Compilers

CMPT 432

Project Three - 100 points

Project

1. Projects one and two working perfectly. [-∞ if not]
2. Create an AST from the tokens or from the CST. Display it. [10 points]
3. Write a semantic analyzer that scope-checks and type-checks the AST based on the grammar found on our class web site at https://www.labouseur.com/courses/compilers/grammar.pdf and the type rules we discussed in class. [60 points]
4. Create and display a symbol table with type and scope information. [30 points]

Notes and Requirements

- Build an Abstract Syntax Tree either from re-parsing the tokens or by traversing the CST. Display it after successful lex and parse.
- Scope-check on the AST.
- While you are scope-checking, build a symbol table of IDs that includes their name, data type, scope, position in the source code, and anything else you think might be important.
- Type-check the source code using the AST and the symbol table.
  - Issue errors for undeclared identifiers, redeclared identifiers in the same scope, type mismatches, and anything else that might go wrong.
  - Issue warnings about declared but unused identifiers, use of uninitialized variables, and the presence of initialized but unused variables.
- Default to verbose output functionality that traces the semantic analysis stages, including scope checking, the construction of the symbol table, and type checking actions.
- When you detect an error, report it in helpful detail including where it was found. Remember, I consider confusing, incomplete, or inaccurate error messages a serious (and intolerable) bug.
- See examples on the next several pages for details and ideas.

Other Requirements

Create a ton of test programs that cause as many different types of errors as you can in order to thoroughly test your code. Include several test cases that show it working as well. Write up your testing results (informally) in a document in your GitHub repository.

Your code must … [−∞ if not]
- separate structure from presentation.
- be professionally formatted.
- use and demonstrate best practices.
- make me proud to be your teacher.

Labs

Labs 5, 6, and 7 focus on the components of this project, as well as the final exam.

Submitting Your Work

Make many commits to GitHub. I will not accept massive commits when I review your code. (It's −∞ if you do that.) Commit early and often.

E-mail me the URL to your private GitHub master repository if you have not done so by now. Remember to add me (Labouseur) as a collaborator. Please send this to me before the due date (see our syllabus).

Compilers

CMPT 432

Input:
```
{
   int a
   boolean b
   {
      string c
      a = 5
      b = true  /* no comment */
      c = "inta"
      print(c)
   }
   print(b)
   print(a)
}$
```

Output:
```
Program 1 Lexical Analysis
Program 1 Lexical analysis produced
0 error(s) and 0 warning(s)

Program 1 Parsing
Program 1 Parsing produced 0
error(s) and 0 warnings

Program 1 Semantic Analysis
Program 1 Semantic Analysis produced
0 error(s) and 0 warning(s)
```

```
Program 1 Concrete Syntax Tree
-------------------------------------
< Program >
-< Block >
--[ { ]
--< Statement List >
---< Statement >
----< Variable Declaration >
-----[ int ]
-----[ a ]
---< Statement List >
----< Statement >
-----< Variable Declaration >
------[ boolean ]
------[ b ]
----< Statement List >
-----< Statement >
------< Block >
-------[ { ]
-------< Statement List >
--------< Statement >
---------< Variable Declaration >
----------[ string ]
----------[ c ]
--------< Statement List >
---------< Statement >
----------< Assignment Statement >
-----------[ a ]
-----------[ = ]
-----------< Expression >
------------< Int Expression >
-------------[ 5 ]
---------< Statement List >
----------< Statement >
-----------< Assignment Statement >
------------[ b ]
------------[ = ]
------------< Expression >
-------------< Boolean Expression >
--------------[ true ]
----------< Statement List >
-----------< Statement >
------------< Assignment Statement >
-------------[ c ]
-------------[ = ]
-------------< Expression >
--------------< String Expression >
---------------[ " ]
---------------< Char List >
----------------[ i ]
----------------< Char List >
-----------------[ n ]
-----------------< Char List >
------------------[ t ]
------------------< Char List >
-------------------[ a ]
-------------------< Char List >
---------------[ " ]
-----------< Statement List >
------------< Statement >
-------------< Print Statement >
--------------[ print ]
--------------[ ( ]
--------------< Expression >
---------------[ c ]
--------------[ ) ]
------------< Statement List >
-------[ } ]
-----< Statement List >
------< Statement >
-------< Print Statement >
--------[ print ]
--------[ ( ]
--------< Expression >
---------[ b ]
--------[ ) ]
------< Statement List >
-------< Statement >
--------< Print Statement >
---------[ print ]
---------[ ( ]
---------< Expression >
----------[ a ]
---------[ ) ]
-------< Statement List >
--[ } ]
-[ $ ]
```

```
Program 1 Abstract Syntax Tree
-------------------------------
< BLOCK >
-< Variable Declaration >
--[ int ]
--[ a ]
-< Variable Declaration >
--[ boolean ]
--[ b ]
-< BLOCK >
--< Variable Declaration >
---[ string ]
---[ c ]
--< Assignment Statement >
---[ a ]
---[ 5 ]
--< Assignment Statement >
---[ b ]
---[ true ]
--< Assignment Statement >
---[ c ]
---[ inta ]
--< Print Statement >
---[ c ]
-< Print Statement >
--[ b ]
-< Print Statement >
--[ a ]


Program 1 Symbol Table
--------------------------------------
Name  Type    Scope  Line
-------------------------------------
a     int      0     2
b     bool     0     3
c     string   1     5
```

Input:
```
{
   int a
   {
      boolean b
      a = 1
   }
   print(b)
}$
```

Output:
```
Program 2 Lexical Analysis
Program 2 Lexical analysis produced
0 error(s) and 0 warning(s)

Program 2 Parsing
Program 2 Parsing produced 0
error(s) and 0 warning(s)

Program 2 Semantic Analysis
```
Error: The id b on line 7 was used before being declared.

