Implementation of SnuPL/2 Scanner (Phase 1)

2019-10562 O Seungtaek 오승택

1. Modifying token list

EToken, ETokenName, ETokenStr were modified to fit the SnuPL/2 specification. As digits and letters are not tokens by themselves anymore, being part of tNumber and tIdent respectively, they were removed. Meanwhile, reserved keywords and necessary operators were added to those lists. I have opted to make every reserved keyword a token of its own.

2. Scanner

2-1. Comments

A new variable 'commenting' was added and a do-while loop containing most of the scanner body was created. When "//" appears, the two slashes and everything afterwards until a new line starts must be ignored as a comment. While the existing structure of scanner code will always return a NewToken after the switch statement ends, the do-while loop makes the code read the rest of the line and return to the start of Scan without returning a token.

2-2. Character Constant

If the scanner has just read a single quote, it reads the character through GetCharacter function and consumes the closing quote, which must be present following a character. If something else is there, then it returns an error.

2-3. String Constant

After reading a opening double quote, it reads characters through GetCharacter and adds them to tokval. This continues until it encounters a corresponding closing quote (which should be there) or new line, end of file or file error (which should not be there before the quote, therefore leading to error or tEOF/tIOError token respectively).

2-4. Numbers and Identifiers

Anything that does not start with set operators are assumed to be either a number or an identifier. If the token starts with a number, it is considered a number and reads the following characters if they are digits. After reading a number, it checks for an 'L' (for long int) at the end, and if there is one that is also part of the number token.

If the token starts with a letter or an underscore, it is considered an identifier and reads the following characters if they are digits, letters, or underscores. Finally, the scanner looks if the identifier matches

a reserved keyword with an iterator over map. If there is a match, token is not an identifier but a separate one for that keyword.

If the first character is not an alphanumeric or an underscore, the token is not a valid one.