Електротехнички факултет Универзитета у Београду

Домаћи задатак 2013

Компајлер за Микројаву

Програмски преводиоци 1

студент:
Марин Маркић
2009/0487
ИР

1. Опис проблема

Циљ пројектног задатка је реализација компајлера за програмски језик Микројаву. Компајлер омогућава превођење синтаксно и семантички исправних Микројава програма у Микројава bytecode који се извршава на виртуелној машини за Микројаву. Синтаксно и семантички Микројава програми су дефинисани спецификацијом.

Програмски преводилац за Микројаву има четири основне функционалности лексичку анализу, синтаксу анализу, семантичку анализу и генерисање кода.

У првој фази потребно је имплементирати лексички анализатор Микројава програма. Лексички анализатор треба да препознаје језичке лексеме и врати скуп токена издвојених из изворног кода, који се даље разматрају у оквиру синтаксне анализе. Ако нема лексичких грешака прелази се на синтаксу анализу.

У другој фази потребно је имплементирати синтаксни анализатор који је оспособљен и за семантичку анализу кроз синтаксно управљано превођење. Након парсирања синтаксно исправних Микројава програма потребно је обавестити корисника о успешности парсирања. Уколико изворни код има синтаксне грешке, потребно је издати адекватно објашњење о детектованој грешци, извршити опоравак и наставити парсирање.

2. Упуство за превођење и покретање програма

Прва фаза – Лексичка анализа

За имплементацију лексичког анализатора користићемо програмски језик Јава и алат jflex. Алат jflex на основу задате текстуалне спецификације генерише изворни код лексичког анализатора у програмском језику Јава.

На основу текстуалне спецификације: **spec/mjlexer.flex**, покретањем класе jflex.Main добија се имплементација анализатора у Јави, класа **Yylex.java**. Приликом покретања програму се прослеђују аргументи, први је директоријум у коме ће бити креирана класа Yylex.java, а други спецификација:

```
-d src/rs/etf/markic spec/mjlexer.flex
```

После успешног генерисања имплементације анализатора, покрећемо тест програм (**Scanner**) који чита изворни код на микројави и враћа скуп токена и евентуалне грешке. Програму се као први аргумент прослеђује име фајла са изворним кодом на Микројави. Скуп токена ће се налазити у излазном фајлу чије име одређује други аргумент програма, или на стандардном излазу ако други аргумент није наведен.

Пример позива за Микројава програм test/lex1.mj:

```
java -jar Scanner.jar lex1 lex1.out
```

Друга фаза – Синтаксна и семантичка анализа

За имплементацију синтаксног и семантичког анализатора користићемо програмски језик Јава и алат java cup - генератор парсера. Алат cup на основу задате текстуалне спецификације генерише изворни код парсера у програмском језику Јава.

На основу текстуалне спецификације: **spec/mjparser.cup**, покретањем класе java_cup.Main добија се имплементација анализатора у Јави, класа **MJParser.java**. Приликом покретања програму се прослеђују аргументи, први је директоријум у коме ће бити креирана класа MJParser.java, други је име новог анализатора, а трећи спецификација:

```
-destdir src/rs/etf/markic -parser MJParser spec/mjparser.cup
```

После успешног генерисања имплементације анализатора, покрећемо тест програм (**ParserTest**) који парсира изворни код на микројави и проверава синтаксу и семантику програма. Програму се као први аргумент прослеђује име фајла са изворним кодом на Микројави.
Пример позива за Микројава програм test/program.mj:

```
java -jar ParserTest.jar program
```

3. Тест примери

Тест примери који садрже изворни код на Микројава програмском језику налазе се у директоријуму test.

Тест лексичког анализатора 1:

Фајл: **lex1.mj**, тест пример садржи неколико класа, променљивих и израза, и демонстрира наслеђивање, операторе, динамичку алокацију, улаз/излаз и циклусе.

Програм је лексички исправно написан, тако да нема грешака. Скуп токена и њима придружене симболичне вредности налазе се у излазном фајлу **lex1.out**.

Тест лексичког анализатора 2:

Фајл: **lex2.mj,** тест пример демонстрира правилан рад са карактерима, стринговима и поништавањем значења специјалних симбола.

Програм је лексички исправно написан, тако да нема грешака. Скуп токена и њима придружене симболичне вредности налазе се у излазном фајлу **lex2.out**.

Тест лексичног анализатора 3:

Фајл: **lex3e.mj**, тест пример садржи више лексичких грешака, поруке о грешкама налазе се у фајлу **lex3e.err**. Скуп токена и њима придружене симболичне вредности налазе се у излазном фајлу **lex3e.out**.

Тест синтаксног анализатора 1:

Фајл: **sin1.mj**, тест пример садржи више синтаксних грешака, поруке о грешкама исписују се на стандардни излаз грешке.

Тест синтаксног анализатора 2:

Фајл: **sem1.mj**, тест пример је синтаксно и семантички исправан, демонстрира разне семантичке структуре. Синтаксна статистика програма, семантичка анализа и садржај табеле симбола исписују се на стандардни излаз.

Тест синтаксног анализатора 3:

Фајл: **sem2.mj**, тест пример је синтаксно исправан, али садржи више семантичких грешака. Поруке о грешкама исписују се на стандарни излаз грешке.

Тест 4:

Фајл: **program.mj**, тест пример је синтаксно и семантички исправан, демонстрира разне семантичке структуре. Синтаксна статистика програма, семантичка анализа и садржај табеле симбола исписују се на стандардни излаз.

Стандарни излаз се може преусмерити у фајл командом >izlaz, а излаз грешке 2>izlaz.