Introdução à Compilação
Entrega 3

Bianca Madoka Shimizu Oe
Luis Fernando Dorelli de Abreu
Rafael Umino Nakanishi

1 Objetivo

O objetivo dessa parte final do projeto da disciplina é a implementação do analisador semântico para o compilador para a linguagem `LALG` disponibilizada para os alunos e a geração de código intermediário. A seguir serão discutidos a abordagem utilizada para lidar com a tabela de símbolos e os tratamentos de erros que são gerados pelo compilador. Em seguida estão listados o meio para compilar o código e o método de executá-lo e alguns exemplos que foram utilizados para testar a compilação e suas respectivas saídas.

2 Decisões de projeto

2.1 Implementação

A implementação do compilador para a linguagem `LALG` foi feita utilizando o `yacc` (*Yet Another Compiler-Compiler*) em linguagem de programação `C`. Essa decisão foi influenciada principalmente devido à possibilidade de continuidade a partir da entrega anterior.

Para armazenar os atributos, utilizamos o recurso proporcionado pelo `yacc`, que consiste na definição de um tipo, que pode ser uma estrutura. Para acessá-los, utilizamos o símbolo reservado $. Uma variável desse tipo é criada sempre que um token é obtido pelo analizados léxico.

2.2 Tabela de símbolos

A tabela de símbolos utilizada nessa implementação é baseado na estrutura de dados *trie*. Nesse caso, cada nó da árvore armazena um caracter correspondente ao nome do identificador. Ao se chegar ao fim da cadeia de caracteres, o nó correspondente armazena os atributos do identificador:

- `type`: tipo do identificador (`integer` ou `real`)
- `category`: categoria a que pertence (constante ou variável)
- `scope`: escopo do identificador
- `ival`: valor da variável caso seja um número inteiro
- `rval`: valor da variável caso seja um número real
- `paramQty`: quantidade de parâmetros, no caso de procedimentos
- `line`: linha em que foi declarado
- `parameters`: lista de parâmetros, no caso de procedimentos
- `name`: nome completo do identificador, com um máximo de 20 caracteres
- `codeLine`: auxilia na resolução de desvios durante a geração de código intermediário

- `address`: endereço da variável no programa

Para resolver problemas de escopo, foram criadas diversas tabelas de símbolos separadas, uma para cada escopo. Quando o analisador volta de um escopo (retorna da declaração de um procedimento), deleta-se a tabela de símbolos do escopo anterior.

2.3 Tratamento de erros

Os seguintes erros são tratados pela análise semântica.

- Variável ou procedimento não declarado
- Variável ou procedimento declarado mais de uma vez
- Incompatibilidade de parâmetros formais e reais: número, ordem e tipo
- Atribuição de um real a um inteiro
- Divisão entre números não inteiros
- `readln` e `writeln` com parâmetros de tipo diferentes

2.4 Geração de código

A geração de código pelo compilador é realizada em paralelo com a análise semântica e é interrompida caso seja encontrado algum erro de natureza léxica, sintática e/ou semântica.

Consideramos que o código inicia na linha 0 e os códigos que são possivelmente gerados podem ser encontrados no Quadro 1. Todos os comando são os mesmos vistos em sala de aula, ao longo da disciplina.

CRCT	CRVL	SOMA	SUBT	MULT
INVE	CONJ	DISJ	NEGA	CPME
CPMA	CPIG	CDES	CPMI	CMAI
ARMZ	DSVI	DSVF	LEIT	IMPR
ALME	INPP	PARA	PUSHER	CHPR
DESM	RTPR	COPVL	PARAM	DIVI

Tabela 1: Códigos possivelmente gerados pela geração de código

Observe que a tabela de símbolos implementada não armazena os valores das constantes, mas apenas substitui os mesmos no código a ser gerado.

3 Compilação

Para compilação do projeto em sistemas *UNIX*, execute o arquivo `Makefile` disponibilizado por meio do comando `make`.

Para o sistema operacional *Windows*, execute a seguinte sequência de comandos:

```
yacc -d -v yacc.yacc --verbose
flex lalg.l
gcc lex.yy.c trie.c symbolTable.c -lfl -o t3.exe
```

Para execução do analisador sintático utilize os comandos:

```
t3.exe <nome_do_arquivo> <nome_do_arquivo_bytecode>
```

Observe que o analisador só termina quando encontra um *end-of-file*. Caso o arquivo de entrada esteja gramaticamente correto, obtém-se como saída um arquivo com código intermediário referente ao código dado como entrada denominado "*youGiveMe.noName*" caso o usuário não entre com algum nome para tal arquivo.

4 Exemplos

4.1 Exemplo 1

Esse exemplo está gramaticamente correto. E por sinal é o código fornecido na oitava avaliação da disciplina. O programa foi compilado com sucesso e obteve um código intermediário.

