From: Vandana Agarwal . (via Nalanda)
Sent: 28 February 2023 15:56
To: RAMAKANT PANDURANG TALANKAR .
Subject: 11551001318_CS F363_JAN_2023: Stage 1: Driver, Compilation and Execution details

11551001318_CS F363_JAN_2023 » Forums » Announcements » Stage 1: Driver, Compilation and Execution details

V

Stage 1: Driver, Compilation and Execution details
by Vandana Agarwal . - Tuesday, 28 February 2023, 3:24 PM

Driver

Your driver must have the following choices.

Press option for the defined task (Use a while loop to receive option choices till option 0 is pressed. Ensure independence of working of all options e.g. if option 3 is pressed, option 2 is not needed)

0 : For exit

1 : For removal of comments - print the comment free code on the console (Ensure that the line numbers of original code are preserved)

2 : For printing the token list (on the console) generated by the lexer. This option performs lexical analysis and prints all tokens and lexemes line number wise. Here, the tokens are not passed to the parser, but printed on the console only. Each token appears in a new line along with the corresponding lexeme and line number. (invoke only lexer)

The format for printing each token is as follows.

Line_number lexeme Token_name

Also print the lexical errors with lexemes and line numbers appropriately.

3 : For parsing to verify the syntactic correctness of the input source code and printing the parse tree appropriately. This option prints all errors - lexical and syntactic, line number wise, on the console and prints parse tree in the file as mentioned in the command line below. (Invoke both lexer and parser) .

4: For printing (on the console) the total time taken by your stage 1 code of lexer and parser to verify the syntactic correctness. Use <time.h> file as follows

---

#include <time.h>

```
clock_t start_time, end_time;

double total_CPU_time, total_CPU_time_in_seconds;

start_time = clock();

// invoke your lexer and parser here

end_time = clock();

total_CPU_time = (double) (end_time - start_time);

total_CPU_time_in_seconds = total_CPU_time /

CLOCKS_PER_SEC;

// Print both total_CPU_time and total_CPU_time_in_seconds
```

Perform actions appropriately by invoking appropriate functions.

Also the driver displays necessary information regarding implementation status of your work at the beginning on the console such as

(a) FIRST and FOLLOW set automated

(b) Only Lexical analyzer module developed

(c) Both lexical and syntax analysis modules implemented

(d) modules compile but give segmentation fault

(e) modules work with testcases 2, 3 and 4 only

(f) parse tree could not be constructed

and so on which ever is applicable.

Compilation:

The name of the make file should be makefile only as I will avoid using -f option always to make your file named something else (that includes searching for the file which is time taking). You can find documentation at the GNU website where you can learn how to write a make file (http://www.gnu.org/software/make/manual/make.html).

Please ensure compatibility with the GCC specifications provided earlier.

Execution

The command line argument for execution of the driver should be as follows, for example

$./stage1exe testcase.txt parsetreeOutFile.txt size_of_buffer

where stage1exe is the executable file generated after linking all the files (your makefile should be absolutely correct). The file testcase.txt is the source code file in the given language to be analyzed and parsetreeOutFile.txt is the file containing parse tree printed as per the format specified earlier. The size_of_buffer is the size of one buffer for twin buffer.

The inorder traversal for an n-ary tree can be described as follows

Leftmost child --> parent node--> remaining siblings (excluding the leftmost child)