Compiler Project 2

Project: C-- compiler syntactic analyzer

Date: November 14, 2018

Platform: NCTU CS FreeBSD

How to run:

1. Use make to compile the parser

```
OpenSSH SSH client
bsd4 [/u/cs/105/0516076/2018/compiler_hw2] -wlshih1214- make
```

2. Execute the parser with testfile(t_0)

```
OpenSSH SSH client
bsd4 [/u/cs/105/0516076/2018/compiler_hw2] -wlshih1214- make
yacc -v -d yacc.y
lex lex.l
gcc -o parser lex.yy.c y.tab.c -lfl -ll -ly
bsd4 [/u/cs/105/0516076/2018/compiler_hw2] -wlshih1214- ./parser t_0
```

3. You will get the result

```
<delim:)>
<delim:{>
36:     for(i=0; i<n; i=i+1){

<KW:break>
<delim:;>
37:             break;

<delim:}>
38:     }

<delim:}>

|--------------------------------|
|  There is no syntactic error!  |
|--------------------------------|
```

4. If parsing is not successful, the parser will print the error line and the unmatch token.

```
|-----------------------------------------------------------------------
| Error found in Line #1: int main = 0;
|
| Unmatched token:
|-----------------------------------------------------------------------
```

Abilities:

With the help of Yacc compiler-compiler, the syntactic analyzer (parser) will parse the c-- grammar, which is LALR(1), and check if the test file is a valid c–program.

Other:

I used the previous scanner for lexeme scanning. To be compatible with the parser, compare to the previous version, return value is added at the end of each token definition.

```
">" {token(">"); return GT;}
"==" {token("=="); return EQ;}
"!=" {token("!="); return NE;}
"&&" {token("&&"); return AND;}
"||" {token("||"); return OR;}
```

The parser will call lex everytime the next token is needed. Each time a lexeme is scanned, scanner will return a value to the parser which is defined in y.tab.h.