Programming Languages

Homework Assignment 1

Announced: 2/15/2016

Due: Tuesday, 3/1/2016 5pm

Submit via Blackboard

Intro

- **The assignment is to:**
  - **Design language grammar based on provided language description/examples**
  - **Use the grammar to implement an interpreter for that language using ANTLR**
    - **(http://www.antlr.org/)**

Grammar

- **Design**
  - **syntax of the language grammar in BNF**
  - **lexical rules using regular expressions**
- **for the language as specified on the following pages**

- **Present the syntax and lexical rules as Word Document**

- **Use it to implement an interpreter that can take a program, check its correctness, and execute it**

The Language

- Language for a string manipulation. Example program:

```
var y, z string = “this”, “a” // y=“this”, z=“a”
var b int; //initialized to b=0
x:=“this” // type of x is inferred to be string
b=(6-2-1)*(-1)*-2; // b becomes 6
var c = 3+2 // type of c is inferred to be int
var v string = x+y;
print v // prints “thisis” and new line
v=v-c // removes c characters from end of v
print v; // prints “thi”
string z=v*(b/2) // repeats string v (b/2) times
print z //prints “thithithi”
print b/2 //prints “3”
z=z+2 // removes 2 characters from start of z
print z+2 // prints “ithithi”
```

The Language

- Variables
  - Variable name can be any alphanumeric string starting with a letter (not a digit)
- Variables are of two types
  - Integer (can be negative)
    - Integer literals: Values are defined using digits 0-9 and optionally “-” e.g.: 123 -1 0 etc.
  - String (of arbitrary length, including empty string “”)
    - String literals: Values are any alphanumeric characters in double quotes (“) e.g. “abc0123”, “” etc.

The Language

- Variable declaration (single variable):
  - Starts with keyword “var”
  - Followed by variable name
  - Followed by variable type
  - Followed (optionally) by “=“ and variable initial value (a valid expression: see later)
- E.g.:
  - `var x int  // x becomes 0`
  - `var x int = 1 // x becomes 1`
  - `var x string // x becomes “” i.e. empty string`
  - `var x string = “abc” // x becomes “abc”`

The Language

- Variable declaration (multiple variables of the same type):
  - Starts with keyword “var”
  - Followed by X variable names separated by commas
  - Followed by variable type
  - Followed (optionally) by X variable initial values separated by commas
- E.g.:
  - `var x,y int  // x becomes 0, y becomes 0`
  - `var x,y int = 1 , 2+4*5-2-2 // x=1, y=18`
  - `var x,y int = 1 // ERROR: needs two init values`
  - `var x,y,z string = “abc”, “def”, “geh”`

The Language

- Variable declaration (inferred type):
  - name := value
- E.g.:
  - `x := 1 // equivalent to: var x int = 1`
  - `x := “abc” //equivalent to: var x string = “abc”`

The Language

- Integer and string arithmetic:
  - Standard integer arithmetic (with rounding down):
    - `2+5*4  // this is 22`
    - `-2 - 1 -1 // this is -4`
    - `3+(-2*2)*-1 // this is 7`
    - `7/2    // this is 3 (floating part is truncated)`
  - String concatenation:
    - `“ab”+”bc” // this gives “abbc”`
  - String repetition
    - `“ab”*3 // this gives “ababab”`
    - `0*“ab” // this gives “”`
    - `“ab”*(-1) // ERROR: this is not allowed`

The Language

- Integer and string arithmetic:
  - Substrings:
    - String + integer: starting point moved X characters right, where X is the value of the integer; if X is length of string or longer than string, returns “”
      - `“abcdef”+2 // this gives “cdef”`
      - `2+“abcdef” // this gives “cdef”`
      - `“abcdef” + 6 // this gives “”`
      - `“abcdef” +7 // this gives “”`
      - `“abcdef” -(-2) // this gives “cdef”`

The Language

- Integer and string arithmetic:
  - Substrings:
    - String - integer: ending point moved X characters left, where X is the value of the integer; if X is length of string or longer than string, returns “”
      - `“abcdef” -2 // this gives “abcd”`
      - `-2 + “abcdef” // this gives “abcd”`
      - `“abcdef” – 6 // this gives “”`
      - `“abcdef” – 8 // this gives “”`
      - `“abcdef” + (-2) // this gives “abcd”`

The Language

- Expressions:
  - Any integer/string arithmetic as described above
    - Involving any mix of literals or variables

The Language

- A program is a set of instructions
  - Instructions need to be followed by “new line” or by “;” (or both)
  - Instruction can be
    - Variable declaration (see above)
    - Printout (with new line, i.e. like Java’s println())
      - Keyword “print” followed by expression, e.g.

```
print z+3
```

    - Assignment
      - Variable name followed by expression, e.g.

```
a=“abc”+”def”+z;
```

The Language

- A program may report runtime errors:
  - In case of runtime error, the program prints the error string and stops (quits)
- Possible errors:
  - `ERROR: DIVIDE BY ZERO`
    - E.g. `a :=0; b:=1; a=b/a;`
  - `ERROR: NEGATIVE STRING MULTIPLIER`
    - E.g. `a:=-1; b:="abc"; b=b*a;`
    - Or: `a:="abc"; a=-a;`
    - Or: `a:="abc"; b:="def"; a=a-b;`

The Language

- The interpreter of the language (to be written in ANTLR) should:
  - Read in and parse a program from standard input and:
    - Print "`SYNTAX ERROR`" and exit
    - or
    - Execute the program
      - All the way till the end, and the exit
      - Or till 1st runtime error is encountered
        - » In that case, print the error (see previous slide) and exit
- See HW1-Examples.txt for examples of the expected output for a given input

Submitting the assignment

- Submit via Blackboard:
  - Document describing the syntax
    - Grading:
      - does it fully match to the language?
      - Is it unambiguous?
  - ANTLR specification/code for the fully-functioning interpreter
    - Grading:
      - Your interpreter will be tested on a number of correct and incorrect programs
        - » Those from HW1-Examples.txt
        - » And many other, longer programs