Електротехнички факултет у Београду
Катедра за рачунарску технику и информатику

Предмет: Програмски преводиоци 1
Наставник: доц. др. Драган Бојић
Асистент: дипл. инг. Немања Којић
Школска година: 2010/2011.
Испитни рок: јануар 2011.

Пројекат за домаћи рад

- Компајлер за Микројаву –

студент**: Милан Бранковић 119/07**

Верзија: 1.0

Садржај

Опис проблема

У оквиру датог пројектног задатка потребно је реализовати компајлер за микројаву који омогућава превођење синтаксно и сематички исправних Микројава програма у Микројава бајткод који може да се извршава на Микројава виртуелној машини. Дати пројекат се извршава у две фазе.

Циљ прве фазе пројекта је да омогући парсирање синтаксно исправних и синтаксно неисправних програма. Током парсирања Микорјава програма потребно је на одговарајући начин омогућити праћење самог процеса парсирања. Након парсирања синтаксно исправних Микројава програма потребно је на крају обавестити корисника о успешности парсирања. Међутим уколико се врши парсирање програма са синтаксним грешкама, потребно је издати адекватно објашњење о детектованој синтаксној грешки, извршити опоравак и наставити парсирање.

Прва фаза се реализује кроз две подфазе:

- Лексичка анализа: У оквиру ове фазе потребно је реализовати лексички анализатор (скенер) Микројава фајлова. Скенер прихвата фајл у коме се налази изворни код који одговара спецификацији Микројаве (у наставку MJ) и дели га на токене. Тип токена се враћа при експлицитном позиву лексичког анализатора (позив next_token()). У случају лексичке грешке анализатор враћа токен грешке (sym.INVALID) и наставља анализу до краја улазног фајла. Имплементација је одрађена на програмском језику JAVA коришћењем алата JFLEX.
- Синтаксна анализа: У оквиру ове фазе потребно је дефинисати LALR(1) граматику и имплементирати синтаксни анализатор (парсер) за програме написане на MJ. У случају успешног парсирања улазног фајла парсер на крају рада на стандардном излазу приказује поруке о броју препознатих језичких елемената и на крају поруку о успешном парсирању. У случају наиласка на грешку парсер пријављује грешку на стандардни излаз за грешке која садржи број линије улазног програма, опис грешке, врши опоравак од грешке и наставља са парсирањем остатка фајла. У случају појаве лексичке грешке грешка се исписује на излазу грешке и игнорише у синтаксној анализи.

У другој фази пројекта наставља се развој компајлера за Микројава програме кроз надоградњу парсера са циљем да се омогући сематичка анализа и превођење програмског кода написаног на Микројави у извршни облик за одабрано извршно окружење.

Друга фаза се такође реализује кроз две подфазе:

- Семантичка анализа: У оквиру ове фазе се синтаксни анализатор који је конструисан у претходној фази проширује тако да врши семантичку анализу MJ фајлова и ажурира табелу симбола. Табела симбола је имплементирана помоћу хеш табеле како је задато поставком задатка. Додавањем семантичких акција у парсер, омогучава се унос симбола у табелу симбола.
- Генерисање кода: У оквиру ове фазе семантички анализатор се проширује да на основу исправног MJ кода генерише MJ бајткод који се може извршавати помоћу MJ виртуелне машине. Користе се приложене класе Code, disasm и Run за генерисање кода, испис генерисаног кода и извршавање (интерпретирање) преведеног програма.

Списак команди

За покретање из командне линије најпре је потребно подесити да се подеси classpath

set PATH=%PATH%; %PATH% = путања_до_JABA_директоријума\jdk1.6.0_xx\bin (где xx представља верзију)

затим је потребно да се лоцирамо у одредишни директоријум, односно у директоријум у коме је рад. То ћемо учинити помоћу следеће команде

cd путања_до_одредишног_директоријума

изворни код парсера и класе sym добијамо помоћу команде

java -jar java-cup-11a.jar -destdir директоријум директоријум\име_фајла.cup

изворни код лексичког анализатора добијамо помоћу команде

java –jar JFlex.jar име_фајла.flex

затим се компајлирају фајлови помоћу команди

javac директоријум\sym.java

javac -cp .;java-cup-11a.jar директоријум\Lexer.java

javac –cp .;java-cup-11a.jar директоријум\parser.java

напомена: потребно је да архива `java-cup-11a.jar` буде у classpath-у

програм се покреће са

java –cp .;java-cup-11a.jar директоријум.parser tst\тест_фајл >tst\ тест_фајл.out

за синтаксно исправне програме, односно са

java –cp .;java-cup-11a.jar директоријум.parser tst\тест_фајл >tst\ тест_фајл.out 2>tst\ тест_фајл.err

за синтаксно неисправне програме.

