Phase 1. Scanner/Screener
(Due Wednesday February 4)

J.R. Cordy – January 2015

In this phase you will undertake the modifications to the Scanner/Screener phase of the PT Pascal compiler to turn it into a Scanner/Screener for Drift. The hardest part of this phase is getting used to using Linux, PT Pascal and S/SL. The actual changes are relatively easy once you figure out how everything works.

Because the scanner/screener is run as a co-routine with the parser, it will be necessary to take some special steps to test the result of your modifications. It is suggested that for test purposes you replace the call to the Parser procedure in the main program of parser.pt with a while loop that simply repeatedly calls the AcceptSyntaxToken procedure until a pEndOfFile token is accepted. You will want to use the –t1 flag of the ptc command to run your scanner/screener and the ssltrace command to print out the tokens it recognizes – ask your TA for details.

Suggestions for Implementing Phase 1

The modification of the Scanner/Screener to implement Drift should be relatively straightforward, so no particular implementation hints will be given for this phase. You will need to change the files scan.ssl, parser.pt, and stdIdentifiers in the parser subdirectory. The following is a list of the changes that are likely to be required.

Keywords

Remove the old PT keywords:

then until do program const
procedure begin repeat

Add the new Drift keywords:

let func switch default elsif loop break module extern return (858 only)

Replace the old PT Pascal predefined identifier:

char

With the new Drift predefined identifier:

string

Character Classes

Add new Drift input character classes (e.g., ISlash) for the characters slash ("/") or bar ("|"), hash ("#") and exclamation ("!"), used in Drift. Reuse the existing IQuote character class to refer to the Drift double quote ("" ) instead of the PT Pascal single quote (’’ ). Don’t forget to update the initialization of the character class map in parser.pt.

Syntax Tokens

Add new Drift syntax tokens (e.g., pElif) for the new Drift keywords:

let, func, switch, default, elsif, loop, break, module, extern and return (858 only), and for the new Drift symbols or bar ("" ), hash ("#"), double colon ("::") and double equals ("=="). Re-use the old pNotEqual syntax token for the new Drift not equals symbol ("!='"). Remove the old PT syntax tokens for the keywords then, until, do, program, const, procedure, begin and repeat, and the PT colon equals symbol ("=").

String Literals

Replace the scanning and screening of PT single-quoted char literals (e.g., ‘hi there’) with the scanning and screening of Drift-style double quoted string literals (e.g., "hi there"). Reuse the old PT character class IQuote to represent the double quote character and reuse the old PT char literal compound token pStringLiteral to represent the new Drift double-quoted strings.

Comments

Replace the scanning of PT { } and (* *) comments with the scanning of Drift /* */ and // to end of line comments. Be careful to preserve the right number of pNewLine tokens in the scanned output when skipping // comments.

What to Hand In

Submission of your assignment will be electronic - ask your tutor for details. Hand in at least the following:

(1) Complete documentation of the modifications you have made to the scanner/screener, including a list of the changes made to the S/SL source, the specification of any semantic operations which were added or modified, a description of the changes to the token streams and tables, and an explanation of any new error signals. The documentation should be easily understandable to someone who already knows the basics of the PT Pascal compiler.

(2) A complete copy of your Drift scanner/screener S/SL source, with changes indicated by appropriate highlighting and commenting. The output of the Linux command:

diff –DNEW oldscanner.ssl newscanner.ssl

is appropriate and automatically does the highlighting for you (but source comments for your changes are still required).

(3) A suite of test inputs designed to demonstrate the (partial) correctness of the changes by forcing the scanner/screener through every new or modified logic path in the S/SL source at least once. The test suite should be accompanied by written comments indicating the purpose of each test input and an argument for the completeness of the test suite based on coverage of every modified section of the program code.

The purpose of what you hand in is to clearly indicate exactly what you have done, provide enough information about the changes to allow further maintenance of the new scanner/screener by someone who already knows the PT Pascal compiler, and to convince them that the changes you have made are complete and correct according to the spec of Drift.

Details of exact expectations and electronic submission will be given in tutorial.

See the handout “CISC 458 Project Requirements” for marking criteria and general expectations on project results.