SPRD

Team 12

Dipak Purbey
Prashanth Murali
Rajiv Prathapan
Sanjana Vasudevan

Introduction

- Statically typed language
- Procedural language with sequential execution of steps
- No semicolons, no indentations
- Curly braces used to separate looping and decision statements
- Handles priority of operators and associativity

Challenges faced

- Enhancements to the design of the language
- Learning the syntax for Antlr
- Need of improvisation in syntax to make it user friendly
- Keeping track of line for looping and branching to write intermediate code
- Handling nested conditional statements and looping statements

Tools

1. The compiler and runtime can be built on MacOS/Windows/Linux/Unix.

2. Grammar was a .g4 file based on Antlr.

3. Antlr v4.7 was used to generate the parser and lexer from the grammar.

4. Intermediate code was generated from implementing the generated Listener interface.

5. Python was used to write the runtime environment.

Language Capabilities

- Variables: Statically typed. Used to store values on assignment
- Types of variables: Boolean, Integer
- Operators:
  - *Arithmetic: +, -, *, /, %*
  - *Relational: <, >, <=, >=, ==, !=*
  - *Logical: and(&&), or(||), is*
  - *Assignment: =*
- Operations:
  - *While-loop*
  - *if-else*

Grammar

- Grammar follows Extended Backus-Naur form (EBNF).
- It is written as a .g4 file using Antlr.
- For example:

```
program : (intDeclaration | boolDeclaration | intAssignment | boolAssignment |
whileLoopConstruct | ifElseConstruct | printFunc)*;
```

Intermediate code

- BEGIN – Start of program
- WRITE – Write the value following the keyword to a stack
- SAVEINT/SAVEBOOL – Saves the variable to symbol table with the value popped from the stack or default value.
- LESSER/ GREATER/ LESSEROREQUAL/ EQUALS/ GREATEROREQUAL/ NOTEQUALS – Arithmetic comparison keywords.
- ADD/ SUBTRACT/ MULTIPLY/ DIVIDE/ MODULUS – Arithmetic operation keywords.
- AND/ OR/ EQUALS- Boolean comparison keywords

Intermediate Code contd..

- IFTRUEGOTO – If the condition is true, go to a particular line
- IFFALSEGOTO – If the condition is false, go to a particular line
- GET – Get value from user
- PRINT- Print the value following the print keyword
- END- End of the program

Runner

- This file aggregates the steps required to generate the intermediate file
- Intermediate file is named as “<filename>.isprd”
- It also generates the Abstract Syntax Tree where the parsing is displayed as a tree

Runtime:

- Runtime was written in Python
- Each line from the intermediate file is read
- Based on the keyword, appropriate function is called to do the necessary actions
- A stack is used for implementation of operations
- A symbol table which is a dictionary keeps track of all the values of the variables.

Example:

