Lab 6 - Changes Made To Lab 5 and AST Output

Vensan Cabardo
CS370 - Compilers and Automata
3-11-2019

Changes from Lab 5:

Production Rule Changes:

I changed the following production rule:

- `SIMP_EXPR -> ADD_EXPR | ADD_EXPR RELOP ADD_EXPR`

To the following production rule, making it left recursive to force left to right parsing.

- `SIMP_EXPR -> ADD_EXPR | SIMP_EXPR RELOP ADD_EXPR`

I also changed the following production rule:

- `ARG_LIST -> EXPR | EXPR ‘,’ EXPR`

To the following production rule, making it right recursive:

- `ARG_LIST -> EXPR | EXPR ‘,’ ARG_LIST`

YACC Changes:

All Left Hand Sides of the productions in YACC were altered to add syntax-directed semantic action that was used to create an AST while parsing the program.
Namely:

- When a variable declaration, function declaration, parameter, block statement, expression, number, read statement, identifier, return statement, write statement, assignment statement, function call, while statement, if-then statement, if-then-else statement or a function argument is found in a program, an AST node is created.
- Any relevant information for these nodes such as type, name, value, etc. is assigned to these nodes upon them being created. If other nodes contain information that is relevant to the current node being created, those nodes are connected using the next, s1, and s2 fields.
- If a production is not creating a node and is not an operator production, it will pass the reference to any created nodes up so that they can be connected to the tree.
- A global variable myprogram stores the final state of the tree, and the contents of that tree are printed after parsing.
- This file no longer includes lex.yy.c

Makefile Changes:

- lex.yy.c is now used to compile inside the makefile, instead of being included in the YACC file.

Additions:

- **ast.h**
  - This file, added to the project, contains the declarations for the AST struct, and the enums for types, operators, and system types. It also contains the prototypes for the ASTcreatenode and the ASTprint methods.
- **ast.c**
  - This file, also added to the project, contains the implementations for the create node function as well as the print routine.

Output when run on lab6test.al:

```
Variable VOID x
Variable INT x  [ 100 ]
INT FUNCTION main
(
  VOID
)
  BLOCK STATEMENT
      Variable INT x
      BLOCK STATEMENT
        Variable INT y
        WHILE STATEMENT
          EXPR <=
           EXPR +
           IDENTIFIER x
           EXPR /
             NUMBER with value 5
             NUMBER with value 2
           EXPR -
             EXPR +
               NUMBER with value 2
               IDENTIFIER z
             NUMBER with value 5
           IF
             EXPR >=
               EXPR -
           IDENTIFIER h
           NUMBER with value 2
         EXPR -
           NUMBER with value 3
           NUMBER with value 2

     THEN
                READ STATEMENT
       IDENTIFIER x
       Array Reference [
         NUMBER with value 100
       ] end array
     ELSE
       WRITE STATEMENT
         EXPR +
           IDENTIFIER x
           Array Reference [
             NUMBER with value 100
           ] end array
           NUMBER with value 200

EXPRESSION STATEMENT
  FUNCTION CALL f
     ARG
     EXPR +
     NUMBER with value 3
     IDENTIFIER x
     Array Reference [
       IDENTIFIER x
       Array Reference [
           NUMBER with value 100
       ] end array
     ] end array
     ARG
     EXPR +
       IDENTIFIER bar
       NUMBER with value 200
     ARG
       NUMBER with value 20

     RETURN STATEMENT
     RETURN STATEMENT
EXPR +
          EXPR +
             IDENTIFIER x
             NUMBER with value 5
          NUMBER with value 7
        IF
          EXPR >
             EXPR +
               IDENTIFIER x
               NUMBER with value 10
             EXPR *
               NUMBER with value 10
               NUMBER with value 20

        THEN
        ASSIGNMENT statement
          IDENTIFIER x
          EXPRESSION STATEMENT
             EXPR !=
               IDENTIFIER x
               NUMBER with value 10
        WRITE STATEMENT
          EXPR
             EXPR OR
               EXPR AND
                 NUMBER with value 3
                 NUMBER with value 5
             EXPR AND
               NUMBER with value 1
               EXPR
                  NUMBER with value 0

VOID FUNCTION f
(
  PARAMETER INT x
  PARAMETER INT y
  PARAMETER VOID z
)
  BLOCK STATEMENT
```