lex scanner of pascal

1. **lex version:** flex 2.6.4
2. **OS:** Ubuntn 22.04 server installed on VM in MacOS Sonoma
3. **Implementation method:**
   - Creating a makefile to compile a lex file and the final executable file.

```
all:
    flex b096060041.l
    gcc lex.yy.c -o b096060041.out -lfl
clean:
    rm -rf lex.yy.c b096060041.out
```

**note: You should install 'flex' and 'gcc' in your OS before execute `make`. Here is a simple example install package and execute makefile on ubuntu 22.04.*

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install flex gcc
$ make all
$ make clean
```

   - Writing a bash script to test all pascal files at a time. this is a simple bash script named 'exec.sh'.

```
#!/usr/bin/bash

echo '| test1.pas |'
cat testdata/1.pas;echo
./b096060041.out < testdata/1.pas

echo '------------------------'
echo '| test2.pas |'
cat testdata/2.pas;echo
./b096060041.out < testdata/2.pas

echo '------------------------'
echo '| test3.pas |'
cat testdata/3.pas;echo
./b096060041.out < testdata/3.pas

echo '------------------------'
echo '| test4.pas|'
```

```
cat testdata/4.pas;echo
./b096060041.out < testdata/4.pas

echo '-----------------------'
echo '| test5.pas |'
cat testdata/5.pas;echo
./b096060041.out < testdata/5.pas

echo '-----------------------'
echo '| test6.pas |'
cat testdata/6.pas;echo
./b096060041.out < testdata/6.pas

echo '-----------------------'
echo '| test7.pas |'
cat testdata/7.pas;echo
./b096060041.out < testdata/7.pas
```

4. **How do you address the issue in the specification?** There have 7 requests: reserved words, identifiers, symbols, real and integer number, quoted string, comments and the error recorver. I dealt with the 7 requests of the specification mentioned above by regular expression(abbreviated regex below). If one of tasks have done, then I will take the next one until all the lex code could be corresponding to the pascal scanner, which is a process trial and error.

    - *reserved word*: Using `|` to define all the the reserved words of pascal on the top of defination area. Moreover, using `%option case-insensitive` to ignore the letter case. Noticing that the "reserved word" in this assignment not cover all the reserved words in pascal and having defining some NOT pascal's reserved words such as `float`. That is, this lex scanner of pacsal only for the assignment, which could not cover all the pascal code.
    - *identifier*: The rule of identifier is starting with letter or underline, which is `_`, and check whether it is valid or invalid. Moreover, the length of identifier needs to less than 15 characters. Last but not least, recording them in symbol table(lex do not generate the symbol table automatically; however, just only have to defined a funtion to record them).
    - *symbol*: The method like reserved word, enumerating all symbols and sepearating them by `|`. In this assignment, I defined the operators addtionally.
    - *real and integer number*: Using regex. Worth noticing that the `+` and `-` have to double check that whether it is an operator or an unary plus/minus. I defined the rule of `{integer}{operator}` and used the C code in user code area to distingish them.
    - *quoted string*: The quoted strings must start and end with the `'`. The length needs to less than 30 characters as well(not includeing the `'` symbols). In real pascal code, `'3ab;` is seen as the invalid string, `ab;`. Yet, in this assignment, the `;` have to be seen as "the other symbol". Hence, the input of `'3ab;` will have the output below:

      ```
      yen@yenubuntu:~/lex$ ./b096060041.out
      '3ab;
      Line: 1, 1st char: 1, "'3ab" is an "invalid string"
      Line: 1, 1st char: 5, ";" is a "symbol"
      ```

In addtion, `'a'ab` is a lex error, the middle `'` should be typed as escape symbol `''`. In pascal compiler, it will return the error code:

```
syntax error, ";" expected but "identifier ab" found.
```

but in the lex of this assignment, it will returns:

```
Line: 1, 1st char: 1, "'a'" is a "string"
Line: 1, 1st char: 4, "ab" is an "id"
Line: 1, 1st char: 6, "'" is an "invalid string"
```

which is a lex error.

- *comments*: The context between `(*` and `*)` will be seen as a comment including the cross lines comments. That is, `(* a**b) *)` and `(*****)` are correct comments, `(*ab*)**)` is an incorrect comment.

> note:
>
> 1. The newline symbol in unix based system is "CR", which is `\n`; however, in Windows is "CR LF", which is `\r\n`.
> 2. Before testing, you should notice the encodig. You could use `dos2unix` or `unix2dos` to change the encoding in unix based system. (You still have to install it before use, typing `sudo apt install dos2unix` in Debian based unix, `sudo dnf install dos2unix` or `sudo yum install dos2unix` in Rad Hat based unix.

- *error recorver*: The implementation was Check the Yacc Scanner for Lexical Rules, not the Yacc Parser that checks Grammar Rules. Hence, if it is a valid token, we should print it out although it has the error grammer. For instance: `int x = 2 + 0.3f ;`.
    1. Unrecognized characters will be output separately as unrecognized characters.

    ```
    . { printf("Line: %d, 1st char: %d, \"%s\" is an \"invalid character\".\n", lineCount, charCount, yytext); charCount++;
    }
    ```

    2. `''` and `''''`, in lex are correct tokens and be seen as invalid strings; yet, the grammer are error.
    3. valid real such as 1.0, 1.00, 1.0000000, 00003.0000, 00134.1000, 00000.
    4. invalid real such as .123, .000123, 1., 022., .123

5. **What problems did you encounter when working on the assignment?** The main problems I faced with were dealing with the strings and comments, which was not an easy way to use regular expression to fix the scanner of pascal.

6. **The result of test:**

- personal pascal test sample

  1. test.pas

```
yen@yenubuntu:~/lex$ cat test.pas
program hello;
begin
    i := 000000
    ''
    ''''
    'a'ab'
    '\''
    writeln('hello world')
end
yen@yenubuntu:~/lex$ ./b096060041.out < test.pas
Line: 1, 1st char: 1, "program" is a "reserved"
Line: 1, 1st char: 9, "hello" is an "id"
Line: 1, 1st char: 14, ";" is a "symbol"
Line: 2, 1st char: 1, "begin" is a "reserved"
Line: 3, 1st char: 2, "i" is an "id"
Line: 3, 1st char: 4, ":=" is a "symbol"
Line: 3, 1st char: 7, "000000" is an "integer"
Line: 4, 1st char: 2, "'" is an "invalid string"
Line: 4, 1st char: 2, "'" is an "invalid string"
Line: 5, 1st char: 2, "''''" is a "string"
Line: 6, 1st char: 2, "'a'" is a "string"
Line: 6, 1st char: 5, "ab" is an "id"
Line: 6, 1st char: 7, "'" is an "invalid string"
Line: 7, 1st char: 2, "'" is an "invalid string"
Line: 7, 1st char: 2, "\" is a "symbol"
Line: 7, 1st char: 3, "'" is a "symbol"
Line: 7, 1st char: 4, "'" is a "symbol"
Line: 8, 1st char: 2, "writeln" is a "reserved"
Line: 8, 1st char: 9, "(" is a "symbol"
Line: 8, 1st char: 10, "'hello world'" is a "string"
Line: 8, 1st char: 23, ")" is a "symbol"
Line: 9, 1st char: 1, "end" is a "reserved"

The symbol table contains:
hello
i
ab
yen@yenubuntu:~/lex$
```

  2. commenttest.pas

```
yen@yenubuntu:~/lex$ cat commentstest.pas
(* comment line 1
   comment line 2
   comment line 3 *)
// comment line 4
```

```
begin
    3i := 'abc'
end
yen@yenubuntu:~/lex$ ./b096060041.out < commentstest.pas
Line: 1, 1st char: 1, "(* comment line 1\n   comment line 2\n
comment line 3 *)" is a "comment".
Line: 4, 1st char: 1, "// comment line 4" is a 'comment'
Line: 5, 1st char: 1, "begin" is a "reserved"
Line: 6, 1st char: 2, "3i" is an "invalid id"
Line: 6, 1st char: 5, ":=" is a "symbol"
Line: 6, 1st char: 8, "'abc'" is a "string"
Line: 7, 1st char: 1, "end" is a "reserved"