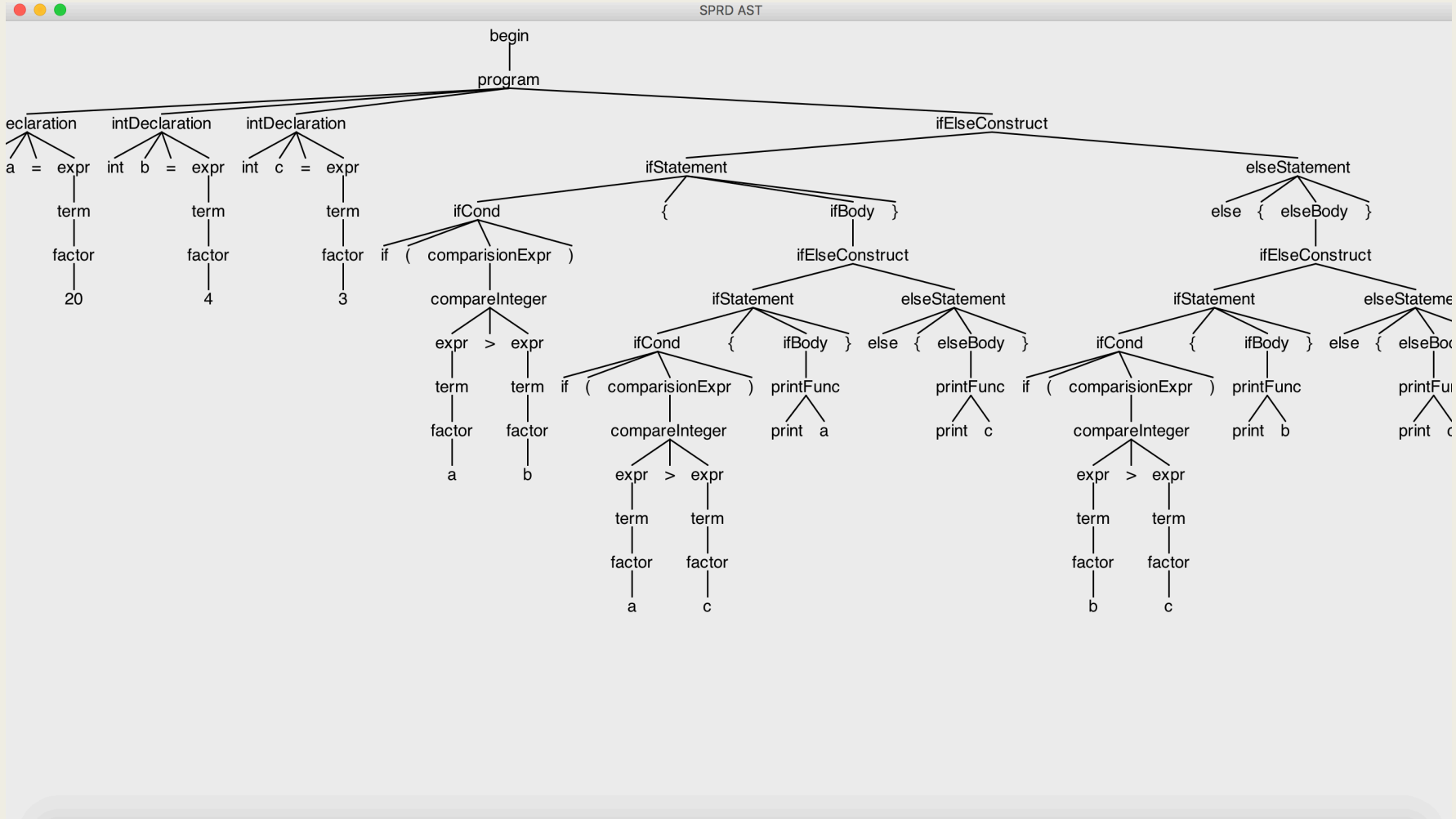
- Find the largest of three numbers

```
int a = 20
int b = 4
int c = 3
if(a>b){
    if(a>c){
       print a
    }
    else{
        print c
    }
}else{
    if(b>c){
        print b
    }
    else{
        print c
    }
}
~
```

Steps to execute

- Download the Compiler.jar file and Interpreter.py file from executable directory of git
- Run: java –jar Compiler.jar “filename.sprd”
- This will generate Abstract Syntax Tree and Intermediate code into file “filename.isprd”
- Run: python Interpreter.py “filename.isprd”
- Output of the program will be obtained.

Abstract Syntax Tree:



Intermediate code:

```
BEGIN
WRITE 20
SAVE a
WRITE 4
SAVE b
WRITE 3
SAVE c
WRITE a
WRITE b
GREATER
IFFALSEGOTO 22
WRITE a
WRITE c
GREATER
IFFALSEGOTO 19
PRINT [a]
WRITE TRUE
IFTRUEGOTO 20
PRINT [c]
WRITE TRUE
IFTRUEGOTO 30
WRITE b
WRITE c
GREATER
IFFALSEGOTO 29
PRINT [b]
WRITE TRUE
IFTRUEGOTO 30
PRINT [c]
END
```

Output from runtime:

- Run : python Interpreter.py largest.isprd

- Output:

20

References

- http://www.antlr.org/
- https://github.com/antlr/antlr4/blob/master/doc/getting-started.md
- http://stackoverflow.com/questions/30128961/trouble-setting-up-antlr-4-ide-on-eclipse-luna-4-4


Thank
you