Linguagem COMP-ITA 2017 ï CES-41/2017

CTA ï ITA ï IEC

1) Introdução

São apresentadas a seguir as especificações da linguagem de programação simplificada **COMP-ITA 2017**, usada como linguagem fonte de um compilador a ser projetado em alguns trabalhos correntes da disciplina CES-41 Compiladores, em 2017. Porém, antes das especificações, é mostrado um programa escrito nesta linguagem, com a finalidade de facilitar a compreensão da estrutura sintática e léxica da mesma.

2) Programa exemplo escrito na linguagem COMP-ITA 2017

O programa a seguir lê palavra por palavra de um texto e produz uma tabela contendo cada palavra não repetida ao lado de seu número de ocorrências nesse texto. Por exemplo, se o texto for:

```
voce pode enganar algumas pessoas em todo o tempo e todas as pessoas
em algum tempo mas nao pode enganar todas as pessoas em todo o tempo
```

então o programa deve exibir uma tabela semelhante a esta:

```
          Palavra              |   Num. de ocorr
-------------------------------------------------
          algum                |          1
          algumas              |          1
          as                   |          2
          e                    |          1
          em                   |          3
          enganar              |          2
          mas                  |          1
          nao                  |          1
          o                    |          2
          pessoas              |          3
          pode                 |          2
          tempo                |          3
          todas                |          2
          todo                 |          2
          voce                 |          1
```

Programa em COMP-ITA 2017:

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

program AnaliseDeTexto ;

/*  Variaveis globais  */

var
     char nomes[50][10], palavra[10];
     int i, ntab, posic, nocorr[50];
     char c; logic fim;

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

function int Procura ();
var
     int i, inf, sup, med, posic, compara;
     logic achou, fimteste;
{
     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;
}
```

```
/*  Procedimento para inserir uma palavra na tabela de palavras  */

procedure Inserir (int posic);
var
     int i, j; logic fim;
{
     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;
}

/*  Procedimento para escrever a tabela de palavras  */

procedure ExibirTabela ();
var
     int i; logic fim;
{
     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]);
     }
}
```

```
/*  Modulo principal  */

{
     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);
          write ("\n\nNova palavra? (s/n): ");
          read (c);
     }
     call ExibirTabela ();
}
```

3) Produções da gramática para a sintaxe de COMP-ITA 2017

Prog : **program** **ID** **SCOLON** Decls SubProgs CompStat

Decls : ε | **var** DeclList

DeclList : Declaration | DeclList Declaration

Declaration : Type ElemList **SCOLON**

Type : **int** | **float** | **char** | **logic**

ElemList : Elem | ElemList **COMMA** Elem

Elem : **ID** DimList

DimList : ε | DimList Dim

Dim : **OPBRAK** **INTCT** **CLBRAK**

```
SubProgs      :   ε | SubProgs SubProgDecl

SubProgDecl   :   Header Decls CompStat

Header        :   FuncHeader | ProcHeader

FuncHeader    :   function Type ID OPPAR CLPAR SCOLON
              |   function Type ID OPPAR ParamList CLPAR
                  SCOLON

ProcHeader    :   procedure ID OPPAR CLPAR SCOLON
              |   procedure ID OPPAR ParamList CLPAR SCOLON

ParamList     :   Parameter | ParamList COMMA Parameter

Parameter     :   Type ID

CompStat      :   OPBRACE StatList CLBRACE

StatList      :   ε | StatList Statement

Statement     :   CompStat | IfStat | WhileStat | RepeatStat
              |   ForStat | ReadStat | WriteStat | AssignStat
              |   CallStat | ReturnStat | SCOLON

IfStat        :   if OPPAR Expression CLPAR Statement ElseStat

ElseStat      :   ε | else Statement

WhileStat     :   while OPPAR Expression CLPAR Statement

RepeatStat    :   repeat Statement while OPPAR Expression
                  CLPAR SCOLON

ForStat       :   for OPPAR Variable ASSIGN Expression SCOLON
                  Expression SCOLON Variable ASSIGN Expression
                  CLPAR Statement