The symbol table contains:
yen@yenubuntu:~/lex$
```

3. numbertest.pas

```
yen@yenubuntu:~/lex$ cat numbertest.pas
1.0
1.00
1.00000000
00003.0000
00134.1000
.123
.000123
1.
002.
.123
yen@yenubuntu:~/lex$ ./b096060041.out < numbertest.pas
Line: 1, 1st char: 1, "1.0" is a "real"
Line: 2, 1st char: 1, "1.00" is a "real"
Line: 3, 1st char: 1, "1.00000000" is a "real"
Line: 4, 1st char: 1, "00003.0000" is a "real"
Line: 5, 1st char: 1, "00134.1000" is a "real"
Line: 6, 1st char: 1, ".123" is an "invalid number"
Line: 7, 1st char: 1, ".000123" is an "invalid number"
Line: 8, 1st char: 1, "1." is an "invalid number"
Line: 9, 1st char: 1, "002." is an "invalid number"
Line: 10, 1st char: 1, ".123" is an "invalid number"

The symbol table contains:
yen@yenubuntu:~/lex$
```

- test data of TA:

```
yen@yenubuntu:~/lex$ ./exec.sh
| test1.pas |
program test;
```

```
var
  i : integer;
begin
  read(i);
end;

Line: 1, 1st char: 1, "program" is a "reserved"
Line: 1, 1st char: 9, "test" is an "id"
Line: 1, 1st char: 13, ";" is a "symbol"
Line: 2, 1st char: 1, "var" is a "reserved"
Line: 3, 1st char: 3, "i" is an "id"
Line: 3, 1st char: 5, ":" is a "symbol"
Line: 3, 1st char: 7, "integer" is a "reserved"
Line: 3, 1st char: 14, ";" is a "symbol"
Line: 4, 1st char: 1, "begin" is a "reserved"
Line: 5, 1st char: 3, "read" is a "reserved"
Line: 5, 1st char: 7, "(" is a "symbol"
Line: 5, 1st char: 8, "i" is an "id"
Line: 5, 1st char: 9, ")" is a "symbol"
Line: 5, 1st char: 10, ";" is a "symbol"
Line: 6, 1st char: 1, "end" is a "reserved"
Line: 6, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
i
------------------------
| test2.pas |
program test;
var
  3i : string;
begin
  3i := 'ab;
end;

Line: 1, 1st char: 1, "program" is a "reserved"
Line: 1, 1st char: 9, "test" is an "id"
Line: 1, 1st char: 13, ";" is a "symbol"
Line: 2, 1st char: 1, "var" is a "reserved"
Line: 3, 1st char: 3, "3i" is an "invalid id"
Line: 3, 1st char: 6, ":" is a "symbol"
Line: 3, 1st char: 8, "string" is a "reserved"
Line: 3, 1st char: 14, ";" is a "symbol"
Line: 4, 1st char: 1, "begin" is a "reserved"
Line: 5, 1st char: 3, "3i" is an "invalid id"
Line: 5, 1st char: 6, ":=" is a "symbol"
Line: 5, 1st char: 9, "'ab" is an "invalid string"
Line: 5, 1st char: 4, ";" is a "symbol"
Line: 6, 1st char: 1, "end" is a "reserved"
Line: 6, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
------------------------
```