За конкретан случај следећа секвенца команди доводи до извршења програма

```
set PATH="c:\Program Files\Java\jdk 1.6.0_20\bin"
cd c:\pp_dz
java -jar java-cup-11a.jar -destdir microJava microJava\spec.cup
```

java -jar JFlex.jar microJava\myFlex.flex
javac microJava\sym.java
javac -cp .;java-cup-11a.jar microJava\Lexer.java
javac -cp .;java-cup-11a.jar microJava\parser.java
java -cp .;java-cup-11a.jar microJava.parser tst\errors\3_deo_error.mj
java -cp .;java-cup-11a.jar microJava.parser tst\errors\3_deo_error.mj >tst\errors\3_deo_error.out 2>tst\ errors\3_deo_error.err

Претходне наредбе покривају први део домаћег задатка. Следеће наредбе употпуњују и комплетирају целокупан домаћи.

Да би се добиле извршне верзије дисасемблера и виртуелне машине користе се следеће наредбе

javac директоријум\disasm.java

javac директоријум\Run.java

Компајлер који је имплементиран покреће се командом

java -cp .;java-cup-11a.jar директоријум.parser директоријум_тест\ тест_фајл.pr директоријум_тест\ тест_фајл.obj

Садржај добијеног фајла може се погледати комадном

java директоријум.disasm директоријум_тест\ тест_фајл.obj

Извршавање преведеног програма у дебуг режиму

java директоријум.Run -debug директоријум_тест\ тест_фајл.obj

или у „обичном“ режиму

java директоријум.Run -debug директоријум_тест\ тест_фајл.obj

За конкретан случај следећа секвенца команди доводи до извршења програма

javac codeGeneration \disasm.java
javac codeGeneration \Run.java
java -cp .;java-cup-11a.jar microJava.parser tst\test_smislen\test_pp1.mj
tst\test_smislen\test_pp1.obj
java codeGeneration.disasm tst\ test_smislen\test_pp1.obj
java codeGeneration.Run -debug tst\ test_smislen\test_pp1.obj

Списак порука о грешкама

Поруке које може да генерише синтаксни анализатор:

```
Sintaksna greska
Izvrsen oporavak do ; u liniji
Izvrsen oporavak do } u liniji
Izvrsen oporavak do ) u liniji
Izvrsen oporavak do ] u liniji
Fatalna greska, parsiranje se ne moze nastaviti
```

Поруке које може да генерише семантички анализатор:

```
nije tip
nekompatibilni tipovi za dodelu
metod main mora biti void
metod mora imati return iskaz jer nije deklarisan sa void
metod mora imati return iskaz sa izrazom jer nije deklarisan sa void
metod ne sme imati return sa izrazom jer je deklarisan sa void
tip izraza nekompatibilan sa deklaracijom metoda
nisu podrzane klase kao formalni parametri
ocekivan metod
procedura pozvana kao funkcija
broj parametara se razlikuje
nekompatibilni tipovi pri pozivu metode
operand mora oznacavati promenljivu, element niza ili polje unutar objekta
ocekivana promenljiva,konstanta ili element niza
tip izraza mora biti int ili char
tip promenljive mora biti int
tip mora biti klasa
levi operand mora oznacavati promenljivu, element niza ili polje unutar objekta
desni operand mora biti tipa int
oba operanda moraju biti tipa int
nije polje unutar klase
ocekivan niz
velicina niza mora biti tipa int
izraz unutar zagrada mora biti int
nisu dozvoljeni logicki uslovi
```

Извештај JFlex и CUP алата

```
Reading "microJava\myFlex.flex"
Constructing NFA : 150 states in NFA
Converting NFA to DFA :
.....................................................................................
92 states before minimization, 86 states in minimized DFA
Old file "microJava\Lexer.java" saved as "microJava\Lexer.java~"
Writing code to "microJava\Lexer.java"


Warning : Terminal "INVALID" was declared but never used
------- CUP v0.11a beta 20060608 Parser Generation Summary -------
  0 errors and 1 warning
  42 terminals, 51 non-terminals, and 115 productions declared,
  producing 205 unique parse states.
  1 terminal declared but not used.
  0 non-terminal declared but not used.
  0 productions never reduced.
  0 conflicts detected (0 expected).
  Code written to "parser.java", and "sym.java".
---------------------------------------------------- (v0.11a beta 20060608)
```

Опис тест примера

За тестирање су коришћени фајлови са предавања и вежби, уз неке додатне фајлове који тестирају грешке. Главни тест пример који покрива целокупан рад компајлера дат је у фолдеру pp_dz\tst\test_smislen.