Instituto Federal do Sul de Minas Gerais

Compiladores

Análise Sintática – Parte 3

douglas.braz@ifsuldeminas.edu.br

Exemplo

```
1  /*+-------------------------------------------------------------------+
2    | Implementacao do algoritmo de analise sintatica LR(K).           |
3    | Por Luiz Eduardo da Silva                                         |
4    +-------------------------------------------------------------------+*/
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8
9  /*+-------------------------------------------------------------------+
```

Exemplo

```
10      | Vocabulario, Regras de Producao de uma gramatica                |
11      | e a Tabela de Analise LR(k) para a esta gramatic                |
12      +-----------------------------------------------------------------+*/
13
14  #define NSIMBOLOS 7
15  #define NREGRAS   4
16  #define NESTADOS  9
17  char alfabeto[NSIMBOLOS+1] = "SLa[];#";
18  char *regras[NREGRAS] =
19          { "S::=a",
20            "S::=[L]",
21            "L::=S",
22            "L::=L;S" };
23
24  struct {
25    char acao;
26    int indice;
27  } TabSint[NESTADOS][NSIMBOLOS] =
```

Exemplo

```
28  { 'e',1, '␣',0, 'e',2, 'e',3, '␣',0, '␣',0, '␣',0,
29    '␣',0, '␣',0, '␣',0, '␣',0, '␣',0, '␣',0, 'a',0,
30    '␣',0, '␣',0, 'r',1, 'r',1, 'r',1, 'r',1, 'r',1,
31    'e',4, 'e',5, 'e',2, 'e',3, '␣',0, '␣',0, '␣',0,
32    '␣',0, '␣',0, 'r',3, 'r',3, 'r',3, 'r',3, 'r',3,
33    '␣',0, '␣',0, '␣',0, '␣',0, 'e',6, 'e',7, '␣',0,
34    '␣',0, '␣',0, 'r',2, 'r',2, 'r',2, 'r',2, 'r',2,
35    'e',8, '␣',0, 'e',2, 'e',3, '␣',0, '␣',0, '␣',0,
36    '␣',0, '␣',0, 'r',4, 'r',4, 'r',4, 'r',4, 'r',4,
37  };
```

Exemplo

```
138 /*+-------------------------------------------------------------+
139   | Pilha sintatica utilizada pelo algoritmo de analise LR(K    |
140   +-------------------------------------------------------------+*/
141 struct {
142   char elem;
143   int ind;
144 } P[20];
145
146 void traco (int);
147 void strins (char *, int, char *, int);
148
```

Exemplo

```
156  int main()
157  {
158    int i, j, k, termino, reduzido, indice, ind, tam, passo,
159        nreducao, indreduz = -1, reducoes[50];
160    char sentenca[40], pilha[60], cadeia[40], str[40];
161    char s, simboloreduzido = '␣', acao;
162
```

Exemplo

```
163     P[0].elem = 'e';
164     P[0].ind = 0;
165     i = termino = reduzido = 0;
166
167     printf ("\nDigite a sentenca: ");
168     gets (sentenca);
169
170     strcat (sentenca, "#");
171     indice = 0;
172     passo = 0;
173     printf ("PASSO %-30s S.R. %-15s  %s\n", "PILHA", "SENTENCA", "ACAO")
            ;
```

Exemplo

```
174     traco (79);
175     while (!termino) {
176       if (reduzido)
177           s = simboloreduzido;
178       else
179           s = sentenca[indice];
180       for (j=0; alfabeto[j] != s && j <= strlen(alfabeto); j++);
181       if (alfabeto[j] != s) {
182           printf ("\nERRO: o simbolo <%c> nao e reconhecido nesta
                  linguagem", s);
183           printf ("\n\nDigite '.' para terminar!");
184           while (getchar()!='.');
185           exit(10);
186       }
187       acao = TabSint[P[i].ind][j].acao;
188       ind = TabSint[P[i].ind][j].indice;
```

