```c
#include <conio.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "posfix-tradutor.h"

/*---------------------------------------------------------*/
/*---------------DEFINICAO DE VARIAVEIS--------------------*/
/*---------------------------------------------------------*/
FILE    *arquivoDeEntrada;
FILE    *arquivoDeSaida;
CAMINHO caminhoArquivo;

TabelaDeSimbolos tabelaSimbolos;

Token   lookahead;

Token   abrePar;
Token   fechaPar;
Token   pontoVirgula;

int     contadorDeLinha;
int     contadorDeColuna;
/*---------------------------------------------------------*/
/*-------------------------FIM-----------------------------*/
/*---------------------------------------------------------*/



/*---------------------------------------------------------*/
/*-------------IMPLEMENTACAO DAS FUNCOES-------------------*/
/*---------------------------------------------------------*/
void inicializar(void) {
    carregarArquivo("entrada.txt");

    lookahead.token          = NONE;
    lookahead.atributo       = ESPACO_BRANCO;
    strcpy(lookahead.lexema, VAZIO);
    tabelaSimbolos.qtdTokens = 0;
    tabelaSimbolos.qtdIds    = 0;
    tabelaSimbolos.qtdNums   = 0;
    contadorDeLinha          = 1;
    contadorDeColuna         = 0;

    abrePar.token          = ABRE_PAR;
    abrePar.atributo       = ESPACO_BRANCO;
    strcpy(abrePar.lexema, VAZIO);
```

```
50:
51:     fechaPar.token          = FECHA_PAR;
52:     fechaPar.atributo       = ESPACO_BRANCO;
53:     strcpy(fechaPar.lexema, VAZIO);
54:
55:     pontoVirgula.token          = PONTO_E_VIRGULA;
56:     pontoVirgula.atributo       = ESPACO_BRANCO;
57:     strcpy(pontoVirgula.lexema, VAZIO);
58: }
59:
60: void carregarArquivo(CAMINHO c) {
61:     printf("\nCarregando o arquivo de entrada...");
62:     arquivoDeEntrada = fopen("entrada.txt", "r");
63:     if(arquivoDeEntrada == NULL) {
64:         printf("\nERRO AO TENTAR ABRIR O ARQUIVO DE ENTRADA!");
65:         printf("\ncrie o arquivo 'entrada.txt' no diretorio do programa!");
66:         aguardar(SAIDA);
67:     }
68:     printf("\nArquivo carregado com sucesso!");
69:
70:     printf("\n\nCarregando o arquivo de saida...");
71:     arquivoDeSaida= fopen("SAIDA.txt", "a+");
72:     if(arquivoDeSaida == NULL) {
73:         printf("\nERRO AO TENTAR ARBIR O ARQUIVO DE SAIDA!");
74:         aguardar(SAIDA);
75:     }
76:     printf("\nArquivo carregado com sucesso!");
77:     aguardar(CONTINUE);
78: }
79:
80: void finalizar(void) {
81:     fclose(arquivoDeEntrada);
82:     fclose(arquivoDeSaida);
83:
84:     printf("\nO arquivo 'SAIDA.txt' contendo os resultados da traducao");
85:     printf("\nfoi gerado no mesmo diretorio do programa. VERIFIQUE!");
86:     aguardar(SAIDA);
87: }
88: /*-------------------------------------------------------*/
89: Token analiseLexica(void) {
90:     Token t;
91:     int   p;
92:     char  c;
93:     for(;;) {
94:         c = getc(arquivoDeEntrada);
95:         contadorDeColuna++;
96:         if(c == EOF)
97:              erro("ERRO! O ARQUIVO CHEGOU AO FINAL.");
98:         else if(c == ESPACO_BRANCO || c == TABULACAO)
```

```
 99:             continue;
100:         else if(c == QUEBRA_LINHA) {
101:             c = getc(arquivoDeEntrada);
102:             contadorDeLinha++;
103:         }else if(isdigit(c)) {
104:             t.token    = NUM;
105:             t.atributo = c;
106:             strcpy(t.lexema, VAZIO);
107:         } else if(isalpha(c)) { /* eh um ID (identificador de variavel) */
108:             t.token    = ID;
109:             t.atributo = tabelaSimbolos.qtdIds+1;
110:             p = 0;
111:             while(isalnum(c)) {
112:                t.lexema[p] = c;
113:                c  = getc(arquivoDeEntrada);
114:                p++;
115:                if(p >= MAX_LEXEMA)
116:                     erro("ERRO! TAMANHO DO LEXEMA NAO SUPORTADO!");
117:             }
118:             ungetc(c, arquivoDeEntrada);
119:         } else { /* eh um operador */
120:             t.token    = c;
121:             t.atributo = ESPACO_BRANCO;
122:             strcpy(t.lexema, VAZIO);
123:         }
124:         inserir(t);
125:         return t;
