Универзитет у Београду – Електротехнички факултет

Софтверско инжењерство



**Програмски преводиоци 1**

**Кандидат**

Aњa Марковић 0420/2017

Београд, јануар 2021. године

* **Кратак опис поставке задатка**

Циљ пројектног задатка је реализација компајлера за програмски језик Микројава. Компајлер омогућава превођење синтаксно и семантички исправних Микројава програма у Микројава бајткод који се извршава на виртуелној машини за Микројаву. Синтаксно и семантички исправни Микројава програми су дефинисани спецификацијом. Задатак је био написати наведене спецификације које посредством приложених извршних библиотека генеришу класе за објекте Лексичког и Синтаксног анализатора.

Такође, задатак је био и да се напишу класе за објекте Семантичког анализатора и генератора кода. Улога семантичког анализатора је да обилази чворове синтаксног стабла насталог као продукт парсирања и да при том обилажењу направи табелу симбола.

* **Команде за генерисање кода**

Позиционирати се у *src* фолдер пројекта.

* + **Генерисање лексера**

java -cp ../lib/JFlex.jar JFlex.Main -d rs\ac\bg\etf\pp1 ..\spec\mjlexer.flex >izlaz.out 2>izlaz.err

* + **Генерисање парсера**

java -cp ../lib/cup\_v10k.jar java\_cup.Main -destdir rs\ac\bg\etf\pp1 -parser MJParser -ast rs.ac.bg.etf.pp1.ast -buildtree ..\spec\mjparser.cup >izlaz.out 2>izlaz.err

* + **Компајлирање**

test\program.mj test\program.obj (локални фајл Compiler.java)

* + **Извршавање бајткода**

-debug test\program.obj (библиотека mj-runtime-1.1.jar класа Run.class)

* + **Испис бајткода**

test\program.obj (библиотека mj-runtime-1.1.jar класа disasm.class)

* **Опис приложених тест примера**

Поред јавних тестова, тест *syntax.mj* тестира исправне и неисправне синтаксне конструкције, и проверава опоравак од синтаксних грешака. Тест *semantic.mj* садржи семантички исправне и неисправне исказе.

* **Уведене класе за помоћ током извршавања**

Додате су следеће помоћне класе:

* + *Variable.java* – служи као апстракција променљиве и константе, користи се и у семантичкој анализи и у генератору кода. При семантичкој анализи се користи како би се променљиве и константе убацивале у табелу симбола, а при генерисању кода за декларацију константи.
  + *Method.java* – служи као апстракција методе и користи се само у семантичкој анализи. Улога ове класе је да упамти сваку декларисану методу у обрађиваном програму и упари њене аргументе при позиву методе.
  + *CondJcc.java –* користи се у фази генерисања кода, да се у њој чувају вредности потребне за попуњавање рупа приликом скокова унапред.