

## Julho de 2019

<https://github.com/CarlosMatheus/CES-41-Exercise-Lists>

O Exercício foi resolvido com sucesso. Seu funcionamento pode ser observado pela saída representada pelo Código 2 que é referente a entrada representada no Código 1.

[illegible]

Figura 1: A tabela de execução para entrada  $id * ((id + id) * (id * (id + id)))$

| A          | Primeiro(A)         |
|------------|---------------------|
| ;          | ;                   |
| int        | int                 |
| real       | real                |
| ID         | ID                  |
| ,          | ,                   |
| {          | {                   |
| }          | }                   |
| =          | =                   |
| +          | +                   |
| (          | (                   |
| )          | )                   |
| CTE        | CTE                 |
| *          | *                   |
| Prog       | int real            |
| Decls      | int real            |
| LDaux      | $\epsilon$ int real |
| Declaracao | int real            |
| Tipo       | int real            |
| ListId     | ID                  |
| LIaux      | $\epsilon$ ,        |
| CmdComp    | {                   |
| ListCmd    | { ID                |
| LCaux      | $\epsilon$ { ID     |
| Comando    | { ID                |
| CmdAtrib   | ID                  |
| Expressao  | ( ID CTE            |
| Eaux       | $\epsilon$ +        |
| Termo      | ( ID CTE            |

Figura 2: Tabela dos Primeiros

## Exercício 2

O Exercício foi resolvido com sucesso. Para a gramática do exemplo da Seção 5.4.4 do Capítulo V dos Slides Teóricos de CES-41 e a tabela de produções de um analisador preditor não-recursivo, no mesmo exemplo, foi feita a tabela de execução simulando a análise da sentença  $id * ((id + id) * (id * (id + id)))$$ . A tabela de execução pode ser vista na Figura 1.

## Exercício 3

O Exercício foi resolvido com sucesso. A tabela de produções do analisador sintático preditor não-recursivo foi feita com sucesso. O Primeiro de cada símbolo pode ser visto na Figura 2. O Seguinte pode ser visto na Figura 3. A tabela dos Primeiros e Seguintes pode ser vista na Figura 4. A tabela de produções do analisador sintático preditor não-recursivo pode ser vista na Figura 5 inteira. Devido à sua largura, ela foi separada em dividida em segmentos em apresentada da Figura 6 à Figura 9 de modo a facilitar a visualização.

```

/* Programa para contar as ocorrencias das
   palavras de um texto */

program AnaliseDeTexto {

/* Variaveis globais */

global:
  char nomes[50,10], palavra[10];
  int ntab, nocorr[50];
  char c; logic fim;

functions:

/* Funcao para procurar uma palavra na tabela de palavras */

int Procura () {

```

| A          | Seguinte(A) |
|------------|-------------|
| Prog       | \$          |
| DeclS      | {           |
| CmdComp    |             |
| Declaracao | int real {  |
| LDaux      | {           |
| Tipo       | ID          |
| ListId     | ;           |
| LIaux      | ;           |
| ListCmd    | }           |
| Comando    | { ID }      |
| LCaux      | }           |
| CmdAtrib   |             |
| Expressao  | ; )         |
| Termo      | + ; )       |
| Eaux       | ; )         |

Figura 3: Tabela dos Seguintes

| A $\rightarrow$ $\alpha$             | Prim( $\alpha$ ) | Prim(A)                | Seg(A)     |
|--------------------------------------|------------------|------------------------|------------|
| Prog $\rightarrow$ DeclsCmdComp      | int real         | int real               | \$         |
| Decls $\rightarrow$ DeclaracaoLDaux  | int real         | int real               | {          |
| LDaux $\rightarrow$ $\varepsilon$    | $\varepsilon$    | $\varepsilon$ int real | {          |
| LDaux $\rightarrow$ Decls            | int real         | $\varepsilon$ int real | {          |
| Declaracao $\rightarrow$ TipoListId; | int real         | int real               | int real { |
| Tipo $\rightarrow$ int               | int              | int real               | ID         |
| Tipo $\rightarrow$ real              | real             | int real               | ID         |
| ListId $\rightarrow$ IDLIaux         | ID               | ID                     | ;          |
| LIaux $\rightarrow$ $\varepsilon$    | $\varepsilon$    | $\varepsilon$ ,        | ;          |
| LIaux $\rightarrow$ ,ListId          | ,                | $\varepsilon$ ,        | ;          |
| CmdComp $\rightarrow$ {ListCmd}      | {                | {                      |            |
| ListCmd $\rightarrow$ ComandoLCAux   | { ID             | { ID                   | }          |
| LCAux $\rightarrow$ $\varepsilon$    | $\varepsilon$    | $\varepsilon$ { ID     | }          |
| LCAux $\rightarrow$ ListCmd          | { ID             | $\varepsilon$ { ID     | }          |
| Comando $\rightarrow$ CmdComp        | {                | { ID                   | { ID }     |
| Comando $\rightarrow$ CmdAtrib       | ID               | { ID                   | { ID }     |
| CmdAtrib $\rightarrow$ ID=Expressao; | ID               | ID                     |            |
| Expressao $\rightarrow$ TermoEaux    | ( ID CTE         | ( ID CTE               | ; )        |
| Eaux $\rightarrow$ $\varepsilon$     | $\varepsilon$    | $\varepsilon$ +        | ; )        |
| Eaux $\rightarrow$ +Expressao        | +                | $\varepsilon$ +        | ; )        |
| Termo $\rightarrow$ (Expressao)      | (                | ( ID CTE               | + ; )      |
| Termo $\rightarrow$ ID               | ID               | ( ID CTE               | + ; )      |
| Termo $\rightarrow$ CTE              | CTE              | ( ID CTE               | + ; )      |

