Instituto Tecnológico de Aeronáutica

Departamento de Engenharia de Software
Divisão de Ciência da Computação
Laboratório de CES-41

Relatório da atividade

Laboratório 4:
Analisador semântico

Alunos:
Lucas França de Oliveira, lucas.fra.oli18@gmail.com
Gabriel Santana Brito, gabriel.brito55@gmail.com
Professor: Fábio Carneiro Mokarzel, mokarzel@ita.br

26 de junho de 2017

1 Resumo

O laboratório 4 da disciplina CES-41/2017.1 tem como prática a implementação do analisador semântico da linguagem COMP-ITA 2017 definida pelo professor em um documento anexado ao roteiro do laboratório. Esta é a terceira etapa da implementação de um compilador desta linguagem, utilizando os analisadores léxico e sintático implementados nos laboratórios anteriores e, também, sendo pré-requisito dos laboratórios posteriores.

2 Objetivos

A atividade de programação teve como objetivos:

- Construção de um analisador semântico para uma linguagem de programação, usando a ferramenta *Yacc*.
- Impressão da tabela de símbolos de um programa e a emissão de possíveis mensagens de erro.

3 Metodologia

O programa foi implementado nos arquivos *build.l* (código *Flex*) e *build.y* (código *Yacc*). A análise léxica foi feita, de forma semelhante aos laboratórios anteriores, foi feita pelo software *Flex* e a sintático-semântica, pelo *Yacc*.

O código dos laboratórios 02 e 03 foram reutilizados e refatorados para a atividade atual. Foram adicionadas diversas subrotinas auxiliares, novas declarações e código em meio às produções da gramática definida no laboratório 03, que não necessitou de novas alterações.

As implementações foram baseadas naquelas dos slides da aula de prática 3 e da aula de teoria 6. Destas aulas, somente a tarefa de subprogramação necessitou de uma implementação inteiramente nova elaborada pelo grupo.

O código pode ser acessado pelo repositório do github: https://github.com/InventiveWeasel/CES41-LAB4.

4 Teste

Foram realizados diversos testes para a validação do programa. Cinco deles serão incluídos neste relatório: três testes de variáveis e de expressões fornecidos pelo professor, o programa completo fornecido pelo professor, o qual sofreu uma pequena alteração para estar correto, e um teste de subprogramação elaborado pelo aluno. Para não prolongar este relatório, somente os dois últimos serão escritos no relatórios.

4.1 ENTRADA: *ptest.dat*

```
program ProgramaTeste ;

var
   int gi; logic gl; char gc;

procedure TesteCorreto(int pi, logic pl,
    char pc);
var int vi; logic vl; char vc; float gi,
    gl, gc;
{
   gi = 1.0;
   gl = 1.0;
```

```
   gc = 1.0;
   if (pl && (vc == 'z'))
      vl = true;
   else vl = false;
   for(pi = 0; pi < vi; pi = pi+1){
      vl = !vl;
   }
   pl = false;
   while(pl){
      pl = false;
   }
   repeat{
      pl = false;
   } while(pl);
}

procedure TesteIncorreto(int pi, logic pl
    , char pc);
var int vi; logic vl; char vc;
{
   call TesteIncorreto();

   if (pi + 10)
      vl = true;
   else vl = false;
   for(pi = 0; vi; vi = vi+1){
      vl = !vl;
   }
   pi = 0;
   while(pi){
      pi = 0;
   }
   repeat{
      pi = 0;
   } while(pi);
}

function int ParamIncorreto(int
    TesteCorreto, logic ProgramaTeste,
    logic v);
var logic v;
{
   return v;
}

function int ParamCorreto(int p);
var int v;
{
   v = 0;
   return v + 0;
}

{
   call ParamCorreto(2);
   call ParamIncorreto(2, false, true);
   call TesteCorreto(ParamCorreto(0),
       false, 'a');
   call TesteCorreto(10, false, 'a');
   call TesteIncorreto(10.0, 2.3, 2.3);
   call TesteIncorreto(10, false);
   call TesteIncorreto(10);
   call TesteIncorreto();
   gi = 1;
}
```

