

```
1  /*
2  再帰的下向き構文解析プログラム
3  (以下の文法に対する構文解析プログラム)
4  Expr    -> CON
5           | FID '(' Expr2 ')'
6  Expr2   -> Expr Rest
7  Rest    -> ',' Expr
8           | ε
9
10 -- 以下は終端記号: 字句解析部で処理
11 EOL     -> '\n'      -- End Of Line
12 CON     -> '0' | '1' | ':' | '9'  -- 一桁の数のみ
13 FID     -> '+' | '-' | '*' | '!'
14 */
15
16 #include <stdio.h>
17 #include <stdlib.h> /* exit()用 */
18 #include <ctype.h> /* isdigit()用 */
19
20 /* 大域変数の宣言 */
21 int token; /* 入力の先頭のトークンを表す */
22 int yylval; /* tokenの属性 */
23
24 int column = 0; /* デバッグ用 */
25
26 /* 終端記号に対応するマクロの定義 */
27 #define CON 256
28 #define FID 257
29 #define EOL 258 /* End of Line -- $に相当 */
30
31 /* 簡易字句解析ルーチン */
32 int yylex(void) {
33     int c;
34
35     if (token==EOL) { /* 前のトークンが改行だったら */
36         column=0;
37     }
38
39     do {
40         c = getchar ();
41         column++;
42     } while (c == ' ' || c == '\t');
43
44     if (isdigit (c)) {
45         yylval = c-'0'; /* 数字から数へ変換 */
46         return CON;
47     }
48
49     if (c == '+' || c == '-' || c == '*' || c == '!') {
50         yylval = c;
51         return FID;
52     }
53
54     if (c == '\n') {
55         return EOL; /* 行末が $ (入力の終わり) に対応する */
56     }
57
58     if (c == EOF) { /* ファイルの終 */
59         exit(0);
60     }
}
```

```
61     /* 上のどの条件にも合わなければ、文字をそのまま返す。*/
62     return c; /* '(', ')', ',', ' ' など */
63 }
64
65 char* tokenName(int t) {
66     switch (t) {
67         case 256: return "CON";
68         case 257: return "FID";
69         case 258: return "EOL";
70         case 259: return "Unknown";
71     }
72 }
73
74 /* token (終端記号) を消費して、次の token を読む */
75 void eat(int t) {
76     if (token == t) {
77         token = yylex();
78         return;
79     } else {
80         if (isprint(t)) {
81             printf("eat: Character '%c' is expected at column %d ", t, column);
82         } else {
83             printf("eat: Token %s is expected at column %d ",
84                 tokenName(t), column);
85         }
86         if (isprint(token)) {
87             printf("instead of '%c'.\n", token);
88         } else {
89             printf("instead of %s.\n", tokenName(token));
90         }
91         exit(1);
92     }
93 }
94
95 /* エラーメッセージの出力 */
96 void errorMessage(char* place) {
97     if (isprint(token)) {
98         printf("%s: Unexpected token: '%c' at column %d.\n", place, token, column);
99     } else {
100         printf("%s: Unexpected token: %s at column %d.\n",
101             place, tokenName(token), column);
102     }
103 }
104 /* 関数プロトタイプ宣言 */
105 void Expr(void);
106 void Expr2(void);
107 void Rest(void);
108
109 /*
110 再帰的構文解析関数群
111 文法の各非終端記号に対応する関数
112 */
113 void Expr(void) {
114     switch (token) {
115         case CON:
116             eat(CON); break;
117         case FID:
118             eat(FID); eat('('); Expr2(); eat(')'); break;
119         default:
120             errorMessage("Expr");
```

```
121         exit(1);
122         break;
123     }
124 }
125
126 void Expr2(void) {
127     if (token ==CON || token==FID ) {
128         Expr(); Rest();
129     } else {
130         errorMessage("Expr2");
131         exit(1);
132         break;
133     }
134 }
135
136 void Rest(void) {
137     switch (token) {
138     case ',':
139         eat(','); Expr(); break;
140     case ')':
141         /* do nothing */ break;
142     default:
143         errorMessage("Rest");
144         exit(1);
145         break;
146     }
147 }
148
149 /* 各行の処理 */
150 void processLine(void) {
151     Expr();
152     if (token==EOL) { /* 入力がブロックしないように改行は特別扱い */
153         printf("Correct!\n"); /* eat(EOL)の前に出力しておく */
154     }
155     eat(EOL);
156 }
157
158 /* main関数 */
159 int main(void) {
160     printf("Ctrl-cで終了します。 \n");
161     token = yylex(); /* 最初のトークンを読む */
162     while (1 /* 無限ループ */) {
163         processLine(); /* 各行を処理する */
164     }
165 }
166
```