintaksnom analizatoru. Leksički analizator detektuje greške u koliko prilikom analize ne prepozna neki deo teksta.

**Sintaksni analizator** ima zadatak da utvrdi da li izdvojeni tokeni iz izvornog koda programa mogu da formiraju gramatički ispravne sentence. On je implementiran kroz mjparser.cup fajl I prima tokene koje je leksički analizator generisao, spaja ih u sentence I proverava njihovu ispravnost. Takođe, vrši detekeciju grešaka I po mogućstvu radi oporavak od grešaka (u nekim slučajevima). Njegov rezultat je apstraktno sintaksno stablo koje prosleđuje Semantičkom analizatoru.

**Semantički analizator** se formira na osnovu apstraktnog sintaksnog stabla. Implementira se kroz klasu SemanticAnalyzer.java u kojoj se vrši posećivanje čvorova tog stable. Prilikom posećivanja čvorova, vrši se semantička provera I detekcija grešaka.

**Generator koda** prevodi sintaksno i semantički ispravne programe u izvršni oblik za izvršno okruženje Mikrojava VM. Implementiran je kroz klasu CodeGenerator.java u kojoj se vrši posećivanje čvorova prethodno navedenog stable. Prilikom posećivanja čvorova, vrši se generisanje koda koji se na kraju izvršava.

# -Opis komandi

-Generisanje klase leksičkog analizatora:

<target name="lexerGen" depends="delete">

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

<arg value="-d"/>

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

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

</java>

</target>

-Generisanje klase parsera:

<target name="parserGen" depends="delete">

<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="-buildtree"/>

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

</java>

</target>

-Kompajliranje i dodavanje neophodnih biblioteka:

<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>

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

<!-- Replaces all of the references to the old package name in files in the "src" directory -->

<replace dir="src" value="rs.ac.bg.etf.pp1.ast" token="src.rs.ac.bg.etf.pp1.ast" summary="true"/>

</target>

-Disasembliranje objektnog fajla:

<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>

-Pokretanje prevedenog programa (uz mogućnost debagovanja):

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

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

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

<arg value="-debug"/>

<classpath>

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

</classpath>

</java>

</target>

# -Opis test primera

* **TestSintaksaOporavak.mj** – testira se ispravnost sintaksnih sentenci kao i oporavak od grešaka prilikom definicija globalnih promenljivih (dva oporavka – do tačke-zareza kao i do zareza). Takođe se testira oporavak od greške prilikom dodele vrednosti.
* **TestSintaksaNizMat.mj** – testira se ispravnost sintaksnih sentenci vezanih za nizove i matrice
* **TestSemDeklaracija.mj** – testira se semantika prilikom pristupa globalnim, lokalnim promenljivim , kao i simboličkim konstantama. Takođe, testira se deklaracija metoda kao i njihov pristup i pristup elementu niza.
* **TestSemStatementa.mj** – testira se semantika svih Statementa koji ne uključuju rad sa klasama.
* **TestSemGreskeDeklaracije.mj** – testiraju se greške prilikom deklaracija globalnih i lokalnih promenljivih, kao i simboličkih konstanti. Takođe, greška prilikom korišćenja simbola koji nije prethodno deklarisan ili je ponovno deklarisan.
* **TestSemGreskeStatementa**.**mj** – testiraju se ostale moguće semantičke greške.
* **TestSemUgradjeniObjekti.mj** – testira se semantika ugrađenih objekata kao što su eol, chr, ord i len; kao i greške koje se mogu pojaviti prilikom pristupa tim objektima.

# -Opis novouvedenih klasa

* TabBool– klasa koja proširuje klasu Tabdodavanjem novog tipa – boolType.
* BoolDumpSymbolTableVisitor– klasa koja proširuje klasu DumpSymbolTableVisitor tako što override-uje metodu visitStructNode dodavanjem case-ova za tip bool kao i za matrice.
* CounterVisitor – klasa koja proširuje klasu VisitorAdaptor tako što sadrži brojač lokalnih promenljivih i formalnih parametara