

rpnCalc.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include "lexical.h"
5 #include "nextInputChar.h"
6 #include "tokenStack.h"
7
8 /*
9  *This is my solution to the lab5 of eecs2031 about building a RPN
  Calculator.
10  *Name: Yuxian Wang
11  *Student Number: 215170418
12  *Date: 2017/11/03
13  */
14
15 /*pop the top element off of the stack */
16 static int popInt(struct tokenStack *s) {
17     if (s->top < 0) {
18         printf("popInt: error, aborting.\n");
19         exit(0);
20     }
21     struct lexToken *token;
22     token = allocToken();
23     token = popTokenStack(s);
24     freeToken(token);
25     return atoi(token->symbol);
26 }
27
28 /* take an int and create a lexToken */
29 static void pushInt(struct tokenStack *s, int v) {
30     struct lexToken *token;
31     token = allocToken();
32     token->kind = LEX_TOKEN_NUMBER; /* holds a LEX_TOKEN_NUMBER */
33     sprintf(token->symbol, "%c", v); /* push it on the stack*/
34     pushTokenStack(s, token);
35 }
36
37 static void doOperator(struct tokenStack *s, char *op) {
38     if (!strcmp(op, "quit")) {
39         exit(0);
40     } else if (!strcmp(op, "print")) {
```

rpnCalc.c

```
41     struct lexToken *t = popTokenStack(s);
42     dumpToken(stdout, t);
43     freeToken(t);
44 } else {
45     fprintf(stderr, "unknown command %s\n", op);
46     exit(1);
47 }
48 }
49
50 int main(int argc, char *argv[]) {
51     setFile(stdin);
52     struct tokenStack *stack;
53     stack = createTokenStack();
54     struct lexToken *token;
55     int kind, num1, num2;
56     char op;
57     while (token = nextToken()) {
58         kind = token->kind;
59         switch (kind) {
60             case LEX_TOKEN_NUMBER:
61                 pushInt(stack, atoi(token->symbol));
62                 break;
63             case LEX_TOKEN_IDENTIFIER:
64                 doOperator(stack, token->symbol);
65                 break;
66             case LEX_TOKEN_OPERATOR:
67                 switch (token->symbol[0]) {
68                     case '+':
69                         pushInt(stack, popInt(stack) + popInt(stack));
70                         break;
71                     case '-':
72                         num1 = popInt(stack);
73                         num2 = popInt(stack);
74                         pushInt(stack, num1 - num2);
75                         break;
76                     case '*':
77                         pushInt(stack, popInt(stack) * popInt(stack));
78                         break;
79                     case '/':
80                         num1 = popInt(stack);
81                         num2 = popInt(stack);
```

rpnCalc.c

```
82         pushInt(stack, num2 / num1);
83         break;
84     default:
85         printf("wrong\n");
86     }
87     break;
88     case LEX_TOKEN_EOF:
89         printf("end of line, quit.\n");
90         exit(1);
91         break;
92     default:
93         printf("wrong");
94 }
95 }
96 return 0;
97 }
98
```