```
| test3.pas |
(* comment 1
   comment 2 *)
program test;
var
  i : integer;
begin
  read(i);
end;
Line: 1, 1st char: 1, "(* comment 1\n   comment 2 *)" is a "comment".
Line: 3, 1st char: 1, "program" is a "reserved"
Line: 3, 1st char: 9, "test" is an "id"
Line: 3, 1st char: 13, ";" is a "symbol"
Line: 4, 1st char: 1, "var" is a "reserved"
Line: 5, 1st char: 3, "i" is an "id"
Line: 5, 1st char: 5, ":" is a "symbol"
Line: 5, 1st char: 7, "integer" is a "reserved"
Line: 5, 1st char: 14, ";" is a "symbol"
Line: 6, 1st char: 1, "begin" is a "reserved"
Line: 7, 1st char: 3, "read" is a "reserved"
Line: 7, 1st char: 7, "(" is a "symbol"
Line: 7, 1st char: 8, "i" is an "id"
Line: 7, 1st char: 9, ")" is a "symbol"
Line: 7, 1st char: 10, ";" is a "symbol"
Line: 8, 1st char: 1, "end" is a "reserved"
Line: 8, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
i
------------------------
| test4.pas|
program test;
var
  f : float;
begin
  f := 12.25e+6;
end;
Line: 1, 1st char: 1, "program" is a "reserved"
Line: 1, 1st char: 9, "test" is an "id"
Line: 1, 1st char: 13, ";" is a "symbol"
Line: 2, 1st char: 1, "var" is a "reserved"
Line: 3, 1st char: 3, "f" is an "id"
Line: 3, 1st char: 5, ":" is a "symbol"
Line: 3, 1st char: 7, "float" is a "reserved"
Line: 3, 1st char: 12, ";" is a "symbol"
Line: 4, 1st char: 1, "begin" is a "reserved"
Line: 5, 1st char: 3, "f" is an "id"
Line: 5, 1st char: 5, ":=" is a "symbol"
Line: 5, 1st char: 8, "12.25e+6" is a "real"
Line: 5, 1st char: 16, ";" is a "symbol"
Line: 6, 1st char: 1, "end" is a "reserved"
Line: 6, 1st char: 4, ";" is a "symbol"
```

```
The symbol table contains:
test
f
------------------------
| test5.pas |
(* a**b) *)
program test;
var
  i : integer;
  _s, _s2, _s3, _s4, _s5 : string;
begin
  i := -100;
  _s := 'db lab';
  _s2 := 'You''ll see';
  _s3 := '';
  _s4 := '''';
  _s5 := ' ';
end;
Line: 1, 1st char: 1, "(* a**b) *)" is a "comment".
Line: 2, 1st char: 1, "program" is a "reserved"
Line: 2, 1st char: 9, "test" is an "id"
Line: 2, 1st char: 13, ";" is a "symbol"
Line: 3, 1st char: 1, "var" is a "reserved"
Line: 4, 1st char: 3, "i" is an "id"
Line: 4, 1st char: 5, ":" is a "symbol"
Line: 4, 1st char: 7, "integer" is a "reserved"
Line: 4, 1st char: 14, ";" is a "symbol"
Line: 5, 1st char: 3, "_s" is an "id"
Line: 5, 1st char: 5, "," is an "invalid character".
Line: 5, 1st char: 7, "_s2" is an "id"
Line: 5, 1st char: 10, "," is an "invalid character".
Line: 5, 1st char: 12, "_s3" is an "id"
Line: 5, 1st char: 15, "," is an "invalid character".
Line: 5, 1st char: 17, "_s4" is an "id"
Line: 5, 1st char: 20, "," is an "invalid character".
Line: 5, 1st char: 22, "_s5" is an "id"
Line: 5, 1st char: 26, ":" is a "symbol"
Line: 5, 1st char: 28, "string" is a "reserved"
Line: 5, 1st char: 34, ";" is a "symbol"
Line: 6, 1st char: 1, "begin" is a "reserved"
Line: 7, 1st char: 3, "i" is an "id"
Line: 7, 1st char: 5, ":=" is a "symbol"
Line: 7, 1st char: 8, "-100" is an "integer"
Line: 7, 1st char: 12, ";" is a "symbol"
Line: 8, 1st char: 3, "_s" is an "id"
Line: 8, 1st char: 6, ":=" is a "symbol"
Line: 8, 1st char: 9, "'db lab'" is a "string"
Line: 8, 1st char: 17, ";" is a "symbol"
Line: 9, 1st char: 3, "_s2" is an "id"
Line: 9, 1st char: 7, ":=" is a "symbol"
Line: 9, 1st char: 10, "'You''ll see'" is a "string"
Line: 9, 1st char: 23, ";" is a "symbol"
Line: 10, 1st char: 3, "_s3" is an "id"
Line: 10, 1st char: 7, ":=" is a "symbol"
```

