Project 1 FAQs

Q) Which software packages are needed for this project?

A) All the required lex packages are pre-installed on the departmental elements cluster machines. For the list of elements machines, refer to the Tech FAQs. If you do not have a CS account required to login to these machines please contact tech asap. Those of you who want to work on your laptops, there should be lex distributions for all popular OSes. For windows, refer to the following link: http://gnuwin32.sourceforge.net/packages/flex.htm. Just make sure that your code works in the elements machines before submission.

Q) Which are the command lines needed to compile and run the lexer?

A) On Linux/Unix machines (assuming your lex file is lexer.l):

```
$ lex lexer.l (produces lex.yy.c)
$ gcc -o lexer -ll lex.yy.c
```

You will probably want to write a Makefile for these commands. On Windows machines, you will probably be working in an IDE so you won't need to manually compile, but if you want to do it on the command line it would be slightly different (e.g. you will probably use a different C compiler from gcc, you will not need to pass the -ll option to link in the Linux library etc.). To run the lexer with an input program 'test.mjava', you will do something like the following:

```
$ cat ./test.mjava | ./lexer
```

Your lexer by default takes the input from stdin, unless your set yyin to another file.

Q) Where do I get started? Do you have some example code other than the one on the slides?

A) The first thing you should do is to read the Lex manual posted on the website. Then you can also try reading the following link for more information on how Lex (the lexer) interactes with Yacc (the parser) and how to pass tokens: http://www.tldp.org/HOWTO/Lex-YACC-HOWTO-6.html Of course, the code is not directly usable since we require things like string tables but it will give you a good idea on how to start.

Q) For string constants, do we need to store the starting and ending quotes too in the string table?

A) No. If you look at the example string table output from the assignment, you will notice that there are no quotes around Hello World. The string that stored in the string table is going to be the value, or token attribute, of that token. That token attribute, in the end, is going to be used when generating code. For example, the token attribute for an integer constant is going to be used to actually embed that integer into the code. The token attribute for a string constant is going to be similarly embedded. Now when you execute that example program, you expect to see the output Hello World !!!, not 'Hello World !!!'. Therefore you have no need to embed those quotes in the final binary.

Q) For string constants, what do we do with escape sequences?

A) The same logic as the above applies. You want to store the string as it would get embedded in the code during code generation. So if you see the escape sequence "\n" in the string, you want to convert that to the actual newline character ('\n' or 0xA in ASCII code). That's what you want to generate, and that's what you want to get output when you print the string.

Q) In the project description, it is mentioned that we need to write a routine for reporting error. Does this mean that we need to write a separate function that will be called whenever there is an error and will print the error, the line no and column no.? Or, is it okay to just print the error, line no and column no. in the rules section for every matched incorrect token?

A) Yes it asks you to write a function ReportError with a message as a parameter. It is meant to make things convenient for you, since all error reporting involves doing the same thing.. reporting the line number and column number along with an error message. You could copy and paste that piece of code for every rule for an incorrect token but it wouldn't be good coding practice.

Q) How do we detect the EOF (end-of-file) token?

A) There is a special regular expression <<EOF>> that you can use to detect the EOF character in the character input stream. You just need to return EOFNum when that regular expression is detected. Try doing 'info flex' on the commandline to see a list of all regular expressions possible in flex.

Comments

You do not have permission to add comments.