Relatório de CES-41 – Compiladores
Laboratório 4

Juan de Castro Pessoa
Thiago Ribeiro Ramos
COMP -14
Professor Mokarzel
16 de Dezembro de 2013

ASSUNTO

Nesse laboratório foi elaborado um analisador semântico, além de um gerador de tabela de símbolos, para a linguagem COMP-ITA-2013, com o auxílio da ferramenta Yacc e do código produzido no laboratório via Flex.

RESULTADOS

Utilizando o programa abaixo:

```
programa teste;

var
      int a, resultado,cx,cy;
      int h[5,6], tos[20];


funcao int fibonnaci(int x);

var int p1, p2, cont, temp;

{
      p1 := 0;
      p2 := 1;
      para(cont:=0;cont<x;cont:=cont+1){
            temp := p2;
            p2 := p1+p2;
            p1 := temp;
      }
      retornar p1;
}

procedimento swap(int a1, int a2);

var int temp;
{
      temp := tos[a1];
      tos[a1] := tos[a2];
      tos[a2] := temp;
      retornar;
}

procedimento bubblesort();

var
      int cx, cy;
{
      para(cx:=0;cx<20;cx:=cx+1){
```

```
            para(cy:=0;cy<19-cx;cy:=cy+1){
                  se(tos[cy] > tos[cy+1]){
                        chamar swap(cy,cy+1);
                  }
            }
      }
      retornar;
}

procedimento imprimirtabela();

var int i;

{
      para(i:=0;i<19;i:=i+1){
            escrever("tos[i]"," ");
      }
      retornar;
}

procedimento preenchertabelah(int inicial, int razaox, int razaoy);

var int cx, cy;
{
      para(cx:=0;cx<5;cx:=cx+1){
            para(cy:=0;cy<6;cy:=cy+1){
                  h[cx,cy] := inicial+cx*razaox+cy*razaoy;
            }
      }
      retornar;
}

{
      ler(a);
      escrever("O ",a," termo de Fibonnaci eh", fibonnaci(a));
      cy:=1;
      para(cx:=0;cx<20;cx:=cx+1){
            tos[cx] := cy;
            cy := (cy*17)%23;
      }
      chamar imprimirtabela();
      chamar bubblesort();
      chamar imprimirtabela();
      resultado := fibonnaci(10-a);
      se(h[1,3]>~5){
            escrever("Resultado eh",resultado);
      }
}
```

Em cada item adicionamos o escopo no qual ele está inserido, para atender aos requisitos de uma variável com o mesmo nome poder ser declarada em duas funcoes/procedimentos diferentes, além do módulo principal.
Temos como resultado:

```
   TABELA  DE  SIMBOLOS:

Classe 0:
  (p1, IDVAR, INTEGER, 1, 1, fibonnaci)
Classe 1:
  (temp, IDVAR, INTEGER, 1, 1, swap)
```

```
  (temp, IDVAR, INTEGER, 1, 1, fibonnaci)
  (p2, IDVAR, INTEGER, 1, 1, fibonnaci)
Classe 5:
  (x, IDVAR, INTEGER, 1, 1, fibonnaci)
  (a, IDVAR, INTEGER, 1, 1, teste)
Classe 6:
  (swap, IDPROC, EH ARRAY
      ndims = 2, dimensoes:[  INTEGER  INTEGER], teste)
Classe 8:
  (a1, IDVAR, INTEGER, 1, 1, swap)
Classe 9:
  (a2, IDVAR, INTEGER, 1, 1, swap)
Classe 12:
  (cx, IDVAR, INTEGER, 1, 1, preenchertabelah)
  (cx, IDVAR, INTEGER, 1, 1, bubblesort)
  (h, IDVAR, INTEGER, 1, 1, EH ARRAY
      ndims = 2, dimensoes:[  5  6], teste)
  (cx, IDVAR, INTEGER, 1, 1, teste)
Classe 13:
  (cy, IDVAR, INTEGER, 1, 1, preenchertabelah)
  (i, IDVAR, INTEGER, 1, 1, imprimirtabela)
  (cy, IDVAR, INTEGER, 1, 1, bubblesort)
  (cy, IDVAR, INTEGER, 1, 1, teste)
  (resultado, IDVAR, INTEGER, 1, 1, teste)
Classe 16:
  (inicial, IDVAR, INTEGER, 1, 1, preenchertabelah)
Classe 17:
  (razaox, IDVAR, INTEGER, 1, 1, preenchertabelah)
  (fibonnaci, IDFUNC, INTEGER, TEM PARAMETROS
      ndims = 1, dimensoes:[  INTEGER], teste)
Classe 18:
  (razaoy, IDVAR, INTEGER, 1, 1, preenchertabelah)
  (imprimirtabela, IDPROC, teste)
  (bubblesort, IDPROC, teste)
Classe 20:
  (tos, IDVAR, INTEGER, 1, 1, EH ARRAY
      ndims = 1, dimensoes:[  20], teste)
  (teste, IDPROG)
Classe 21:
  (preenchertabelah, IDPROC, TEM PARAMETROS
      ndims = 3, dimensoes:[  INTEGER  INTEGER  INTEGER], teste)
Classe 22:
  (cont, IDVAR, INTEGER, 1, 1, fibonnaci)
```