Exemplo

```
189     for (j = 0, k = indice; sentenca[k]; j++, k++)
190         cadeia[j] = sentenca[k];
191     cadeia[j] = '\0';
192     strcpy (pilha, "");
193     for (k = 0; k <= i; k++) {
194         sprintf (str,"%c%d", P[k].elem, P[k].ind);
195         strcat (pilha, str);
196     }
197     printf ("%3d    %-30s %c     %-15s  %c%d\n",
198                 passo++, pilha, simboloreduzido, cadeia, acao, ind);
199     switch (acao) {
200         case 'e':  i++;
201                    P[i].elem = s;
202                    P[i].ind = ind;
203                    if (reduzido) {
204                        reduzido = 0;
205                        simboloreduzido = ' ';
```

Exemplo

```
206                     }
207                     else
208                         indice++;
209                     break;
210         case 'r':   tam = strlen (regras[ind-1]);
211                     i = i - tam + 4;
212                     reduzido = 1;
213                     reducoes[++indreduz] = ind-1;
214                     simboloreduzido = regras[ind-1][0];
215                     break;
216         case 'a':   termino = 1;
217                     printf ("\nA sentenca <%s> esta correta", sentenca);
218                     break;
219         case ' ':   printf ("\nA sentenca <%s> NAO e reconhecida",
                sentenca);
220                     printf ("\n\nDigite '.' para terminar!");
221                     while (getchar()!='.');
222                     exit (1);
223     }
224     /*getchar();*/
225 }
```

```
226     /*- Mostra a serie de derivacoes mais a direita para produzir a
            sentenca -*/
227     printf ("\n\nGramatica:");
228     printf ("\n==========\n");
229     for (i = 0; i < NREGRAS; i++)
230         printf ("(%d).  %s\n", i+1, regras[i]);
231     printf ("\n\nSequencia de Derivacoes mais a direita:");
232     printf ("\n================================================\n\n");
233     sentenca[0] = regras[0][0];
234     sentenca[1] = '\0';
235     printf ("%s ", sentenca);
236     while (indreduz >= 0)
237     {
238       i = strlen (sentenca) - 1;
239       while (sentenca[i] < 'A' || sentenca[i] > 'Z') i--;
240       nreducao = reducoes[indreduz--];
241       strins (regras[nreducao], 4, sentenca, i);
242       printf ("=%d=> %s ", nreducao+1, sentenca);
243     }
244     printf ("\n\nDigite '.' para terminar!");
245     while (getchar()!='.');
246 }
247
248 /*+----------------------------------------------------------------+
```

```
249    | Desenha uma linha de hifens na tela.                          |
250    +---------------------------------------------------------------+*/
251  void traco (int i) {
252    int k;
253    for (k = 0; k < i; k++)
254        printf ("-");
255    printf ("\n");
256  }
257
258  /*+---------------------------------------------------------------+
259    |Insere substring da pos1 ate o final de s1 na pos2 do string s2 |
260    +---------------------------------------------------------------+*/
261  void strins (char *s1, int pos1, char *s2, int pos2) {
262    int i, tam_s1, tam_s2;
263    for (tam_s1 = 0; s1[tam_s1]; tam_s1++);
264    for (tam_s2 = 0; s2[tam_s2]; tam_s2++);
265    for (i = tam_s2; i >= pos2; i--)
266        s2[i+tam_s1-pos1-1] = s2[i];
267    for (i = pos1; i < tam_s1; i++)
268        s2[i+pos2-pos1] = s1[i];
269    s2[tam_s1+tam_s2-pos1]='\0';
270  }
```

Referência Bibliográfica

- Silva, L. E. Notas de Aula de Compiladores. Luiz Eduardo da Silva. Alfenas, MG: UNIFAL-MG, Universidade Federal de Alfenas, 2016.

- Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles, techniques, and tools*. Vol. 2. Reading: Addison-wesley, 2007.