```
Line: 10, 1st char: 10, "'" is an "invalid string"
Line: 10, 1st char: 10, "'" is an "invalid string"
Line: 10, 1st char: 2, ";" is a "symbol"
Line: 11, 1st char: 3, "_s4" is an "id"
Line: 11, 1st char: 7, ":=" is a "symbol"
Line: 11, 1st char: 10, "''''" is a "string"
Line: 11, 1st char: 14, ";" is a "symbol"
Line: 12, 1st char: 3, "_s5" is an "id"
Line: 12, 1st char: 7, ":=" is a "symbol"
Line: 12, 1st char: 10, "' '" is a "string"
Line: 12, 1st char: 13, ";" is a "symbol"
Line: 13, 1st char: 1, "end" is a "reserved"
Line: 13, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
i
_s
_s2
_s3
_s4
_s5
-----------------------
| test6.pas |
ProGram test;
var
  #db : float;
  _f2 : float;
begin
  #db := .1;
  _f2 := 12.100;
end;
Line: 1, 1st char: 1, "ProGram" is a "reserved"
Line: 1, 1st char: 9, "test" is an "id"
Line: 1, 1st char: 13, ";" is a "symbol"
Line: 2, 1st char: 1, "var" is a "reserved"
Line: 3, 1st char: 3, "#db" is an "invalid id"
Line: 3, 1st char: 7, ":" is a "symbol"
Line: 3, 1st char: 9, "float" is a "reserved"
Line: 3, 1st char: 14, ";" is a "symbol"
Line: 4, 1st char: 3, "_f2" is an "id"
Line: 4, 1st char: 7, ":" is a "symbol"
Line: 4, 1st char: 9, "float" is a "reserved"
Line: 4, 1st char: 14, ";" is a "symbol"
Line: 5, 1st char: 1, "begin" is a "reserved"
Line: 6, 1st char: 3, "#db" is an "invalid id"
Line: 6, 1st char: 7, ":=" is a "symbol"
Line: 6, 1st char: 10, ".1" is an "invalid number"
Line: 6, 1st char: 12, ";" is a "symbol"
Line: 7, 1st char: 3, "_f2" is an "id"
Line: 7, 1st char: 7, ":=" is a "symbol"
Line: 7, 1st char: 10, "12.100" is a "real"
Line: 7, 1st char: 16, ";" is a "symbol"
Line: 8, 1st char: 1, "end" is a "reserved"
```

```
Line: 8, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
_f2
------------------------
| test7.pas |
(* This line is a comment. *)
program test;
var
  i : integer;
begin
  i := 1+2;
end;
Line: 1, 1st char: 1, "(* This line is a comment. *)" is a "comment".
Line: 2, 1st char: 1, "program" is a "reserved"
Line: 2, 1st char: 9, "test" is an "id"
Line: 2, 1st char: 13, ";" is a "symbol"
Line: 3, 1st char: 1, "var" is a "reserved"
Line: 4, 1st char: 3, "i" is an "id"
Line: 4, 1st char: 5, ":" is a "symbol"
Line: 4, 1st char: 7, "integer" is a "reserved"
Line: 4, 1st char: 14, ";" is a "symbol"
Line: 5, 1st char: 1, "begin" is a "reserved"
Line: 6, 1st char: 3, "i" is an "id"
Line: 6, 1st char: 5, ":=" is a "symbol"
Line: 6, 1st char: 8, "1" is an "integer"
Line: 6, 1st char: 9, "+" is an "operator"
Line: 6, 1st char: 10, "2" is an "integer"
Line: 6, 1st char: 11, ";" is a "symbol"
Line: 7, 1st char: 1, "end" is a "reserved"
Line: 7, 1st char: 4, ";" is a "symbol"

The symbol table contains:
test
i
yen@yenubuntu:~/lex$
```