Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра «Информатики и защиты информации»

Курсовая работа на тему:

Разработка компилятора подмножества процедурного языка в ассемблер

Специальность: 10.05.04 – Информационно-аналитические системы безопасности

КОРОЛЕВА Александра Александровна,
ст. гр. ИСБ-118

г. Владимир 2021

Введение

Цель работы:

разработать компилятор подмножества процедурного языка в ассемблер.

Стадии разработки:

- Описание языка
- Контекстно-свободная грамматика
- Лексический анализатор
- Таблица символов
- Синтаксический анализатор
- Генерация промежуточного кода
- Трансляция в целевой код

Описание компилятора и языка

Компилятор реализован на Java с использованием библиотек Antlr4 и ASM. Основная функция компилятора – проверка принадлежности исходной цепочки входному языку и генерация выходной цепочки символов в виде байт-кода для JVM.

Разработанный язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполняет исполнитель под ее управлением.

Ключевые слова языка:

int, float, char, bool, true, false, print, if, else, while, break, continue, const, and, or.

А также прочие символы наподобие “{}”, “=“, “>” и т.д.

КС-грамматика

КС-грамматика – частный случай формальной грамматики, у которой левые части всех продукций являются одиночными нетерминалами.

На основе построенной КС-грамматики с помощью библиотеки Antlr4 генерируется синтаксическое дерево, которое по грамматике заполняется элементами подмножества входного языка..

```
grammar cring;

program
    : function? mainProg EOF
    ;

function
    : type FUNC IDENTIFIER OPAR pa
    ;

mainProg
    : MAIN block
    ;

block
    : OBRACE (statement)* CBRACE
    ;
```

```
statement
    : varDeclaration SCOL
    | callMethod SCOL
    | expression SCOL
    | print SCOL
    | ifStatement
    | assign
    | whileStatement
    | forStatement
    | breakStat SCOL
    | contStat SCOL
    | retStat SCOL
    ;
```

Лексический анализатор

Лексический анализ (“токенизация”) – процесс аналитического разбора входной последовательности символов на распознанные группы – лексемы – с целью получения на выходе идентифицированных последовательностей, называемых “токенами”.

Лексический анализатор реализован при помощи библиотеки Antlr4.

```
private static String[] makeLiteralNames() {
    return new String[] {
        null, "','", "''''", "'.'", "'string'", null, null, null, "'=='", "'!='",
        "'>'", "'<'", "'>='", "'<='", "'+'", "'-'", "'*'", "'/'", "'%'", "'^'",
        "'!'", "';'", "'='", "'('", "')'", "'{'", "'}'", "'true'", "'false'",
        "'for'", "'if'", "'else'", "'while'", "'print'", "'function'", "'main()'",
        "'break'", "'continue'", "'return'", "'int'", "'char'", "'float'", "'bool'"
    };
}
```

Синтаксический анализатор

```
private static String[] makeRuleNames() {
    return new String[] {
        "program", "function", "mainProg", "block", "statement", "varDeclaration",
        "callMethod", "print", "ifStatement", "assign", "whileStatement", "forStatement",
        "forInside", "forInit", "breakStat", "contStat", "retStat", "parametr",
        "parametrs", "parametrList", "expression", "expressionList", "literal",
        "intLiteral", "charLiteral", "floatLiteral", "boolLiteral", "type", "varname"
    };
}
```

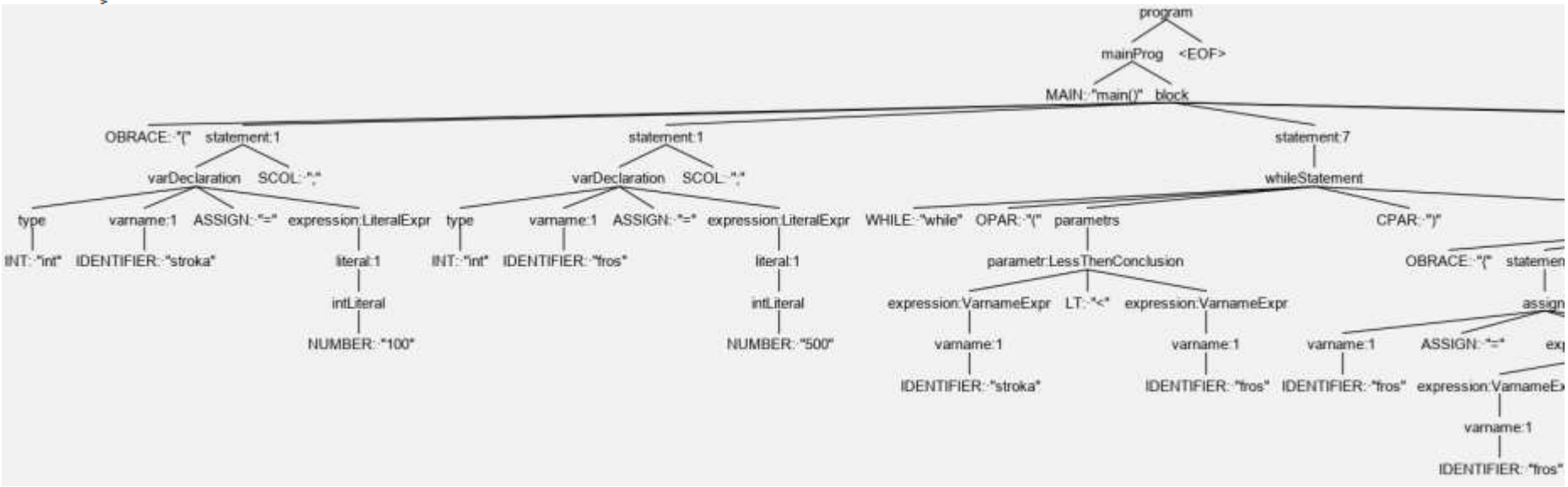

Таблица символов

Таблица символов реализована в классе, где переменная является ключом, а значением – ее тип.

Генерация кода и трансляция в целевой код

Генерация промежуточного кода и его трансляция в целевой код реализована с помощью библиотеки ASM для каждого элемента дерева. При помощи функций библиотеки можно записывать нужные инструкции в выходной поток при обходе дерева.

При генерации промежуточного кода выходные данные синтаксического анализатора преобразуются в Java Bytecode, который позднее будет обработан интерпретатором в JVM.

Выходные данные из синтаксического анализатора передаются в транслятор, который продолжает обработку. Итогом является компилируемый файл “название_файла.class”, в который записывается Java Bytecode.

В созданном файле находится байт-код, или машинный код – т.е. набор команд (кодов операций), выполняемых виртуальной машиной (JVM).

Пример работы компилятора

```
main() {
    int stroka = 100;
    int fros = 500;

    while (stroka < fros) {
        fros = fros - 2;
        break;
    }
    print(fros);
}
```

```
public class cring {
    public cring() {
    }

    public static void main(String[] var0) {
        byte var1 = 100;

        int var2;
        for(var2 = 500; var1 <= var2; var2 -= 2) {
        }

        System.out.print(var2);
    }
}
```

```
E:\IntelliJ IDEA Community Edition 2019.2.4\Kurs>java cring
98
```

Спасибо за внимание!

КОРОЛЕВА Александра Александровна,
ст. гр. ИСБ-118