ReadStat      :   read OPPAR ReadList CLPAR SCOLON

ReadList      :   Variable | ReadList COMMA Variable

WriteStat     :   write OPPAR WriteList CLPAR SCOLON

WriteList     :   WriteElem | WriteList COMMA WriteElem

WriteElem     :   STRING | Expression
```

```
CallStat      :    call  ID  OPPAR  CLPAR  SCOLON
              |    call  ID  OPPAR  ExprList  CLPAR  SCOLON

ReturnStat    :    return  SCOLON  |  return  Expression  SCOLON

AssignStat    :    Variable  ASSIGN  Expression  SCOLON

ExprList      :    Expression  |  ExprList  COMMA  Expression

Expression    :    AuxExpr1  |  Expression  OR  AuxExpr1

AuxExpr1      :    AuxExpr2  |  AuxExpr1  AND  AuxExpr2

AuxExpr2      :    AuxExpr3  |  NOT  AuxExpr3

AuxExpr3      :    AuxExpr4  |  AuxExpr4  RELOP  AuxExpr4

AuxExpr4      :    Term  |  AuxExpr4  ADOP  Term

Term          :    Factor  |  Term  MULTOP  Factor

Factor        :    Variable  |  INTCT  |  FLOATCT  |  CHARCT
              |    true  |  false  |  NEG  Factor
              |    OPPAR  Expression  CLPAR  |  FuncCall