Figura 4: Tabela dos Primeiros e Seguintes

[illegible]

Figura 5: Tabela de produções

| Não terminal  | ;   | int   | real   |
|---|---|---|--|
| Prog<br>Decls<br>CmdComp<br>Declaracao<br>LDaux<br>Tipo<br>ListId<br>LIAux<br>ListCmd<br>Comando<br>LCaux<br>CmdAtrib<br>Expressao<br>Termo<br>Eaux | LIAux → ε<br><br><br><br><br><br><br><br><br><br><br><br>Eaux → ε | Prog → DeclsCmdComp<br>Decls → DeclaracaoLDaux<br><br>Declaracao → TipoListId;<br>LDaux → Decls<br>Tipo → int | Prog → DeclsCmdComp<br>Decls → DeclaracaoLDaux<br><br>Declaracao → TipoListId;<br>LDaux → Decls<br>Tipo → real |

Figura 6: Tabela de produções, parte 1

| Átomo de entrada   |                             |  |                                   |  |
|--|-----------------------------|--|-----------------------------------|--|
| ID   | ,                           | {  | }                                 |  |
| ListId $\rightarrow$ IDLIaux<br>ListCmd $\rightarrow$ ComandoLCaux<br>Comando $\rightarrow$ CmdAtrib<br>LCaux $\rightarrow$ ListCmd<br>CmdAtrib $\rightarrow$ ID=Expressao;<br>Expressao $\rightarrow$ TermoEaux<br>Termo $\rightarrow$ ID | LIaux $\rightarrow$ ,ListId | CmdComp $\rightarrow$ {ListCmd}<br>LDaux $\rightarrow$ $\varepsilon$<br>ListCmd $\rightarrow$ ComandoLCaux<br>Comando $\rightarrow$ CmdComp<br>LCaux $\rightarrow$ ListCmd | LCaux $\rightarrow$ $\varepsilon$ |  |

Figura 7: Tabela de produções, parte 2

| = | +                             | (  | )                                |  |
|---|-------------------------------|--|----------------------------------|--|
|   | Eaux $\rightarrow$ +Expressao | Expressao $\rightarrow$ TermoEaux<br>Termo $\rightarrow$ (Expressao) | Eaux $\rightarrow$ $\varepsilon$ |  |

Figura 8: Tabela de produções, parte 3

| CTE  | * | \$ |
|--|---|----|
| Expressao $\rightarrow$ TermoEaux<br>Termo $\rightarrow$ CTE |   |    |

Figura 9: Tabela de produções, parte 4

```

local:
    int i, inf, sup, med, posic, compara;
    logic achou, fimteste;
statements:
    achou <- false; inf <- 1; sup <- ntab;
    while (!achou && sup >= inf) {
        med <- (inf + sup) / 2;
        compara <- 0; fimteste <- false;
        for (i <- 0; !fimteste && compara = 0; i <- i+1) {
            if (palavra[i] < nomes[med,i])
                compara <- ~1;
            else if (palavra[i] > nomes[med,i])
                compara <- 1;
            if (palavra[i] == '\0' || nomes[med,i] == '\0')
                fimteste <- true;
        }
        if (compara = 0)
            achou <- true;
        else if (compara < 0)
            sup <- med - 1;
        else inf <- med + 1;
    }
    if (achou) posic <- med;
    else posic <- ~inf;
    return posic;

} /* Fim da funcao Procura */

/* Funcao para inserir uma palavra na tabela de palavras */

void Inserir (int posic) {

local:
    int i, j; logic fim;
statements:
    ntab <- ntab + 1;
    for (i <- ntab; i >= posic+1; i <- i-1) {
        fim <- false;
        for (j <- 0; !fim; j <- j+1) {
            nomes[i,j] <- nomes[i-1,j];
            if (nomes[i,j] == '\0') fim <- true;
        }
        nocorr[i] <- nocorr[i-1];
    }
    fim <- false;
    for (j <- 0; !fim; j <- j+1) {
        nomes[posic,j] <- palavra[j];
        if (palavra[j] == '\0') fim <- true;
    }
    nocorr[posic] <- 1;

} /* Fim da funcao Inserir */

/* Funcao para escrever a tabela de palavras */

void ExibirTabela () {

local:
    int i; logic fim;
statements:
    write ("          ", "Palavra          ",
           "          Num. de ocorr.");
    for (i <- 1; i <= 50; i <- i+1) write (" ");
    for (i <- 1; i <= ntab; i <- i+1) {
        write ("\n          "); fim <- false;
        for (j <- 0; !fim; j <- j+1) {
            if (nomes[i,j] == '\0') fim <- true;
            else write (nomes[i,j]);
        }
    }
}

