Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SCC0217 – Linguagens de Programação e Compiladores

Trabalho 2 – Analisador Sintático para LALG

Elias Italiano Rodrigues – 7987251
Vinicius Katata Biondo – 6783972

São Carlos, 24 de maio de 2015

Sumário

1 Introdução

Este trabalho implementa um analisador sintático com tratamento de erro "modo pânico" para a linguagem de programação `LALG` utilizando as ferramentas `flex` e `bison`. Foram seguidas as instruções dadas em sala de aula assim como consultadas em tutoriais na Web [1], em manual [2] e em livro [3].

2 Como Usar

2.1 Compilação

O trabalho entregue, como requisitado, já foi previamente compilado (`Linux`), não havendo necessidade de executar esse passo. Porém, caso queira ou precise compilar novamente, basta estar dentro do diretório do trabalho e executar:

```
make
```

É necessário ter instalado o compilador `gcc`, as ferramentas `flex` e `bison`, assim como o utilitário `make` em sistema operacional `Linux`.

2.2 Execução

Para executar o trabalho, basta estar dentro de seu diretório e executar:

```
./main
```

Dessa maneira, o programa `LALG` será lido da entrada padrão `stdin`.

Para executá-lo sobre um arquivo, basta redirecionar a entrada:

```
./main < meu-programa.lalg
```

No diretório `./test/sin` encontram-se alguns exemplos **sintáticos** de programa em `LALG` para testar. Por exemplo:

```
./main < ./test/sin/programa1.lalg
```

Opcionalmente, para rodar para todos os programas `.lalg` de `./test/sin`, execute:

```
make run
```

As saídas serão escritas em arquivos com sufixo `_out` na própria pasta `./test/sin`.

2.3 Exemplo de Execução

Arquivo `./test/sin/error_varios1.lalg` – programa fictício com vários erros:

```
 1. program ; { esqueceu o nome do programa }
 2.    const c 10;  { esqueceu '=' }
 3.    var i, a, b d: integer; { esqueceu uma ',' nas variaveis }
 4.
 5.    procedure meu_proc(v: integer; x); { esqueceu tipo do segundo parametro }
 6.    var contador: string; { tipo de dado inexistente }
 7.    begin
 8.       contador := contador + * v; { usou '*' de modo errado }
 9.       write(contador);
10.       write(x);
11.    end;
12.
13.    function minha_func(v: integer, x: integer): integer; { usou ',' }
14.
15. begin
16.    read(a);
17.    read(b);
18.    a = c; { fez atribuicao com sinal de '=' }
19.    read d); { esqueceu '(' }
20.
21.    if b < then { esqueceu valor na condicao }
22.       write(b);
23.
24.    for i := 10 to 20 { esqueceu 'do' }
25.    begin
26.       minha_func(a, d);
27.    end;
28. end { esqueceu '.' }
```

Comando:

```
./main < ./test/sin/error_varios1.lalg
```

Saída:

```
[ 1,9 ]: syntax error, unexpected ;, expecting identificador
[ 2,10]: syntax error, unexpected valor inteiro, expecting =
[ 3,14]: syntax error, unexpected identificador, expecting :
[ 5,34]: syntax error, unexpected ), expecting :
[ 6,16]: syntax error, unexpected identificador, expecting char or
         integer or real
[ 8,26]: syntax error, unexpected *, expecting ( or identificador or
         valor inteiro or valor real
[13,32]: syntax error, unexpected ",", expecting )
[18,4 ]: syntax error, unexpected =, expecting else or ; or ( or :=
[18,6 ]: syntax error, unexpected identificador
[19,7 ]: syntax error, unexpected identificador, expecting (
```

```
[21,9 ]: syntax error, unexpected then, expecting ( or identificador or
         valor inteiro or valor real
[25,2 ]: syntax error, unexpected begin, expecting do
[26,15]: syntax error, unexpected ",", expecting ; or )
[28,5 ]: syntax error, unexpected $end, expecting .
```

Onde `[i,j]` indica linha `i` na coluna `j`.

Mais exemplos estão disponíves no diretório `./test/sin`.

3 Organização dos Arquivos

O diretório do trabalho está organizado da seguinte maneira:

`./doc` : diretório dos arquivos LaTeX fonte deste relatório.

`./test` : diretório com exemplos de programa `LALG` para testes.
`|-- ./lex` : exemplos léxicos.
`|-- ./sin` : exemplos sintáticos.

`LALG` : definição da linguagem `LALG`.

`Makefile` : arquivo para automizar compilação e execução usando o utilitário `make`.

`RELATORIO.pdf` : este relatório PDF compilado a partir de `./doc`.