Agora, vamos mostrar cada item da análise semântica fazendo pequenas alteraçoes no programa acima. Vou deixar o programa imprimindo o código original por enquanto para mostrar melhor o erro e não precisar colocar também o código original toda vez:

- **Qualquer identificador deve ser declarado antes de usado.**

```
programa teste;

var
      int p;

{
      a

***** Identificador Nao Declarado: a *****
```

```
 := 3;
}
```

- **Um identificador não pode ser declarado mais de uma vez dentro de um subprograma, ou no programa principal, mas pode estar declarado ao mesmo tempo nesse último e num subprograma qualquer, ou em dois ou mais subprogramas quaisquer.**

```
procedimento imprimirtabela();

var
      int i;
      real i

***** Declaracao Repetida: i *****

;
```

O caso onde duas variáveis com mesmo nome e escopos diferentes já está mostrada no programa principal.

- **Identificadores podem ser do tipo nome de programa, nome de variável, nome de função ou nome de procedimento.**

Basta observar o programa principal acima para ver tais casos

- **Variáveis escalares, expressões e elementos de variáveis indexadas podem ser do tipo inteiro, real, caractere ou lógico.**

```
var
      int a, resultado,cx,cy;
      int h[5,6], tos[20];
      cadeia l;

Syntax Error!
```

- **A constante inteira usada no dimensionamento de uma variável indexada deve ser maior do que zero.**

```
var
      int a, resultado, cx, cy;
      int h[0

***** Esperado: Valor inteiro positivo *****

, 6], tos[20];
```

- **Toda variável escalar e ao menos um elemento de cada variável indexada deve ser inicializado e referenciado pelo menos uma vez no programa.**

```
programa teste;

var
      int a, resultado, cx, cy;
      real h[1, 6], tos[20];
      carac c;
```

```
{
      a := 0;
      resultado := a;
      tos[3]  := 4.200000;
      c := cx;
}

   TABELA  DE  SIMBOLOS:

Classe 5:
  (a, IDVAR, INTEGER, 1, 1, teste)
Classe 7:
  (c, IDVAR, CHAR, 1, 1, teste)
Classe 12:
  (h, IDVAR, FLOAT, 0, 0, EH ARRAY
      ndims = 2, dimensoes:[  1  6], teste)
  (cx, IDVAR, INTEGER, 0, 1, teste)
Classe 13:
  (cy, IDVAR, INTEGER, 0, 0, teste)
  (resultado, IDVAR, INTEGER, 1, 1, teste)
Classe 20:
  (tos, IDVAR, FLOAT, 1, 1, EH ARRAY
      ndims = 1, dimensoes:[  20], teste)
  (teste, IDPROG)


***** Identificador nao inicializado: h *****

***** Identificador nao referenciado: h *****

***** Identificador nao inicializado: cx *****

***** Identificador nao inicializado: cy *****

***** Identificador nao referenciado: cy *****
```

