Compiladores – 2016/2
Roteiro de Laboratório 7

1 Objetivos

O objetivo deste laboratório é implementar um *parser* para a linguagem TINY usando o `bison`.

2 Sintaxe da linguagem TINY

A sintaxe da linguagem está apresentada em notação BNF abaixo, aonde os terminais (*tokens*) estão escritos em fonte `mono-espaçada` e os não-terminais em *itálico*.

program → *stmt-sequence*
stmt-sequence → *stmt-sequence stmt* | *stmt*
stmt → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → `if` *expr* `then` *stmt-sequence* `end`
| `if` *expr* `then` *stmt-sequence* `else` *stmt-sequence* `end`
repeat-stmt → `repeat` *stmt-sequence* `until` *expr*
assign-stmt → `identifier :=` *expr* `;`
read-stmt → `read identifier ;`
write-stmt → `write` *expr* `;`
expr → *expr bin-op expr* | `(` *expr* `)` | `number` | `identifier`
bin-op → `<` | `=` | `+` | `-` | `*` | `/`

A gramática acima é equivalente à apresentada no roteiro de Laboratório 05, em notação EBNF. No entanto, a gramática acima possui ambiguidades que levam a conflitos de *shift-reduce*. Utilize os comandos do `bison` para definição de prioridades de operadores (`%left`, etc) para remover as ambiguidades da gramática.

3 Implementado um *Parser* para a linguagem TINY

As convenções léxicas da linguagem TINY já foram apresentadas no roteiro do Laboratório 2. Você pode usar o mesmo *scanner* do Laboratório 5 para realizar a atividade deste roteiro.

Utilize as demais opções do `bison` como demonstrado pelo professor.

4 Entrada e Saída do *Parser*

O seu programa de entrada é lido da entrada padrão (*stdin*), como abaixo:

```
$ ./parser < program.tny
```

Se o programa estiver correto, o seu *parser* deve exibir uma mensagem indicando que o programa foi aceito, como abaixo:

```
$ ./parser < program.tny
PARSE SUCESSFUL!
```

Se o programa possuir erros léxicos, exiba uma mensagem informativa como abaixo:

```
$ ./parser < program.tny
SCANNING ERROR (XX): Unknown symbol SS
```

Aonde `XX` é a linha aonde o símbolo desconhecido `SS` apareceu.

Se o programa possuir erros sintáticos, exiba uma mensagem informativa como abaixo:

```
$ ./parser < program.tny
PARSE ERROR (XX): syntax error, unexpected UT, expecting ET
```

Aonde `UT` e `ET` são os tipos de *tokens* lido e esperado, respectivamente. Utilizando as opções `%define parse.error verbose` e `%define parse.lac full`, a mensagem depois do `:` já é gerada automaticamente pelo `bison`. Neste caso, basta definir a função de erro como abaixo:

```
// Error handling.
void yyerror (char const *s) {
    printf("PARSE ERROR (%d): %s\n", yylineno, s);
}
```

Observações importantes:

- O seu *parser* pode terminar a execução ao encontrar o primeiro erro no programa de entrada.
- Veja mais exemplos de entrada e saída no AVA.
- Uma implementação de referência para esse laboratório será disponibilizado pelo professor em um futuro próximo. No entanto, você é *fortemente* encorajado a realizar a sua implementação completa antes de ver uma solução em outro lugar.