Relatório
Construção de Compiladores

Gerador de analisador sintático – Bison

Victor Hugo Cândido de Oliveira
RA 121041451

Objetivo

Desenvolvimento de um gerador de analisador sintático utilizando a ferramenta GNU Bison em conjunto com a ferramenta de análise léxica Flex.

Linguagem

A linguagem L aceita pelo analisador sintático desenvolvido será especificada pelas regras a seguir.

As regras foram tiradas do arquivo parser.output gerado pelo Bison.

Os tokens reconhecidos são:

MAIN – token da função principal

IF – token “if”

ELSE – token “else”

DO – token “do”

WHILE – token “while”

FOR – token “for”

BREAK – token “break”

PRINT – token “print”

RETURN – token “return”

DTYPE – tokens “if” e “float”

STRING – strings utilizadas no comando “print”

NUMBER – inteiros e reais

ID – token para identificadores e nome de funções

UNLOP – operador lógico unário (Unary Logical OPerator)

BINLOP – operador lógico binário (BINary Logical OPerator)

MATHOP – operador matemático (MATHematical OPerator)

ATTR – operador de atribuição

Código gerado pelo Bison (parser.output)

```
Grammar
    0 $accept: program $end

    1 program: MAIN '(' ')' '{' commands '}' program_end
    2        | error

    3 program_end: %empty
    4            | program_end1

    5 program_end1: ID '(' ')' '{' commands '}' program_end1
```

```
 6             | error

 7 commands: %empty
 8          | commands1

 9 commands1: '{' commands '}'
10           | commands1 command
11           | command
12           | ';'

13 command: cmd ';'
14         | DTYPE declarations ';'
15         | if_stmt
16         | for_stmt
17         | while_stmt
18         | dowhile_stmt

19 cmd: PRINT STRING
20     | BREAK
21     | RETURN
22     | attribution
23     | error

24 if_stmt: IF '(' fsecond ')' '{' commands '}' else_stmt

25 else_stmt: %empty
26           | ELSE '{' commands '}'

27 for_stmt: FOR '(' ffirst ';' fsecond ';' fthird ')' '{' commands '}'

28 ffirst: %empty
29        | declarations

30 fsecond: %empty
31         | binlop

32 fthird: %empty
33        | fthird_body

34 fthird_body: cmd ',' fthird_body
35             | cmd

36 while_stmt: WHILE '(' fsecond ')' '{' commands '}'
```

```
37 dowhile_stmt: DO '{' commands '}' WHILE '(' fsecond ')' ';'

38 declarations: declarations ',' decl
39              | decl

40 decl: ID
41     | attribution
42     | error

43 attribution: ID '=' expr

44 expr: binlop
45     | binmop
46     | '(' expr ')'
47     | UNLOP expr
48     | ID
49     | NUMBER
50     | error

51 binlop: expr BINLOP expr

52 binmop: expr MATHOP expr
```

Conteúdo

Seguem os seguintes arquivos:

- scanner.l – código do gerador de analisador léxico (Flex)
- parser.y – código do gerador de analisador sintático (Bison)
- symboltable.h/symboltable.c – tabela de símbolos
- compila.sh – código em shell script para compilar o scanner.l, o parser.y e gerar o executável exec com os códigos lex.yy.c, parser.tab.c e symboltable.c
- input_scanner – exemplo de código aceito pela linguagem

Para compilar e executar

Execute o script compila.sh com o comando

./compila.sh

Execute o executável exec gerado pelo script e insira o código pelo próprio stdin

./exec

ou execute direcionando o conteúdo de um arquivo para a entrada do programa (ex: input_scanner)

./exec < input_scanner

Saída do analisador

O analisador acusará erros de sintaxe como falta de parênteses, chaves, ponto e vírgula, declarações incompletas, etc.

Caso não exista erros sintáticos a análise será concluída com sucesso.