Compiladores – 2016/2
Roteiro de Laboratório 5

1 Objetivos

O objetivo deste laboratório é implementar um *parser* de descida recursiva para TINY.

2 Sintaxe da linguagem TINY

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

program → *stmt-sequence*
stmt-sequence → *stmt* { *stmt* }
stmt → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → `if` *exp* `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 → *simple-expr* [*comparison-op simple-expr*]
comparison-op → `<` | `=`
simple-expr → *term* { *add-op term* }
add-op → `+` | `-`
term → *factor* { *mul-op factor* }
mul-op → `*` | `/`
factor → `(` *expr* `)` | `number` | `identifier`

3 Implementado um *Parser* para a linguagem TINY

As convenções léxicas da linguagem TINY já foram apresentadas no roteiro do Laboratório 2. Você deve adaptar o *scanner* desenvolvido anteriormente para compor o *parser* desse laboratório.

Você deve desenvolver um *parser* de descida recursiva que reconhece corretamente todos os programas TINY com sintaxe válida. Além disso, para programas incorretos, o seu *parser* deve indicar os erros léxicos e sintáticos presentes no código. (Veja próxima seção.)

Crie uma função em C para cada não-terminal da gramática. Utilize como ponto de partida os exemplos disponibilizados no AVA e os códigos apresentados na aula teórica (veja slides da Aula 04).

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): unexpected token type UT, expected ET
```

Aonde `UT` e `ET` são os tipos de *tokens* lido e esperado, respectivamente.

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.