Entrada

```
1   program fatorial;
2   var n: integer;
3   procedure fat(x : integer);
4   var resultado: integer;
5   begin
6     resultado := 1;
7     while x > 1 do
8     begin
9       resultado := resultado * x;
10      x := x - 1;
11    end;
12
13    writeln(resultado);
14  end;
15
16  begin
17    readln(n);
18    fat(n);
19  end.
```

Saída gerada

```
0    INPP
1    ALME1
2    DSVI 24
3    COPVL
4    ALME1
5    CRCT 1
6    ARMZ 3
7    CRVL 2
8    CRCT 1
9    CPMA
10   DSVF 20
11   CRVL 3
12   CRVL 2
13   MULT
14   ARMZ 3
15   CRVL 2
16   CRCT 1
17   SUBT
18   ARMZ 2
19   DSVI 7
20   CRVL 3
21   IMPR
22   DESM 2
23   RTPR
24   LEIT
25   ARMZ 0
26   PUSHER 29
27   PARAM 0
28   CHPR 3
29   PARA
```

4.2 Exemplo 2

Entrada

```
1   prog prog; { program escrito errado }
2   { declaracao de variaveis }
3   const pi=3.14; { constante declarada depois de variavel }
4   var x: integer;
```

```
5     var y:real;
6     { declaracao de procedimento }
7     procedure proc1 (x:real, y:real);
8     begin
9       readln(x,y);
10      if x > 3 then
11        x := 3
12      else
13        x := 4 { faltou ; }
14    end;
15    procedure proc2 (x:integer);
16    begin
17      writeln(x);
18      while x/y > 3 do
19        x := 3x + 2;
20    end;
21    begin
22      proc1 (x; y);
23      proc2;
24    end.
```

Saída gerada

```
1   Erro na linha 1: 'identificador' inesperado [prog], esperava 'program'
2   Erro na linha 3: '=' inesperado, esperava ':='
3   Erro na linha 7: ',' inesperado, esperava ';' ou ')'
4   Erro na linha 9: tipos diferentes no comando readln.
5   Erro na linha 14: 'end' inesperado, esperava ';'
6   Erro na linha 18: Divisao de numeros nao inteiros.
7   Erro na linha 19: 'Numero_mal_formado' inesperado [3x], esperava '
        identificador' ou '(' ou 'numero_inteiro' ou 'numero_real'
8   Erro na linha 22: Parametro 1 do procedimento proc1:
9     Inteiro obtido, Real esperado.
10  Erro na linha 23: Falta de parametros na chamada do procedimento proc2. 1
        parametro
```

4.3 Exemplo 3

Entrada

```
1   program testecerto;
2   { declaracao de constantes }
3   const constante := 3;
4   const constante2 := 3.57;
5   { declaracao de variaveis }
6   var variavel : integer;
7   var variavel2, variavel4 : real;
8   { declaracao de procedimentos }
9   { procedimento sem parametros }
10  procedure procedimento;
11  var variavel3 : char;
12  begin
13    readln(variavel);
14    if variavel > 3 then
15      begin
16        variavel2 := constante2;
```

```
17          variavel := constante;
18        end
19      else
20        while variavel < 3 do
21          begin
22            variavel := variavel + (variavel + 2)/3;
23            variavel := constante2/3 +  variavel;
24          end;
25      writeln (variavel3);
26    end;
27    { procedimento com parametros }
28    procedure procedimento2 (a,b: integer; c:real);
29    var variavel3: integer;
30    begin
31      repeat
32        begin
33          variavel3 := a+b;
34          c := variavel3-1;
35        end;
36      until variavel3 > 10;
37    end;
38    { inicio do programa }
39    begin
40      procedimento;
41      procedimento2 (variavel; variavel; variavel2);
42      variavel := variavel * (variavel2 + 3*variavel) / 5;
43    end.
```

Saída

```
0   INPP
1   ALME1
2   ALME1
3   ALME1
4   DSVI 59
5   ALME1
6   LEIT
7   ARMZ 0
8   CRVL 0
9   CRCT 3
10  CPMA
11  DSVF 17
12  CRCT 3.570000
13  ARMZ 1
14  CRCT 3
15  ARMZ 0
16  DSVI 36
17  CRVL 0
18  CRCT 3
19  CPME
20  DSVF 36
21  CRVL 0
22  CRVL 0
23  CRCT 2
24  SOMA
25  CRCT 3
26  DIVI 0
27  SOMA
28  ARMZ 0
29  CRCT 3.570000
30  CRCT 3
31  DIVI 0
32  CRVL 0
33  SOMA
34  ARMZ 0
35  DSVI 17
36  CRVL 4
37  IMPR
38  DESM 1
39  RTPR
40  COPVL
41  COPVL
42  COPVL
43  ALME1
44  CRVL 4
45  CRVL 5
46  SOMA
47  ARMZ 7
48  CRVL 7
49  CRCT 1
50  SUBT
51  ARMZ 6
52  CRVL 7
53  CRCT 10
54  CPMA
55  DSVF 57
56  DSVI 44
57  DESM 4
58  RTPR
59  PUSHER 61
60  CHPR 5
61  PUSHER 66
62  PARAM 0
63  PARAM 0
64  PARAM 1
65  CHPR 40
66  CRVL 0
67  CRVL 1
68  CRCT 3
69  CRVL 0
70  MULT
71  SOMA
72  MULT
73  CRCT 5
74  DIVI 0
75  ARMZ 0
76  PARA
```