4.2 SAÍDA: *ptestout.txt*

```
program ProgramaTeste;

var
    int gi;
    logic gl;
    char gc;
procedure TesteCorreto(int pi, logic pl,
    char pc);
var
    int vi;
    logic vl;
    char vc;
    float gi, gl, gc;
{
    gi = 1;
    gl = 1;
    gc = 1;
    if(pl && (vc == 'z'))
        vl = true;
    else
        vl = false;
    for(pi = 0; pi < vi; pi = pi + 1)
    {
        vl = !vl;
    }
    pl = false;
    while(pl)
    {
        pl = false;
    }
    repeat
    {
        pl = false;
    }
    while(pl);
}
procedure TesteIncorreto(int pi, logic pl
    , char pc);
var
    int vi;
    logic vl;
    char vc;
{
    call TesteIncorreto();
```

```
***** Erro de linguagem: Proibida
    recursividade. *****



***** Incompatibilidade: Quantidade de
    parametros nao compativel. *****

    if(pi + 10)


***** Incompatibilidade: Expressao do
    comando 'if' deve ser logica *****

        vl = true;
    else
        vl = false;
    for(pi = 0; vi; vi = vi + 1)


***** Incompatibilidade: Variavel de
    atualizacao deve ser a mesma de
    inicializacao *****



***** Incompatibilidade: Expressoes de
    tipo inadequado *****

    {
        vl = !vl;
    }
    pi = 0;
    while(pi)


***** Incompatibilidade: Expressao do
    comando 'while' deve ser logica *****

    {
        pi = 0;
    }
    repeat
    {
        pi = 0;
    }
    while(pi);


***** Incompatibilidade: Expressao do
    comando 'do repeat' deve ser logica
    *****

}
function int ParamIncorreto(int
    TesteCorreto, logic ProgramaTeste,
    logic v);
var
    logic v
***** Declaracao Repetida: v *****

;
{
    return v;


***** Incompatibilidade: [return]
    Compatibilidade de inteiro: esperado
    inteiro ou caractere. *****

}
function int ParamCorreto(int p);
var
    int v;
{
    v = 0;
    return v + 0;
}
{
    call ParamCorreto(2);
    call ParamIncorreto(2, false, true);
    call TesteCorreto(ParamCorreto(0),
        false, 'a');
    call TesteCorreto(10, false, 'a');
    call TesteIncorreto(10, 2.3, 2.3);


***** Incompatibilidade: [param]
    Compatibilidade de caractere:
    esperado inteiro ou caractere. *****



***** Incompatibilidade: [param]
    Compatibilidade de logico: esperado
    logico. *****



***** Incompatibilidade: [param]
    Compatibilidade de inteiro: esperado
    inteiro ou caractere. *****

    call TesteIncorreto(10, false);


***** Incompatibilidade: Quantidade de
    parametros nao compativel. *****

    call TesteIncorreto(10);


***** Incompatibilidade: Quantidade de
    parametros nao compativel. *****

    call TesteIncorreto();
```

```
***** Incompatibilidade: Quantidade de
    parametros nao compativel. *****

    gi = 1;
}
gi do escopo TesteCorreto nao
    referenciado
gi do escopo ProgramaTeste nao
    referenciado
gl do escopo TesteCorreto nao
    referenciado
gl do escopo ProgramaTeste nao
    referenciado
gl do escopo ProgramaTeste nao
    inicializado
vc do escopo TesteIncorreto nao
    referenciado
vc do escopo TesteIncorreto nao
    inicializado
vc do escopo TesteCorreto nao
    inicializado
vi do escopo TesteCorreto nao
    inicializado
