Lexical Analyzer Report

121250156
Wang Tianyu
2014.11.05

A. Aiming

We want to read the input characters and produce a sequence of tokens used by the parser for syntax analysis. The terms of the lexical analyzer have three parts: token, lexeme and pattern.

B. Content Description

Build a lexical analyzer to analyse a program and show the sequence.

C. Ideas

For I did't have enough time to do this homework(I will explain the reason in Part J), I just first defined the REs and then converted them into NFAs. After that I merged these NFAs into a DFA(looking like the switch-case clause).

D. Assumptions

We assume that this program can recognise very easy program language defined by Part E. And if we meet comments, omit that.

E. Related FA Descriptions

I defined the following key words as tokens:

void, int, while, double, if, else, return, +, -, *, /, >, <, =, >=, <=, !=, ;, ,, ., (,), [,], {, }.

F. Description of Important Data Structure

- Stack rawInputLines: save the file context.
- StringTokenizer tokenedInputLines[]: save what token the char/word is.
- Object processedInput[]: a list used in processing the context.
- String[] tokenImage: save the token I defined in Part E and be used in switch-case clause to test matching the innercode(just the token number).

G. Description of Core Algorithms

- Line 93 to 523: check the DFA states.
- If we meet a comment just omit it.

H. Use Cases on Running

I used the program.txt to check the correctness.

I. Problems Occurred and Related Solution

Nope, for the DFA is very simple and easy. :)

J. Feelings and Comments

Well, this is a hard challenge to me. Because there were many tasks that I had to do assigned by the youth league committee during this month. And in fact, before I wrote this program, I had written a program using RE to NFA and NFA to DFA. But this program always crashed for the stack size. So I used the easiest way to rewrote the program.