Warning: The id a on line 2 was declared and initialized but never used.

```
Program 2 Semantic Analysis produced
1 error(s) and 1 warning(s).
```

```
Program 2 Concrete Syntax Tree
--------------------------------
< Program >
-< Block >
--[ { ]
--< Statement List >
---< Statement >
----< Variable Declaration >
-----[ int ]
-----[ a ]
---< Statement List >
----< Statement >
-----< Block >
------[ { ]
------< Statement List >
-------< Statement >
--------< Variable Declaration >
---------[ boolean ]
---------[ b ]
-------< Statement List >
--------< Statement >
---------< Assignment Statement >
----------[ a ]
----------[ = ]
----------< Expression >
-----------< Int Expression >
------------[ 1 ]
--------< Statement List >
------[ } ]
----< Statement List >
-----< Statement >
------< Print Statement >
-------[ print ]
-------[ ( ]
-------< Expression >
--------[ b ]
-------[ ) ]
-----< Statement List >
--[ } ]
-[ $ ]

Program 2 Abstract Syntax Tree
--------------------------------
< BLOCK >
-< Variable Declaration >
--[ int ]
--[ a ]
-< BLOCK >
--< Variable Declaration >
---[ boolean ]
---[ b ]
--< Assignment Statement >
---[ a ]
---[ 1 ]
-< Print Statement >
--[ b ]

Program 2 Symbol Table
not produced due to error(s) detected by
semantic analysis.
```

Input:

```
{
  int a
  {
    boolean b
    {
      string c
      {
        a = 5
        b = false
        c = "inta"
      }
      print(c)
    }
    print(b)
  }
  print(a)
}$
```

Output:

```
LEX: BEGINNING LEXING PROCESS ON PROGRAM 3...
LEX: SUCCESSFULLY COMPLETED WITH (0) errors.

PARSE: BEGINNING PARSING PROCESS ON PROGRAM 3...
PARSE: PARSE PROCESS SUCCESSFULLY COMPLETED.

CST: Printing Program 3:
 <Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<ID>
------[a]
---<StatementList>
----<Statement>
-----<Block>
------[{]
------<StatementList>
-------<Statement>
--------<VarDecl>
---------<Type>
----------[boolean]
---------<ID>
----------[b]
-------<StatementList>
--------<Statement>
---------<Block>
----------[{]
----------<StatementList>
-----------<Statement>
------------<VarDecl>
-------------<Type>
--------------[string]
-------------<ID>
--------------[c]
-----------<StatementList>
------------<Statement>
-------------<Block>
--------------[{]
--------------<StatementList>
---------------<Statement>
----------------<Assign>
-----------------<ID>
------------------[a]
-----------------[=]
-----------------<Expr>
------------------<IntExpr>
-------------------<Digit>
--------------------[5]
---------------<StatementList>
----------------<Statement>
-----------------<Assign>
------------------<ID>
-------------------[b]
------------------[=]
------------------<Expr>
-------------------<BooleanExpr>
--------------------<BoolVal>
---------------------[false]
----------------<StatementList>
-----------------<Statement>
------------------<Assign>
-------------------<ID>
--------------------[c]
-------------------[=]
-------------------<Expr>
--------------------<StringExpr>
---------------------["]
---------------------<CharList>
----------------------<Char>
-----------------------[i]
----------------------<CharList>
-----------------------<Char>
------------------------[n]
-----------------------<CharList>
------------------------<Char>
-------------------------[t]
------------------------<CharList>
-------------------------<Char>
--------------------------[a]
-------------------------[CharList]
---------------------["]
-----------------[StatementList]
--------------[}]
------------<StatementList>
-------------<Statement>
--------------<Print>
---------------[print]
---------------[(]
---------------<Expr>
----------------<ID>
-----------------[c]
---------------[)]
-------------[StatementList]
----------[}]
--------<StatementList>
---------<Statement>
----------<Print>
-----------[print]
-----------[(]
-----------<Expr>
------------<ID>
-------------[b]
-----------[)]
---------[StatementList]
------[}]
----<StatementList>
-----<Statement>
------<Print>
-------[print]
-------[(]
-------<Expr>
--------<ID>
---------[a]
-------[)]
-----[StatementList]
--[}]
-[$]
```

```
SEMANTIC: STARTING SEMANTIC ANALYSIS ON PROGRAM 3.
SEMANTIC: SEMANTIC ANALYSIS SUCCESSFULLY COMPLETED ON PROGRAM 3.

AST: Printing AST for Program 3:
 <Block>
-<VarDecl>
--[int]
--[a]
-<Block>
--<VarDecl>
---[boolean]
---[b]
--<Block>
---<VarDecl>
----[string]
----[c]
---<Block>
----<Assign>
-----[a]
-----[5]
----<Assign>
-----[b]
-----[false]
----<Assign>
-----[c]
-----["inta"]
---<Print>
----[c]
--<Print>
---[b]
-<Print>
--[a]

SYMB: Printing Symbol Table for Program 3:
NAME TYPE      isINIT?   isUSED?   SCOPE
[a   int       true      true      0]
[b   boolean   true      true      1]
[c   string    true      true      2]
```