CSC 446

Assign #3
Hamer
Due: Feb 19, 2025

Create a recursive descent parser for the CFG given in the previous assignment.

You may use the test programs given in the previous assignment and those included at the end to test your parser. Be sure to realize that this is not an exhaustive test of your parser and you should develop as many test cases as you can think of.

To turn in your assignment submit both your lexical analyzer and parser programs in a zip file placed into the Assign 3 dropbox on D2L. Submit all parts of your assignment. Use proper data abstraction techniques when you write your program. This means that the parser and lexical analyzers need to be in separate source files. Include any other needed programs so that they will compile without modification. Programs written in C or C++ are required to compile and run on the machine cscssh.sdstate.edu. This is a dual processor Zeon 12 core 64 bit processor and not an Intel Pentium or AMD Ryzen processor. To test your software you may connect to cscssh.sdstate.edu using the ssh protocol and run your program there. When working on the Linux system in the lab your files are hosted on this machine, so you do not have to transfer any files. You simply open a shell prompt window and type the command “ssh cscssh.sdstate.edu” and hit enter. You will be prompted for your password and then you are in. Notice that the prompt has now changed to “username@csc-linux1”. The real name for cscssh is csc-linux1! Programs written in C# will be run under Visual Studios 2022 with the test files located in the same folder as your executable program.

To ensure that the program is indeed legal your parser must terminate with the end of file token!

Test Programs

The simplest Ada program is then:

```
PROCEDURE one IS

BEGIN

END one;
```

A more typical program would be

```
PROCEDURE MAIN IS

PROCEDURE PROC1 IS
BEGIN
END PROC1;

BEGIN
END MAIN;
```

A more complicated program would look like

```
PROCEDURE seven IS
  count:CONSTANT:=5;
  a,b:INTEGER;
 PROCEDURE eight(x:INTEGER; y:INTEGER) IS
 BEGIN
 END eight;
BEGIN
END seven;
```

Finally, you could use this program too!

```
procedure five is
 a,b,c,d:integer;
 procedure fun(a:integer; out b:integer) is
  c:integer;
 begin
end fun;
begin
end five;
```