UNIVERZITET U BEOGRADU – ELEKTROTEHNIČKI FAKULTET

PROGRAMSKI PREVODIOCI 1 (13E114PP1)

KOMPAJLER ZA PROGRAMSKI JEZIK MIKROJAVA

Izveštaj o urađenom domaćem zadatku

Predavač:

prof. dr. Dragan Bojić

Student:

Jovan Nikolov 2016/0040

Beograd, jun 2020.

Postavka zadatka:

Cilj domaćeg zadatka je realizacija kompajlera za programski jezik MikroJava. Kompajler omogućava prevođenje sintaksno i semantički ispravnih MikroJava programa u MikroJava bajtkod koji se izvršava na MikroJava virtuelnoj mašini. Sintaksno i semantički ispravni MikroJava programi su definisani specifikacijom.

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

Generisanje leksičkog analizatora pomoću alata JFlex:

Klasa **MJLexer** služi za obavljanje leksičke analize. Može se generisati iz **mjlexer.flex** fajla korišćenjem alata **JFlex** iz komandne linije. Opcija **-d** specificira destinacioni folder za generisani izvorni fajl **MJLexer.java**.

```
java -jar lib\JFlex.jar -d src/rs/ac/bg/etf/pp1 spec/mjlexer.flex
```

Prilikom izrade projekta korišćen je build alat Ant i stoga je napravljena i specifikacija koja sadrži cilj **generate.lexer** koji postiže isti rezultat.

Generisanje sintaksnog analizatora pomoću alata Java AST-CUP:

Klasa **MJParser** služi za obavljanje sintaksne analize. Može se generisati iz **mjparser.cup** fajla korišćenjem alata **Java AST-CUP** (ekstenzije alata **Java CUP**) iz komandne linije. Osim ove klase generiše se i klasa **sym** koja predstavlja vezu između leksičkog i sintaksnog analizatora kao i **klase čvorova u sintaksnom stablu** koje generiše ekstenzija AST.

```
java -jar lib\cup_v10k.jar
      -destdir rs/ac/bg/etf/pp1
      -ast rs.ac.bg.etf.pp1.ast
      -parser MJParser -buildtree
      ../spec/mjparser.cup
```

Ovu komandu je potrebno pokrenuti u **src/** direktorijumu kako bi se imena paketa ispravno generisala. Drugi način za postizanje istog rezultata je korišćenjem Ant cilja **generate.parser**.

Prevođenje MikroJava koda u izvršivi objektni fajl:

MikroJava kod (.mj) se može prevesti u objektni fajl (.obj) izvršiv na MikroJava virtuelnoj mašini na sledeći način:

```
java -cp .;config;out\production\MJCompiler;lib\*
      rs.ac.bg.etf.pp1.Compiler <ime_izvornog_fajla>.mj
```

Postoje i dodatne opcije sa kojima se može pokrenuti kompajler i mogu se videti pozivom kompajlera sa opcijom **-h**:

```
usage: mjc [options] input_file
  options:
    -d, --debug                Enables logging of debug messages.
    -h, --help                 Prints this message.
    -o, --output output_file   Specifies output file name.
```

Disasembliranje objektnog fajla:

Kako bi se dobio objektni kod u ljudski čitljivom formatu može se disasemblirati kreirani objektni fajl korišćenjem sledeće komande:

```
java -cp lib\mj-runtime-1.1.jar rs.etf.pp1.mj.runtime.disasm
      <ime_objektnog_fajla>.obj
```

Izvršavanje objektnog fajla:

Za izvršavanje objektnog fajla koristi se MikroJava virtuelna mašina:

```
java -cp lib\mj-runtime-1.1.jar rs.etf.pp1.mj.runtime.Run
      <ime_objektnog_fajla>.obj
```

Testiranje rešenja:

U cilju boljeg testiranja datog rešenja napisano je nekoliko klasa koje testiraju različite delove kompajlera. Sve klase se nalaze u paketu **rs.ac.bg.etf.pp1.test** i izvedene su iz osnovne klase **MJTest** koja sadrži zajedničke metode koje se koriste za testiranje rešenja. Klase koje pokreću testiranje određenog dela kompajlera su **MJLexerTest**, **MJParserTest**, **MJAnalyzerTest** i **MJGeneratorTest**.

