Prikshet Sharma

Assignment 2

To compile and run:

1. Make parser
2. ./parser < test.in

Note: The test.in file contains the code similar to the one on Assignment 2 webpage but with declarations and types.

Basics

1. The code has been converted into C++. There are no printf statements and only C++ standard libraries are used.
2. The language has been expanded to include “if”, and “while” statements, as shown in the grammar on the assignment 2 webpage.
3. We have implemented **context-specific error recovery** using “first” and “follow” sets and check_for_errors function.
4. We print the syntax tree similar to the one on the Assignment 2 webpage (no indentation however).

Context-Specific Error Recovery

If the input has any syntax errors we output the message “syntax error” and also token_image of the token being deleted. Tokens are deleted until we find a token from where we can start parsing correctly again.
After displaying the “syntax error” messages and deleted token numbers, we print the syntax tree as if the deleted tokens were not present in the input. To delete the tokens we call delete_token() which simply scans a token and returns nothing.

Extra Credit

1. Context specific error recovery using First and Follow sets. The creation of these sets is implemented in the function “first_follow” in parse.cpp lines 138-190.
2. Language extended with typed variable declarations “int” and “real”, and casting with “trunc()” and “float()”. So instead of (num “7”), we will have (int_const “7”) or (real_const “7.0”), depending on whether the number has a “.” or not. This has been implemented in scan.cpp lines 41-61. Adding types also leads us to create a new symbol “decl” for declarations which can be found in parse.cpp lines 328-358.

3. Redeclaration of an “id” reports error along with the name of the id that was redeclared. See the “decl” function for the implementation.
4. We have implemented the “List” class to store the representation of the parse tree. So not only is the parse tree printable but is also flexible in case we want to add static and type checking. List class implementation can be found in parse.cpp lines 16-118.

Limitations and bugs

1. Declarations and initialization cannot be done in a single line like so: “real r := 5.0” hasn’t been implemented.
2. We haven’t implemented type checking. Doing
    - real a
    - a := 3.0
    - int b
    - b := 4
    - real c
    - c := a + b, is valid though it shouldn’t be because “b” hasn’t been casted to a real.
3. We haven’t implemented indentation when printing the syntax tree.
4. The syntax tree has a space after every literal and id, as in “(int_const “9”)”, even though we don’t want that.
5. If we have a syntax error that looks like “c := ” without a literal assigned to “c”, we incorrectly include the list (:= c) in the syntax tree. Fortunately this syntax error is still reported during error recovery.
6. We might at times get a cascade of error messages that refer to the same syntax error.