- **Deve haver compatibilidade entre os tipos dos dois lados de um comando de atribuição**

```
programa teste;

var
      int a;
      real b;
      carac c;
      logic d;

{
      a := 1;
      a := 0.200000;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      a := 'c';
      a := falso;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****
```

```
      b := 1;
      b := 0.200000;
      b := 'c';
      b := falso;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      c := 1;
      c := 0.200000;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      c := 'c';
      c := falso;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      d := 1;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      d := 0.200000;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      d := 'c';


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      d := falso;
}
```

- **Variáveis escalares não podem ter subscritos.**

```
programa teste;

var
      int a;

{
      a[1]

***** Nao Esperado: Subscrito(s) *****

 := 1;
      a[0,2]

***** Nao Esperado: Subscrito(s) *****

 := 2;
      a[a]

***** Nao Esperado: Subscrito(s) *****
```

```
 := 3;
}
```

- **O número de subscritos de uma variável indexada deve ser igual ao seu número de dimensões.**

```
programa teste;

var
      int a[3, 4, 5];
      real h[1, 2];

{
      a

***** Esperado: Subscrito(s) *****

 := 0;
      a[1]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 := 1;
      a[0,2]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 := 2;
      a[1,2,3]  := 3;
      a[1,2,3,4]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 := 5;
      h[1]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 := 1.800000;
      h[0,2]  := 2.200000;
      h[1,2,3,4]

***** Incompatibilidade: Numero de subscritos incompativel com declaracao *****

 := 5;
}
```

- **Os elementos de uma variável indexada de qualquer tipo só poderão ser lidos, escritos ou atribuídos ou receber atribuição um de cada vez.**

Esse termo segue diretamente do acima: como cada variável tem que ter seu tamanho correto, um termo acaba sendo modificado por vez.

- **Os tipos dos resultados das diversas classes de expressões**

```
programa teste;
```

```
var
      int a;
      logic l, l1, l2;

{
      a := 0;
      l := 1 + 2;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      l := a - 3;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      a := l || falso;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      a := l && verdade;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

      a := 1 < 3;


***** Incompatibilidade: Lado direito de comando de atribuicao improprio *****

}
```

- **Os tipos dos operandos admitidos pelos operadores**

```
programa teste;

var
      int a;
      logic l;
      real r;
      carac c;

{
      a := 0;
      a := 1 + l

***** Incompatibilidade: Operando improprio para operador aritmetico de
adicao/subtracao *****

;
      a := a * c;
      a := 1 / falso

***** Incompatibilidade: Operando improprio para operador aritmetico
multiplicacao/divisao *****

;
      l := a && verdade
```

```
***** Incompatibilidade: Operando improprio para operador de AND *****

;
      l := c || falso

***** Incompatibilidade: Operando improprio para operador de OR *****

;
      l := !r

***** Incompatibilidade: Operando improprio para operador de negacao *****

;
      l := r > verdade

***** Incompatibilidade: Operando improprio para operador de
maior/menor/maig/menig *****

;
      l := r <= l

***** Incompatibilidade: Operando improprio para operador de
maior/menor/maig/menig *****

;
      l := verdade != c

***** Incompatibilidade: Operando improprio para operador de igual/diferente
*****

;
      l := r = falso

***** Incompatibilidade: Operando improprio para operador de igual/diferente
*****

;
}
```