Klasa **MJTestRunner** sadrži **main** metodu koja služi za pokretanje testera i prima različite parametre u zavisnosti od testa koji želimo da izvršimo. Ukoliko se pokrene bez ikakvih parametara onda će pokrenuti testiranje svih delova kompajlera za sve test primere koji se nalaze u direktorijumima **test/lexer**, **test/parser**, **test/analyzer** i **test/generator**.

Test primeri u ovim direktorijumima su napravljeni za automatsko testiranje direktorijumi sadrže i očekivane rezultate koji će biti poređeni sa trenutnim rezultatima. Ukoliko se poklapaju test prolazi, u suprotnom se ispisuje greška. **MJTestRunner** se takođe može pokrenuti sa parametrima **lexer**, **parser**, **analyzer** ili **generator**, a nakon njih, **opciono**, može biti i **ime test fajla** koji želimo da proverimo (test fajl se mora nalaziti u odgovarajućem test direktorijumu). Na primer, pokretanje testiranja za semantički analizator može se učiniti na sledeći način:

```
java -cp .;config;out\production\MJCompiler;lib\*
      rs.ac.bg.etf.pp1.test.MJTestRunner analyzer
```

Kratak opis priloženih test primera:

Test primeri za automatsko testiranje pokrivaju širok spektar programa, sintaksno neispravne (**test/parser/parser_recovery_*.mj**), sintaksno ispravne (**test/parse/parser_*.mj**), semantički neispravne (**test/analyer/semantic_errors_*.mj**) i konačno semantički ispravne programe (**test/analyzer/analyzer_*.mj** i **test/generator/*.mj**). Programi koji se nalaze u direktorijumu **test/generator/** se nakon kompajliranja i pokreću u datoj MikroJava virtuelnoj mašini i testira se njihovo izvršavanje. Jedini uslov da test „prođe“ jeste da se izlazi kompajliranog programa poklapaju sa očekivanim. Izlazi kompajlera prilikom kompajliranja programa se proveravaju ali nisu odlučujući. Ovim je postignuto da se kompajler može promeniti i time dobiti drugačiji objektni fajl, ali ukoliko se izlaz novog objektnog fajla poklapa sa očekivanim izlazom pretpostavka je da je test „prošao“.

Testovi od interesa se nalaze u direktorijumu **test/generator/**. Za svaku fazu projekta postoji bar po jedan test koji pokriva većinu slučajeva od interesa. Testovi su sledeći:

- **test/generator/generator_A.mj**
    - Testira generisanje koda za elemente jezika koji se odnose na **nivo A**. Testiraju se **naredbe dodele**, **inkrementiranja** i **dekrementiranja**, **read** i **print** naredbe. Testiraju se **razni tipovi izraza** (semantički ispravnih), **svi tipovi faktora osim** faktora koji predstavlja **poziv metoda**. Testira se **alokacija** i **pristup elementima nizova**. Testiraju se **globalne i lokalne promenljive** i **simboličke konstante**.
- **test/generator/generator_B.mj**
    - Testira generisanje koda za elemente jezika koji se odnose na **nivo B**. Testiraju se **pozivi metoda sa i bez argumenata** kao **zasebne naredbe** ili kao **faktori** u sklopu kompleksnih izraza. Testira se **optimalno izračunavanje kompleksnih kondicionala** koji se potom koriste za testiranje rada **if/if-else** naredbi kao i **for** i **foreach** petlji. Testiraju se naredbe **break** i **continue** u okviru **for** i **foreach** petlji. Testiraju se vrednosti indeksa i iteratora nakon izlaska iz petlji.
- **test/generator/generator_C1.mj**
    - Predstavlja jedan osnovni test program za **nivo C** iz materijala sa vežbi adaptiran za trenutnu specifikaciju MikroJava programskog jezika. Testira nasleđivanje konkretne klase, reimplementaciju nasleđenog metoda, instanciranje objekata konkretnih klasa, generisanje virtuelne tabele metoda.
- **test/generator/generator_C2.mj**
    - Predstavlja kompleksniji test program za **nivo C** koji je napisan specifično za testiranje svih objektno-orijentisanih koncepata MikroJava programskog jezika. Osim testiranja komponenti koje prethodni test obuhvata, testira se i nasleđivanje apstraktnih klasa (implementacija apstraktnih metoda), instanciranje nizova objekata apstraktnih klasa i zamena referenci osnovne klase sa referencama na objektne izvedenih klasa. Testira se takođe polimorfno povezivanje klasnih metoda invokacijama metoda osnovne apstraktne klase koje rezultuju pozivima metoda izvedenih klasa.
- **test/generator/generator_C3.mj**
    - Predstavlja test osnovne gramatike (bez modifikacija za junsko-julski rok) za **nivo C** koji je dat u tekstu projektnog zadatka.
- **test/generator/generator_C4.mj**
    - Predstavlja test izmenjene gramatike (sa modifikacijama za junsko-julski rok) za **nivo C** koji je dat u specifikaciji programskog jezika MikroJava.
- **test/generator/july301.mj**
    - Predstavlja test izmenjene gramatike (sa modifikacijama za junsko-julski rok) za **nivo A** koji je dat u tekstu izmene projektnog zadatka.