gc do escopo TesteCorreto nao
    referenciado
gc do escopo ProgramaTeste nao
    referenciado
gc do escopo ProgramaTeste nao
    inicializado


   TABELA DE SIMBOLOS:

Classe 1:
  gi(IDVAR REAL), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 0

  gi(IDVAR INTEIRO), escopo:
      ProgramaTeste
   Inicializado: 1, Referenciado: 0

Classe 3:
  v(IDVAR INTEIRO), escopo: ParamCorreto
   Inicializado: 1, Referenciado: 1

  v(IDVAR LOGICO), escopo: ParamIncorreto
   Inicializado: 1, Referenciado: 1

Classe 4:
  pc(IDVAR CARACTERE), escopo:
      TesteIncorreto
   Inicializado: 1, Referenciado: 1

  gl(IDVAR REAL), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 0

  pc(IDVAR CARACTERE), escopo:
      TesteCorreto
   Inicializado: 1, Referenciado: 1

  gl(IDVAR LOGICO), escopo: ProgramaTeste
   Inicializado: 0, Referenciado: 0

Classe 8:
  ProgramaTeste(IDVAR LOGICO), escopo:
      ParamIncorreto
   Inicializado: 1, Referenciado: 1

  ProgramaTeste(IDPROG NAOVAR), escopo:
      null
   Funcoes: ParamCorreto(INTEIRO)
       ParamIncorreto(INTEIRO)
       TesteIncorreto(NAOVAR)
       TesteCorreto(NAOVAR)
   Variaveis: gc(CARACTERE) gl(LOGICO)
       gi(INTEIRO)

Classe 9:
  TesteCorreto(IDVAR INTEIRO), escopo:
      ParamIncorreto
   Inicializado: 1, Referenciado: 1

  TesteCorreto(IDPROC NAOVAR), escopo:
      ProgramaTeste
   Parametros (3): pc(CARACTERE) pl(
       LOGICO) pi(INTEIRO)
   Variaveis: gc(REAL) gl(REAL) gi(REAL)
       vc(CARACTERE) vl(LOGICO) vi(
       INTEIRO)

Classe 10:
  vc(IDVAR CARACTERE), escopo:
      TesteIncorreto
   Inicializado: 0, Referenciado: 0

  pi(IDVAR INTEIRO), escopo:
      TesteIncorreto
   Inicializado: 1, Referenciado: 1

  vc(IDVAR CARACTERE), escopo:
      TesteCorreto
   Inicializado: 0, Referenciado: 1

  pi(IDVAR INTEIRO), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 1

Classe 12:
  ParamCorreto(IDFUNC INTEIRO), escopo:
      ProgramaTeste
   Parametros (1): p(INTEIRO)
   Variaveis: v(INTEIRO)

Classe 13:
  pl(IDVAR LOGICO), escopo:
      TesteIncorreto
   Inicializado: 1, Referenciado: 1

  pl(IDVAR LOGICO), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 1
```

```
Classe 16:
  vi(IDVAR INTEIRO), escopo:
      TesteIncorreto
   Inicializado: 1, Referenciado: 1

  vi(IDVAR INTEIRO), escopo: TesteCorreto
   Inicializado: 0, Referenciado: 1

Classe 17:
  TesteIncorreto(IDPROC NAOVAR), escopo:
      ProgramaTeste
   Parametros (3): pc(CARACTERE) pl(
       LOGICO) pi(INTEIRO)
   Variaveis: vc(CARACTERE) vl(LOGICO)
       vi(INTEIRO)

Classe 18:
  gc(IDVAR REAL), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 0

  gc(IDVAR CARACTERE), escopo:
      ProgramaTeste
   Inicializado: 0, Referenciado: 0

Classe 19:
  vl(IDVAR LOGICO), escopo:
      TesteIncorreto
   Inicializado: 1, Referenciado: 1

  vl(IDVAR LOGICO), escopo: TesteCorreto
   Inicializado: 1, Referenciado: 1

Classe 20:
  p(IDVAR INTEIRO), escopo: ParamCorreto
   Inicializado: 1, Referenciado: 1

  ParamIncorreto(IDFUNC INTEIRO), escopo:
      ProgramaTeste
   Parametros (3): v(LOGICO)
      ProgramaTeste(LOGICO) TesteCorreto
      (INTEIRO)
```