```

```

        write ( " | " , nocorr[i]);
    }
} /* Fim da funcao ExibirTabela */

/* Modulo principal */

main {
local:
    int i, posic;
    char c; logic fim;
statements:
    ntab <- 0;
    write ("Nova palavra? (s/n): ");
    read (c);
    while (c = 's' || c = 'S') {
        write ("\nDigite a palavra: ");
        fim <- false;
        for (i <- 0; !fim; i <- i+1) {
            read (palavra[i]);
            if (palavra[i] = '\n') {
                fim <- true;
                palavra[i] <- '\0';
            }
        }
        posic <- Procura ();
        if (posic > 0)
            nocorr[posic] <- nocorr[posic] + 1;
        else
            call Inserir (~posic, i);
            write ("\n\nNova palavra? (s/n): ");
            read (c);
    }
    call ExibirTabela ();
} /* Fim da funcao main */

} /* Fim do programa AnaliseDeTexto */

```

Código 1: Código de entrada para o analisador

```

program AnaliseDeTexto {
    global:
        char nomes[50, 10], palavra[10];
        int ntab, nocorr[50];
        char c;
        logic fim;

    functions:

        int Procura ( ) {
            local:
                int i, inf, sup, med, posic, compara;
                logic achou, fimteste;
            statements:
                achou <- false ;
                inf <- 1 ;
                sup <- ntab ;
                while ( !achou && sup >= inf )
                {
                    med <- ( inf + sup ) / 2 ;
                    compara <- 0 ;
                    fimteste <- false ;
                    for ( i <- 0 ; !fimteste && compara = 0 ; i <- i + 1 )
                    {
                        if ( palavra [i] < nomes [med , i ])
                            compara <- ~ 1 ;
                    }
                }
            return ( achou );
        }
}

```

```

        else
            if ( palavra [i] > nomes [med , i ])
                compara <- 1 ;
            if ( palavra [i] = '\0' || nomes [med , i] = '\0' )
                fimteste <- true ;
        }
        if ( compara = 0 )
            achou <- true ;
        else
            if ( compara < 0 )
                sup <- med - 1 ;
            else
                inf <- med + 1 ;
    }
    if ( achou )
        posic <- med ;
    else
        posic <- ~ inf ;
    return posic ;
}

void Inserir ( int ) {
    local:
        int i, j;
        logic fim;
    statements:
        ntab <- ntab + 1 ;
        for ( i <- ntab ; i >= posic + 1 ; i <- i - 1 )
        {
            fim <- false ;
            for ( j <- 0 ; !fim ; j <- j + 1 )
            {
                nomes [i , j] <- nomes [i - 1 , j ] ;
                if ( nomes [i , j] = '\0' )
                    fim <- true ;
            }
            nocorr [i] <- nocorr [i - 1 ] ;
        }
        fim <- false ;
        for ( j <- 0 ; !fim ; j <- j + 1 )
        {
            nomes [posic , j] <- palavra [j ] ;
            if ( palavra [j] = '\0' )
                fim <- true ;
        }
        nocorr [posic] <- 1 ;
    }
}

void ExibirTabela ( ) {
    local:
        int i;
        logic fim;
    statements:
        write ( "          ", "Palavra          ", "      Num. de ocorr." ) ;
        for ( i <- 1 ; i <= 50 ; i <- i + 1 )
            write ( "—" ) ;
        for ( i <- 1 ; i <= ntab ; i <- i + 1 )
        {
            write ( "\n          " ) ;
            fim <- false ;
            for ( j <- 0 ; !fim ; j <- j + 1 )
            {
                if ( nomes [i , j] = '\0' )
                    fim <- true ;
                else
                    write ( nomes [i , j] ) ;
            }
            write ( " | ", nocorr [i] ) ;
        }
    }
}

```

```

main {
  local:
    int i, posic;
    char c;
    logic fim;
  statements:
    ntab <- 0 ;
    write ( "Nova palavra? (s/n): " ) ;
    read ( c ) ;
    while ( c = 's' || c = 'S' )
    {
      write ( "\nDigite a palavra: " ) ;
      fim <- false ;
      for ( i <- 0 ; !fim ; i <- i + 1 )
      {
        read ( palavra [i] ) ;
        if ( palavra [i] = '\n' )
        {
          fim <- true ;
          palavra [i] <- '\0' ;
        }
      }
      posic <- Procura( ) ;
      if ( posic > 0 )
        nocorr [posic] <- nocorr [posic] + 1 ;
      else
        call Inserir( ~ posic , i ) ;
      write ( "\n\nNova palavra? (s/n): " ) ;
      read ( c ) ;
    }
  call ExibirTabela( ) ;
}
}

```

Código 2: Código de saída do analisador