- **test/generator/july302.mj**
    - Predstavlja test izmenjene gramatike (sa modifikacijama za junsko-julski rok) za **nivo B** koji je dat u tekstu izmene projektnog zadatka.
- **test/generator/test301.mj**
    - Predstavlja **javni test** za **nivo A** koji je dat kao deo projektne specifikacije.
- **test/generator/test302.mj**
    - Predstavlja **javni test** za **nivo B** koji je dat kao deo projektne specifikacije.
- **test/generator/test303.mj**
    - Predstavlja **javni test** za **nivo C** koji je dat kao deo projektne specifikacije.

Ostali testovi se odnose na testiranje pojedinih delova kompajlera i stoga uglavnom nisu semantički ispravni (gotovo svi testovi za **parser** i određeni **semantički** testovi) ili imaju logičkog smisla pa njihovo ponašanje nije detaljnije opisano. Oni su pak ključni u procesu testiranja kompajlera zato što omogućavaju lako uočavanje posledica izmene određenih delova kompajlera. Njihovo ponašanje je donekle opisano komentarima koji se nalaze u izvornim kodovima testova pa se tako može detaljnije promatrati.

Kratak opis novouvedenih klasa:

Novouvedene klase su smeštene u odgovarajuće pakete pa tako postoje sledeći paketi:

- **rs.ac.bg.etf.pp1.ast**
    - Sadrži sve generisanje klase koje predstavljaju čvorove apstraktnog sintaksnog stabla. Ove klase su generisane prilikom generisanja sintaksnog analizatora i koriste se za semantičku analizu kao i za generisanje koda.
- **rs.ac.bg.etf.pp1.exceptions**
    - Sadrži klase izuzetaka koji se koriste u raznim delovima kompajlera. Osnovna klasa svih klasa izuzetaka koji se koriste u kompajleru jeste klasa **MJCompilerException**.
    - Klase **MJParserException**, **MJSemanticAnalyzerException** i **MJCodeGeneratorException** su izvedene iz klase **MJCompilerException** u cilju jednostavnog „hvatanja“ svih izuzetaka koje kompajler može da „baci“.
- **rs.ac.bg.etf.pp1.helpers**
    - Sadrži pomoćne klase koje se koriste u raznim delovima kompajlera.
    - Klasa **MJConstants** sadrži globalne konstante koje se koriste u kompajleru.
    - Klasa **ActualParametersStack** se koristi prilikom semantičke analize kako bi se prilikom poziva metoda proveravali stvarni parametri (argumenti) funkcije/metode koja se poziva.

- Klasa **JumpAddressStack** se koristi prilikom generisanja koda kako bi se na jedno mesto smestile liste adresa skokova koje treba prepraviti u određenom bloku koda.

- **rs.ac.bg.etf.pp1.loggers**

  - Sadrži klase koje apstrahuju pristup **log4j** logger-u kako bi se jednostavnije i na jednom mestu smeštale poruke za razne informacije i greške. Osnovna klasa iz koje su pojedine klase logger-a izvedene jeste klasa **MJLogger**. Ona sadrži zajedničke metode za logovanje i pruža jednostavan API za ispisivanje poruka.

  - Klase **MJLexerLogger**, **MJParserLogger**, **MJSemanticAnalyzerLogger** i **MJCodeGeneratorLogger** su izvedene iz klase **MJLogger** i svaka sadrži sopstvenu podklasu **MessageType** koja govori logger-u koju poruku treba da ispiše. Sve ove klase imaju svoje poruke koje imaju i različite parametre i stoga je korišćen parametar metoda tipa Object… **context** koji omogućava slanje 0 ili više parametara određenoj metodi. Na osnovu tipa poruke logger iz **context**-a preuzima odgovarajuće parametre i kreira poruku.