4.3 ENTRADA: *completeprog.dat*

Este programa apresentava um único erro: uma variável j não havia sido declarada em uma das funções. Isso foi corrigido para que o programa fosse aceito pelo analisador semântico.

```
/*  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 (int k);
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 (logic k);
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, j; 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 (43%10);
        if (posic > 0)
            nocorr[posic] = nocorr[posic] +
                1;
        else
            call Inserir (true);
            write ("\n\nNova palavra? (s/n):
                ");
            read (c);
    }
    call ExibirTabela ();
}
```

4.4 SAÍDA: *completeprogout.txt*

```
program AnaliseDeTexto;

var
    char nomes[50][10], palavra[10];
    int i, ntab, posic, nocorr[50];
    char c;
    logic fim;
function int Procura(int k);
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;
}
procedure Inserir(logic k);
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;
}
procedure ExibirTabela();
var
        int i, j;
        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]);
        }
}
{
        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(43 % 10);
            if(posic > 0)
                nocorr[posic] = nocorr[posic]
                     + 1;
            else
                call Inserir(true);
            write("\n\nNova palavra? (s/n): "
                );
            read(c);
        }
        call ExibirTabela();
}
c do escopo AnaliseDeTexto nao
    inicializado


   TABELA DE SIMBOLOS:

Classe 0:
  ExibirTabela(IDPROC NAOVAR), escopo:
      AnaliseDeTexto
    Variaveis: fim(LOGICO) j(INTEIRO) i(
        INTEIRO)

Classe 3:
  compara(IDVAR INTEIRO), escopo: Procura
    Inicializado: 1, Referenciado: 1

Classe 7:
  c(IDVAR CARACTERE), escopo:
      AnaliseDeTexto
    Inicializado: 0, Referenciado: 1

  ntab(IDVAR INTEIRO), escopo:
      AnaliseDeTexto
    Inicializado: 1, Referenciado: 1

  palavra(IDVAR CARACTERE), escopo:
      AnaliseDeTexto
    Inicializado: 1, Referenciado: 1
    Array: (1 dimensoes): 10

Classe 11:
  med(IDVAR INTEIRO), escopo: Procura
```

```
    Inicializado: 1, Referenciado: 1

Classe 13:
  i(IDVAR INTEIRO), escopo: ExibirTabela
   Inicializado: 1, Referenciado: 1

  i(IDVAR INTEIRO), escopo: Inserir
   Inicializado: 1, Referenciado: 1

  posic(IDVAR INTEIRO), escopo: Procura
   Inicializado: 1, Referenciado: 1

  i(IDVAR INTEIRO), escopo: Procura
   Inicializado: 1, Referenciado: 1

  posic(IDVAR INTEIRO), escopo:
      AnaliseDeTexto
   Inicializado: 1, Referenciado: 1

  i(IDVAR INTEIRO), escopo:
      AnaliseDeTexto
   Inicializado: 1, Referenciado: 1

Classe 14:
  j(IDVAR INTEIRO), escopo: ExibirTabela
   Inicializado: 1, Referenciado: 1

  j(IDVAR INTEIRO), escopo: Inserir
   Inicializado: 1, Referenciado: 1

  fimteste(IDVAR LOGICO), escopo: Procura
   Inicializado: 1, Referenciado: 1

Classe 15:
  k(IDVAR LOGICO), escopo: Inserir
   Inicializado: 1, Referenciado: 1

  k(IDVAR INTEIRO), escopo: Procura
   Inicializado: 1, Referenciado: 1

  nocorr(IDVAR INTEIRO), escopo:
      AnaliseDeTexto
   Inicializado: 1, Referenciado: 1
   Array: (1 dimensoes): 50

Classe 17:
  fim(IDVAR LOGICO), escopo: ExibirTabela
   Inicializado: 1, Referenciado: 1

  fim(IDVAR LOGICO), escopo: Inserir
   Inicializado: 1, Referenciado: 1

  fim(IDVAR LOGICO), escopo:
      AnaliseDeTexto
   Inicializado: 1, Referenciado: 1

  nomes(IDVAR CARACTERE), escopo:
      AnaliseDeTexto
   Inicializado: 1, Referenciado: 1
   Array: (2 dimensoes): 50 10

Classe 18:
  inf(IDVAR INTEIRO), escopo: Procura
   Inicializado: 1, Referenciado: 1

Classe 19:
  Inserir(IDPROC NAOVAR), escopo:
      AnaliseDeTexto
   Parametros (1): k(LOGICO)
   Variaveis: fim(LOGICO) j(INTEIRO) i(
       INTEIRO)

  Procura(IDFUNC INTEIRO), escopo:
      AnaliseDeTexto
   Parametros (1): k(INTEIRO)
   Variaveis: fimteste(LOGICO) achou(
       LOGICO) compara(INTEIRO) posic(
       INTEIRO) med(INTEIRO) sup(INTEIRO)
        inf(INTEIRO) i(INTEIRO)

Classe 22:
  achou(IDVAR LOGICO), escopo: Procura
   Inicializado: 1, Referenciado: 1

  sup(IDVAR INTEIRO), escopo: Procura
   Inicializado: 1, Referenciado: 1

  AnaliseDeTexto(IDPROG NAOVAR), escopo:
      null
   Funcoes: ExibirTabela(NAOVAR) Inserir
       (NAOVAR) Procura(INTEIRO)
   Variaveis: fim(LOGICO) c(CARACTERE)
       nocorr(INTEIRO) posic(INTEIRO)
       ntab(INTEIRO) i(INTEIRO) palavra(
       CARACTERE) nomes(CARACTERE)
```

5 Conclusão

Ao longo do laboratório, o grupo encontrou algumas dificuldades, como o cuidado com ponteiros $NULL$s gerados por erros semânticos anteriores. Por exemplo: Ao se referenciar uma variável não declarada, a procura de símbolo retorna um ponteiro $NULL$ que deveria representar o símbolo. Erros como esse causaram, em diversas ocasiões, erros em tempo de execução do processo de análise semântica.

Outra dificuldade encontrada foi a complexidade da estrutura de dados que é a tabela de símbolos, implementada com uma hash table em que cada elemento possui diversas listas ligadas na representação de variáveis, parâmetros e funções ou procedimentos. Foram criadas diversas rotinas auxiliares pra modularizar e organizar a operação da tabela de símbolos.

Apesar das dificuldades, o grupo conseguiu implementar todas as restrições semânticas impostas pela linguagem, além de manter todas as restrições léxico-sintáticas dos laboratórios anteriores. Os testes com os programas fornecidos e criados demonstraram o correto funcionamento da detecção de todos os possíveis erros semânticos, o que garante o correto funcionamento do analisador.