126:     }
127: }
128:
129: void inserir(Token t) {
130:     tabelaSimbolos.qtdTokens++;
131:     if(t.token == ID)  tabelaSimbolos.qtdIds++;
132:     if(t.token == NUM) tabelaSimbolos.qtdNums++;
133:     tabelaSimbolos.simbolos[tabelaSimbolos.qtdTokens].token = t.token;
134:     tabelaSimbolos.simbolos[tabelaSimbolos.qtdTokens].atributo = t.atributo
135:     strcpy(tabelaSimbolos.simbolos[tabelaSimbolos.qtdTokens].lexema, t.lexe
136: }
137:
138: int buscar(Token t) {
139:     int i;
140:     for(i = 0; i <= tabelaSimbolos.qtdTokens; i++) {
141:         if((tabelaSimbolos.simbolos[i].token == t.token) &&
142:            (tabelaSimbolos.simbolos[i].atributo == t.atributo) &&
143:            (strcmp(tabelaSimbolos.simbolos[i].lexema, t.lexema) == 0)
144:           ) return i;
145:     }
146:     return NONE;
147: }
```

```
148: /*-------------------------------------------------------*/
149: void traduzir(void) {
150:      inicializar();
151:      lookahead = analiseLexica();
152:      cmd();
153:      finalizar();
154: }
155:
156: void cmd() {
157:     imprimir(ID);
158:     reconhecer(ID);
159:     reconhecer(ATR);
160:     expr();
161:     imprimir(ATR);
162: }
163:
164: void expr(void) {
165:     termo();
166:     R1();
167: }
168:
169: void R1() {
170:     if(lookahead.token == ADD) {
171:         reconhecer(ADD);
172:         termo();
173:         imprimir(ADD);
174:         R1();
175:     }
176: }
177:
178: void termo(void) {
179:     fator();
180:     R2();
181: }
182:
183: void R2() {
184:     if(lookahead.token == MUL) {
185:         reconhecer(MUL);
186:         fator();
187:         imprimir(MUL);
188:         R2();
189:     }
190: }
191:
192: void fator(void) {
193:     if(lookahead.token == ID) {
194:            imprimir(ID);
195:            reconhecer(ID);
196:     } else if(lookahead.token == NUM) {
```

```
197:             imprimir(NUM);
198:             reconhecer(NUM);
199:     } else if(lookahead.token ==  ABRE_PAR) {
200:             reconhecer(ABRE_PAR);
201:             expr();
202:             reconhecer(FECHA_PAR);
203:     } else if(lookahead.token == SUB) {
204:             reconhecer(SUB);
205:             fator();
206:             imprimir(SUB);
207:     } else erro("OCORREU UM ERRO DE SINTAXE!\n");
208: }
209:
210: void reconhecer(TOKEN t) {
211:      if(lookahead.token == t)
212:          lookahead = analiseLexica();
213:      else
214:          erro("OCORREU UM ERRO DE SINTAXE!\n");
215: }
216: /*-------------------------------------------------------*/
217: void imprimir(TOKEN t) {
218:      int p;
219:      switch(t) {
220:           case ADD:
221:           case SUB:
222:           case MUL:
223:           case DIV:
224:           case ATR:
225:                fprintf(arquivoDeSaida, "%c", t);
226:                printf("%c", t);
227:                break;
228:           case NUM:
229:                fprintf(arquivoDeSaida, "%c", lookahead.atributo);
230:                printf("%c", lookahead.atributo);
231:                break;
232:           case ID:
233:                fprintf(arquivoDeSaida, "%s", lookahead.lexema);
234:                printf("%s", lookahead.lexema);
235:                break;
236:           default:
237:                erro("OCORREU UM ERRO DE SINTAXE!");
238:      }
239: }
240: /*-------------------------------------------------------*/
241: void erro(MENSAGEM m) {
242:     fprintf(arquivoDeSaida, "\n[LINHA %d, COLUNA %d]: %s\n", contadorDeLinh
243:     printf("\n[LINHA %d, COLUNA %d]: %s\n", contadorDeLinha, contadorDeColu
244:     aguardar(SAIDA);
245: }
```

```
246: /*------------------------------------------------------*/
247: void aguardar(int opcao){
248:      if(opcao) {
249:          printf("\n\nPressione qualquer tecla para continuar...\n\n");
250:          getch();
251:      }else {
252:          printf("\n\nPressione qualquer tecla para finalizar...\n\n");
253:          getch();
254:          exit(1);
255:      }
256:
257: }
258: /*------------------------------------------------------*/
259: /*-----------------------FIM----------------------------*/
260: /*------------------------------------------------------*/
```