- **As expressões nos cabeçalhos de comandos se, enquanto e para e no encerramento de comandos repetir devem ser relacionais ou lógicas.**

```
programa teste;

var
      int a;
      logic l;
      real r;
      carac c;

{
      enquanto (a + 3

***** Incompatibilidade: Expressao deveria ser logica *****

)
            {
                  ler(c);
            }
      para(a := 0; r;
```

```
***** Incompatibilidade: Expressao deveria ser logica *****

a := a + 1)
            {
                  ler(c);
            }
      se(3

***** Incompatibilidade: Expressao deveria ser logica *****

)
            {
                  ler(c);
            }
      repetir
            {
                  ler(c);
            }
      enquanto(r * a

***** Incompatibilidade: Expressao deveria ser logica *****

);
}
```

- **A variável da inicialização do cabeçalho de um comando para deve ser escalar do tipo inteiro ou caractere.**

```
programa teste;

var
      int a;
      logic l;
      real r;
      carac c;

{
      para(r :=

***** Incompatibilidade: Variavel inicializada do PARA deveria ser inteira ou
caractere *****

0; r < 3; r := r + 1.200000)
            {
                  ler(c);
            }
      para(l :=

***** Incompatibilidade: Variavel inicializada do PARA deveria ser inteira ou
caractere *****

verdade; l; l := !l)
            {
                  ler(c);
            }
}
```

- **A variável da atualização do cabeçalho de um comando para deve ser a mesma daquela de sua inicialização.**

```
programa teste;

var
      int a;
      real r;
      carac c;

{
      para(a := 0; a < 3; r :=

***** Incompatibilidade: Variavel atualizada nao eh a mesma inicializada *****

r + 1.200000)
            {
                  ler(c);
            }
}
```

- **A expressão aritmética no subscrito de uma variável indexada deve ser do tipo inteiro ou caractere.**

```
var
      int a, h[5, 6];
      logic l;
      real r;
      carac c;

{
      h[a,a + 1]  := 3;
      h[c,3]  := a;
      h[r

***** Incompatibilidade: Tipo inadequado para subscrito *****

,a]  := c;
      h[1.300000

***** Incompatibilidade: Tipo inadequado para subscrito *****

,0.900000

***** Incompatibilidade: Tipo inadequado para subscrito *****

]  := 8;
}
```

- **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.**

```
var
      int a, h[5, 6];
      logic l;
      real r;
      carac c;

procedimento b();
{
```

```
}

funcao int d();

var
      int k;

{
}

{
      chamar b();
      a := d();
      a := a()

***** Identificador de Tipo Inadequado: a *****

;
      chamar r();


***** Identificador de Tipo Inadequado: r *****

}
```

- **Um identificador de variável deve ser do tipo nome de variável.**

```
programa teste;

var
      int a;

procedimento b();
{
}

funcao int d();

var
      int k;

{
}

{
      b

***** Identificador de Tipo Inadequado: b *****

 := a + 1;
      d

***** Identificador de Tipo Inadequado: d *****

 := 3;
}
```

- **O número de argumentos na chamada de um subprograma deve ser igual ao número de parâmetros do mesmo.**

```
programa teste;

var
      int a;

procedimento b();
{
      retornar;
}

funcao int c();

var
      int k;

{
      retornar 1;
}

funcao int d(int a, int b, int c);
{
      retornar a;
}

procedimento e(real a, int b, carac c);
{
      retornar;
}

{
      chamar b();
      chamar b(1);


***** Nao Esperado: Funcao/Procedimento nao espera parametros(s) *****

      chamar b(1, 2, 4);


***** Nao Esperado: Funcao/Procedimento nao espera parametros(s) *****

      a := c(3)

***** Nao Esperado: Funcao/Procedimento nao espera parametros(s) *****

;
      a := d(1)

***** Incompatibilidade: Numero de parametros incompativel com declaracao *****

;
      a := d(1, 4, 5);
      a := d(1, 6, 9, 1)

***** Incompatibilidade: Numero de parametros incompativel com declaracao *****

;
      chamar e(3.100000, 3, 'a');
      chamar e(3, 'a');


***** Incompatibilidade: Numero de parametros incompativel com declaracao *****
```

```
      chamar e(3.100000, 3, 'a', 5);


***** Incompatibilidade: Numero de parametros incompativel com declaracao *****

}
```