`README` : arquivo com instruções.

`lalg.l` : programa `Lex` para a linguagem `LALG`.

`lalg.y` : programa em `Bison` para a linguagem `LALG`.

`lex.yy.c` : programa `C` gerado pelo `flex`.

`y.tab.c` : programa principal `C` gerado pelo `bison`.

`y.tab.h` : cabeçalho `C` gerado pelo `bison`.

`main` : o programa principal a ser executado para fazer a análise sintática.

4 Decisões de Projeto

4.1 Mudanças em relação ao trabalho anterior

As seguintes mudanças foram feitas em relação ao trabalho anterior:

- Para facilitar o processo de compilação e linkagem, os diretórios `./bin` e `./src` foram removidos e agora os arquivos `.y`, `.l`, `.h` e `.c` encontram-se todos no diretório raiz do trabalho.

- Optou-se por mudar a implementação do léxico de `<palavra_reservada, palavra_reservada>` para `<palavra_reservada, simbolo_palavra_reservada>` devido a maior facilidade ao trabalhar no `Bison`. Confira a Tabela 1.
- A busca para conferir se um identificador é uma palavra reservada agora é feita no `lalg.l`.
- Os símbolos dos *tokens* foram movidos do arquivo `lalg.l` para o arquivo `lalg.y`.
- Removeu-se o "pseudo-token" `RESERVED` criado somente para auxiliar na implementação.

Palavra Reservada	**Símbolo**
begin	W_BEGIN
char	W_CHAR
const	W_CONST
do	W_DO
else	W_ELSE
end	W_END
for	W_FOR
function	W_FUNCTION
if	W_IF
integer	W_INTEGER
procedure	W_PROCEDURE
program	W_PROGRAM
read	W_READ
real	W_REAL
repeat	W_REPEAT
when	W_THEN
to	W_TO
until	W_UNTIL
var	W_VAR
while	W_WHILE
write	W_WRITE

Tabela 1: Listagem dos símbolos adotados para as palavras reservadas.

4.2 `lalg.y`

O arquivo `lalg.y` contém a função principal `main()` em que é chamada a execução do analisador sintático e nele está descrita a gramática da linguagem no formato `Bison`.

Juntamente com o gramática em `lalg.y`, está também implementado o **tratamento de erro sintático modo pânico**. Sua implementação consistiu de uma reescrita na gramática em que foram acrescentadas regras para tratar os erros usando o símbolo especial `error` e informando uma lista de *tokens* de sincronização composta por: seguidores dos *tokens* esperados mais seguidores do pai e adicionais no contexto de cada regra. Confira o arquivo `lalg.y` para detalhes. Nas ações dessas regras, foi utilizado a macro `yyerrok` que instrui o `Bison` a continuar a análise mesmo diante do erro encontrado.

Além disso, fez-se uso da diretiva `%error-verbose`, para deixar o `Bison` emitir as mensagens de erro invocando a função `yyerror()`, e da diretiva `%locations` para que o `Bison` acione o recurso de localização – o que possibilita usar a informação de linha e coluna vinda do analisador léxico por meio da variável `yylloc`.

O único conflito do tipo *shift/reduce* da gramática, causado pela produção dos comandos `if/else`, foi deixado a cargo do `Bison` resolver e usou-se a diretiva `%expext 1` para não exíbi-lo durante a compilação. Na implementação da parte semântica, esse conflito será revisto.

4.3 `lalg.l`

Foram inseridas ações para capturar os valores de alguns *tokens* que posteriormente serão usados na semântica/sintática do próximo trabalho.

5 Observações

Os seguintes problemas foram encontrados na gramática do LALG durante a implementação deste trabalho:

- Uma função não possui corpo;
- O valor retornado de uma função não pode ser usado para ser atribuído a uma variável.

Essas faltas na gramática influenciarão na análise semântica.

6 Conclusão

O trabalho desenvolvido cumpre a especificação dada. Foi possível aprender mais sobre a ferramenta `bison` e concluir o analisador sintático de `LALG` que usou o analisador léxico implementado no trabalho anterior.

Referências

[1] Part 01: Tutorial on lex/yacc
<https://www.youtube.com/watch?v=54bo1qaHAfk>
Acesso em: 6 de maio de 2015

Part 02: Tutorial on lex/yacc.
<https://www.youtube.com/watch?v=__-wUHG2rfM>
Acesso em: 6 de maio de 2015

[2] Bison 3.0.4
<http://www.gnu.org/software/bison/manual/html_node/index.html>
Acesso em: 6 de maio de 2015

[3] LEVINE, John. flex & bison. United States of America: O'Reilly, 2009.