- **rs.ac.bg.etf.pp1.mj.runtime**

  - Sadrži klasu **MJCode** koja je izvedena iz klase **Code**. Ona uvodi dodatne metode koji olakšavaju generisanje koda, ali i omogućavaju bolje logovanje grešaka.

- **rs.ac.bg.etf.pp1.symboltable.concepts**

  - Sadrži klase **MJScope**, **MJSymbol** i **MJType** koje su izvedene iz klasa **Scope**, **Obj** i **Struct** respektivno.

  - Klasa **MJScope** sadrži enum **ScopeID** i polje **id** tog tipa koji se koristi za opis trenutnog otvorenog opsega u tabeli simbola.

  - Klasa **MJSymbol** uvodi nekoliko novih polja koji se koriste u različitim delovima kompajlera. Polje **parent** služi za pamćenje roditeljske klase iz koje je nasleđeno polje ili metoda. Ono olakšava generisanje virtuelne tabele metoda za izvedene klase. Polje **abstract_** služi za deklarisanje **apstraktne klase** (simbol koji sadrži tip klase ima ovo polje postavljeno na **true**) kao i za deklarisanje **apstraktnih metoda** apstraktne klase. Polje **access** tipa enum-a **Access** se koristi za članove klasa kako bi opisalo prava pristupa simbolu. Konačno, polje **readOnly** se koristi u semantičkoj analizi za zabranu upisa u simbol iteratora u telu foreach petlje.

  - Klasa **MJType** menja implementacije metoda **compatibleWith** i **assignableTo** i uvodi novi metod **isDescendantOf** koji služi za proveru da li je neka klasa roditeljska klasa trenutnoj u hijerarhiji klasa.

- **rs.ac.bg.etf.pp1.symboltable.visitors**

  - Sadrži klasu **MJDumpSymbolTableVisitor** koja je izvedena iz klase **DumpSymbolTableVisitor**. Ona modifikuje način štampanja tabele simbola kako bi dodala novouvedena polja i bolje prikazala stanje u tabeli simbola.

- **rs.ac.bg.etf.pp1.symboltable**
    - Sadrži klasu **MJTable** koja implementira tabelu simbola korišćenjem
- **rs.ac.bg.etf.pp1.test**
    - Sadrži prethodno opisane klase koje se koriste za automatsko testiranje rešenja. Osnovna klasa **MJTest** sadrži zajedničke metode za izvršavanje testova.
    - Klase **MJLexerTest**, **MJParserTest**, **MJSemanticTest** i **MJCodeGenTest** implementiraju metode **testName** i **processTestFile** i time definišu ponašanje odgovarajućeg testa.
- **rs.ac.bg.etf.pp1.util**
    - Sadrži tri klase koje apstrahuju različite delove programa.
    - Klasa **CLIUtils** se koristi za obradu argumenata kompajlera i na osnovu njih postavlja odgovarajuća statička polja u klasi **Compiler**.
    - Klasa **Log4JUtils** se koristi za učitavanje konfiguracije logger-a kao i za preimenovanje postojećih log fajlova kako bi se oni sačuvali prilikom svakog novog pokretanja kompajlera.
    - Klasa **MJUtils** sadrži razne statičke metode koje se koriste u različitim delovima kompajlera.
- **rs.ac.bg.etf.pp1**
    - Osnovni paket sadrži tri klase u skladu sa specifikacijom projekta koje izvršavaju odgovarajuće poslove.
    - Klasa **Compiler** preuzima argumente sa komandne linije korišćenjem metoda klase **CLIUtils** i pokreće kompajliranje izvornog MikroJava koda.
    - Klasa **SemanticAnalyzer** prolazi kroz generisano sintaksno stablo i kreira i stavlja simbole u tabelu simbola kako bi se koristili prilikom generisanja koda. Osim toga proverava sve neophodne kontekstne uslove u skladu sa specifikacijom.
    - Klasa **CodeGenerator** koristi prethodno inicijalizovanu tabelu simbola da generiše izvršni objektni kod ponovnim prolaženjem kroz sintaksno stablo.