- **Deve haver compatibilidade entre um argumento de chamada de um módulo e seu parâmetro correspondente.**

```
programa teste;

var
      int a;

funcao int d(int a, int b, int c);
{
      retornar a;
}

procedimento e(real a, int b, carac c);
{
      retornar;
}

procedimento b(logic v);
{
}

{
      a := d(1, 4, 'c');
      a := d(1, 6.400000

***** Incompatibilidade: Parametro incompativel com tipo *****

, 9);
      chamar e(3, 3, 'a');
      chamar b(4

***** Incompatibilidade: Parametro incompativel com tipo *****

);
      chamar e(falso

***** Incompatibilidade: Parametro incompativel com tipo *****

, 'a', 5.400000

***** Incompatibilidade: Parametro incompativel com tipo *****

);
}
```

- **Todo comando retornar dentro de um procedimento não deve ser seguido de expressão e dentro de uma função deve ser seguido por uma expressão.**

```
programa teste;
funcao int d(int a, int b, int c);
```

```
{
      retornar;


***** Esperado: Funcao precisa de variavel sendo retornada *****

}

procedimento b(logic v);
{
      retornar falso

***** Nao Esperado: Procedimento nao aceita variavel sendo retornada *****

;
}

{
}
```

- **Deve haver compatibilidade entre o tipo de uma função e o tipo da expressão de qualquer comando retornar em seu escopo**

```
programa teste;
funcao int d(int a, int b, int c);
{
      retornar 3.400000

***** Incompatibilidade: Tipos incompativeis para retorno *****

;
}

funcao logic b(logic v);
{
      retornar 'c'

***** Incompatibilidade: Tipos incompativeis para retorno *****

;
}

funcao real a();
{
      retornar falso

***** Incompatibilidade: Tipos incompativeis para retorno *****

;
}

{
}
```

- **Subprogramas não são usados como parâmetros ou argumentos de chamada de outros subprogramas.**

```
programa teste;
```

```
var
      int p;

funcao int d(int a, int b, int c);
{
      retornar 3;
}

funcao logic b(logic v);
{
      retornar falso;
}

procedimento a();
{
      retornar;
}

procedimento c(int l);
{
}

{
      chamar c(a

***** Identificador de Tipo Inadequado: a *****



***** Incompatibilidade: Parametro incompativel com tipo *****

);
      p := d(3, 4, d

***** Identificador de Tipo Inadequado: d *****



***** Esperado: Subscrito(s) *****

);
}
```

- **A linguagem não admite recursividade.**

O modo de resolver esse problema na linguagem é feito da seguinte maneira: Como uma função não pode chamar outra que não tenha sido declarado depois dela (porque isso já causa um “Nao Declarado”), não é preciso nesse momento se incomodar com duas funções chamando uma a outra, ou algum ciclo de funções, porque isso não tem como ocorrer sem um “Nao Declarado”. Então o único caso de recursividade que falta seria uma função chamando ela mesma.

```
programa teste;

var
      int p;

funcao int d(int a, int b, int c);
{
      chamar x();
```

```
***** Identificador Nao Declarado: x *****

}

funcao logic b(logic v);
{
      retornar b(falso)

***** Recursividade: Procedimento/funcao chamando ela mesma *****

;
}

procedimento x();

var
      int l;

{
      l := d(3, 3, 3);
      retornar;
}

procedimento c(int l);
{
      chamar c(

***** Recursividade: Procedimento/funcao chamando ela mesma *****

3);
      retornar;
}

{
}
```