SCC0217 - Linguagens de programação e compiladoers
Prof. Diego Raphael Amancio

Trabalho 2
Analisador Sintático da LALG - Relatório

Danilo Françoso Tedeschi - 8937361
Lucas de Carvalho Rodrigues da Silva - 8624511
Matheus de França Cabrini - 8937375
Rita Raad - 8061452
Rodrigo de Andrade Santos Weigert - 8937503
Universidade de São Paulo
São Carlos

11/06/2017

1 Decisões de Projeto

1.1 Implementação

O analisador sintático foi implementado utilizando GNU Bison (extensão do Yacc). Usá-lo pareceu ser o caminho mais natural, já que o analisador léxico (parte 1 do trabalho) foi implementado utilizando flex, e esses dois programas frequentemente são utilizados juntos.

1.2 Tratamento de Erros

2 Visão Geral do Projeto até Agora

2.1 Analisador Léxico

O analisador léxico permanece como foi especificado no relatório anterior, a menos de pequenas mudanças feitas para realizar a integração entre os módulos léxico e sintático e também para adaptar apropriadamente a lógica do tratamento de erros, já que, a partir de agora, parte significativa dessa lógica residirá na (previamente ausente) análise sintática.

Por exemplo, para começar, o analisador léxico não é mais *standalone*, ou seja, não possui mais uma função *main* e não pode mais ser executado independentemente do analisador sintático.

Além disso, a maneira como os tokens são numerados foi alterada. Antes, havia um *enum* criado manualmente. Agora o mesmo é gerado automaticamente pelo Bison (com auxílio da diretiva *%token*) e exportado para o código do analisador léxico via um arquivo de cabeçalho.

Outra mudança foi na maneira como o léxico trata erros. Anteriormente, o léxico retornava um token de erro sempre que detectava um comentário não terminado. Agora, tal token não existe mais e, quando esse erro é encontrado, o léxico apenas imprime uma mensagem e prossegue com a análise, saindo do estado de leitura de comentário. Isso ocorre de modo transparente ao analisador sintático, o qual recebe o programa de entrada já com os comentários eliminados. É válido notar, porém, que o outro token de erro retornado pelo léxico - relativo à detecção de um símbolo não pertencente a gramática - permanece sendo retornado. Isso ocorre logo após a ativação de uma nova *flag* que sinaliza o sintático do erro ocorrido.

2.2 Analisador Sintático

O código do analisador sintático (*parser.y*) tem estrutura similar ao do léxico, sendo dividido em três partes. A parte 1 contém algumas declarações em C e as definições dos tokens. A parte 2 contém as definições das regras sintáticas da LALG, e a parte 3 contém a função principal do analisador sintático.

O sintático se comunica com o léxico no momento em que faz chamadas à função mais importante deste, *yylex*, responsável por ler a entrada e retornar os tokens apropriados. A realização dessas chamadas faz parte do código gerado automaticamente pelo Bison e, logo, não é de responsabilidade do usuário da ferramenta. O sintático consegue acesso à função *yylex* pois é compilado junto ao léxico e apresenta o cabeçalho da função declarado na parte 1 citada anteriormente.

3 Compilação e Execução

Um *Makefile* foi fornecido com o trabalho. Assumindo ambiente Linux com GNU Make, flex e GNU Bison instalados, para gerar os códigos dos analisadores léxico e sintático e compilar o projeto basta usar:

$$make$$

E, para executar:

$$./parser$$

ou

$$./parser{<}arquivo\_de\_entrada$$

Alternativamente, para compilação, pode-se apenas compilar conjuntamente os arquivos *lex.yy.c* e *y.tab.c* fornecidos. Nesse caso, não há necessidade de ter o flex e o Bison instalados.

Ao executar, o programa reporta todos os erros que encontrar no programa de entrada. Se nenhum erro for encontrado, o programa termina silenciosamente, como fazem certos compiladores como o GCC.

4 Exemplo de Execução

MDC com algoritmo de Euclides e alguns erros

```
1  program mdc;
2    {variaveis do programa principal}
3    var a, b: integer;
4
5    {dado inteiros a e b, imprime na tela}
6    {o maximo divisor comum entre a e b}
7    procedure calcmdc(a, b : integer);
8      var aux : integer;
9      begin
10
11       if a > b then
12       begin
13         aux := a;
14         a := b;
15         b := aux;
16       end\;
17       while (a > 0) do
18       begin
19         b := b - a;
20         if a > b then
21         begin
22           aux := a;
23           a := b;
24           b := aux;
25         end;
26       end;
27       write(b);
28     end;
29   {programa principal
30   begin
31     read(a, b);
32     calcmdc(a, b);
33   end;
34 end. {Comentario mal formado no final do arquivo
```

Saída do analisador léxico para o programa acima

```
program                         PROGRAM
mdc                             ID
;                               SEMICOLON
var                             VAR
a                               ID
,                               COMMA
b                               ID
:                               COLON
integer                         INTEGER
;                               SEMICOLON
procedure                       PROCEDURE
calcmdc                         ID
(                               LBRACKET
a                               ID
,                               COMMA
b                               ID
:                               COLON
integer                         INTEGER
)                               RBRACKET
;                               SEMICOLON
var                             VAR
aux                             ID
:                               COLON
integer                         INTEGER
;                               SEMICOLON
begin                           BEGIN
if                              IF
a                               ID
>                               GREATER
b                               ID
then                            THEN
begin                           BEGIN
aux                             ID
:=                              ATTRIB
a                               ID
;                               SEMICOLON
a                               ID
:=                              ATTRIB
b                               ID
;                               SEMICOLON
b                               ID
:=                              ATTRIB
aux                             ID
;                               SEMICOLON
end                             END
\                               16: erro: caractere invalido.
;                               SEMICOLON
while                           WHILE
```

```
(                               LBRACKET
a                               ID
>                               GREATER
0                               INT
)                               RBRACKET
do                              DO
begin                           BEGIN
b                               ID
:=                              ATTRIB
b                               ID
-                               MINUS
a                               ID
;                               SEMICOLON
if                              IF
a                               ID
>                               GREATER
b                               ID
then                            THEN
begin                           BEGIN
aux                             ID
:=                              ATTRIB
a                               ID
;                               SEMICOLON
a                               ID
:=                              ATTRIB
b                               ID
;                               SEMICOLON
b                               ID
:=                              ATTRIB
aux                             ID
;                               SEMICOLON
end                             END
;                               SEMICOLON
end                             END
;                               SEMICOLON
write                           WRITE
(                               LBRACKET
b                               ID
)                               RBRACKET
;                               SEMICOLON
end                             END
;                               SEMICOLON
                                30: erro: comentario nao fechado.
begin                           BEGIN
read                            READ
(                               LBRACKET
a                               ID
,                               COMMA
b                               ID
)                               RBRACKET
;                               SEMICOLON
calcmdc                         ID
(                               LBRACKET
a                               ID
,                               COMMA
b                               ID
)                               RBRACKET
;                               SEMICOLON
end                             END
;                               SEMICOLON
end                             END
.                               PERIOD
                                34: erro: comentario nao fechado.
```