Variable      :    ID  SubscrList

SubscrList    :    ε  |  SubscrList  Subscript

Subscript     :    OPBRAK  AuxExpr4  CLBRAK

FuncCall      :    ID  OPPAR  CLPAR  |  ID  OPPAR  ExprList  CLPAR
```

4) Convenções para a descrição das produções da gramática de COMP-ITA 2017

Não-terminais:
Estilo: normal
1ª letra: maiúscula
Outras letras: minúsculas e/ou maiúsculas

Palavras reservadas:
Estilo: negrito
Letras: minúsculas

Outros átomos (identificadores, constantes, operadores, separadores)**:**
Estilo: negrito

Letras: maiúsculas

Lado direito de produção vazia:

Estilo: negrito

Caractere: ε

5) Especificações léxicas de COMP-ITA 2017

5.1 ï Palavras reservadas

call	**char**	**else**	**false**	**float**	**for**
function	**if**	**int**	**logic**	**procedure**	**program**
read	**repeat**	**return**	**true**	**var**	**while**
write					

5.2 ï Sintaxes

Descrição	Tipo	Sintaxe
Identificador	**ID**	**Letra (Letra | Dígito)***
Constante inteira	**INTCT**	**(Dígito) +**
Constante caractere	**CHARCT**	**á(\)? Caractereô**
Constante real	**FLOATCT**	**(Dígito)+ . (Dígito)* ((E | e) (+ | -)? (Dígito) +)?**
Cadeia de caracteres	**STRING**	**ñ ((\)? Caractere)* ò**

5.3 ï Operadores

Classe	Tipo	Operadores
Or lógico	**OR**	**||**
And lógico	**AND**	**&&**
Not lógico	**NOT**	**!**
Relacionais	**RELOP**	**< <= > >= == !=**
Aditivos	**ADOP**	**+ -**
Multiplicativos	**MULTOP**	*** / %**
Negadores	**NEG**	**~**

5.4 ï Separadores e outros

Descrição	Símbolo	Tipo	Descrição	Símbolo	Tipo
Abre-parêntesis	**(**	**OPPAR**	**Abre-chave**	**{**	**OPBRACE**
Fecha-parêntesis	**)**	**CLPAR**	**Fecha-chave**	**}**	**CLBRACE**
Abre-colchete	**[**	**OPBRAK**	**Ponto e vírgula**	**;**	**SCOLON**
Fecha-colchete	**]**	**CLBRAK**	**Vírgula**	**,**	**COMMA**
			Atribuição	**=**	**ASSIGN**

5.5 ï Comentários nos programas:

Tudo o que estiver entre **/*** e ***/**

6) Especificações semânticas de COMP-ITA 2017

- Qualquer identificador deve ser declarado antes de usado.
- Um identificador não pode estar declarado mais de uma vez dentro de um subprograma, ou como global, mas pode estar declarado ao mesmo tempo como global e num subprograma qualquer, ou em dois ou mais subprogramas quaisquer.
- Identificadores podem ser do tipo nome de programa, nome de variável, nome de função ou nome de procedimento.
- Um subprograma não pode ter o mesmo nome que o de uma variável global.
- Variáveis escalares, expressões e elementos de variáveis indexadas podem ser do tipo inteiro, real, caractere ou lógico.
- A constante inteira usada no dimensionamento de uma variável indexada deve ser maior do que zero.
- Toda variável escalar e ao menos um elemento de cada variável indexada deve ser inicializado e referenciado pelo menos uma vez no programa.
- Deve haver compatibilidade entre os tipos dos dois lados de um comando de atribuição, conforme a seguinte tabela:

Tipo do lado esquerdo	**Tipo do lado direito**
Inteiro	Inteiro ou Caractere
Real	Inteiro, Real ou Caractere
Caractere	Inteiro ou Caractere
Lógico	Lógico

- Variáveis escalares não podem ter subscritos.
- O número de subscritos de uma variável indexada deve ser igual ao seu número de dimensões declarado.
- Os elementos de uma variável indexada só poderão ser atribuídos ou receber atribuição um de cada vez.
- Os elementos de uma variável indexada só poderão ser lidos, ou escritos um de cada vez.
- Os tipos dos resultados das diversas classes de expressões só podem ser os seguintes:

Classe da expressão	Tipo
Aritmética	Inteiro, Real ou Caractere
Relacional	Lógico
Lógica	Lógico

- Os tipos dos operandos admitidos pelos operadores de expressões são os seguintes:

Operadores	Tipos admitidos dos operandos
|| && !	Lógico
< <= > >=	Inteiro, Real ou Caractere
== !=	Todos (se um for lógico o outro também deve ser)
+ - * / ~	Inteiro, Real ou Caractere
%	Inteiro ou Caractere

- As expressões nos cabeçalhos de comandos **if** e **while** e no encerramento de comandos **repeat** devem ser relacionais ou lógicas.
- A variável da inicialização do cabeçalho de um comando **for** deve ser escalar do tipo inteiro ou caractere.
- A variável da atualização do cabeçalho de um comando **for** deve ser a mesma daquela de sua inicialização.
- A primeira e a terceira expressão de um comando **for** deve ser do tipo inteiro ou caractere e a segunda expressão deve ser do tipo lógico.
- A expressão aritmética no subscrito de uma variável indexada deve ser do tipo inteiro ou caractere.
- O identificador de uma chamada de procedimento deve ser do tipo nome de procedimento e o identificador de uma chamada de função deve ser do tipo nome de função.
- Um identificador de variável e de parâmetro deve ser do tipo nome de variável.
- O número de argumentos na chamada de um subprograma deve ser igual ao número de parâmetros do mesmo.
- Deve haver compatibilidade entre um argumento de chamada de um subprograma e seu parâmetro correspondente, conforme a seguinte tabela:

Tipo do parâmetro	Tipo do argumento
Inteiro	Expressão inteira ou caractere
Real	Expressão inteira, real ou caractere
Caractere	Expressão inteira ou caractere
Lógico	Expressão de valor lógico

- Todo comando **retornar** dentro de um procedimento não deve ser seguido de expressão e dentro de uma função deve ser seguido por uma expressão.
- Deve haver compatibilidade entre o tipo de uma função e o tipo da expressão de qualquer comando **retornar** em seu escopo, conforme a seguinte tabela:

Tipo da função	Tipo da expressão retornada
Inteiro	Inteiro ou Caractere
Real	Inteiro, Real ou Caractere
Caractere	Inteiro ou Caractere
Lógico	Lógico

- Subprogramas não são usados como parâmetros ou argumentos de chamada de outros subprogramas.
- A linguagem não admite recursividade.