Analisador léxico TIGER

Primeiro trabalho prático de Compiladores I

Ábner de Marcos Neves

2018054605

1. Introdução

Um compilador é seccionado em várias fases: as análises léxica, sintática e semântica; a geração dos códigos intermediário e objeto (otimizado ou não); e, por fim, a geração do próprio programa objeto. O objetivo deste trabalho é implementar um analisador léxico para a linguagem TIGER[1], isto é, a primeira fase de um compilador que será incrementalmente construído.

A principal tarefa da análise léxica é ler os caracteres de um programa-fonte e os agrupar em sequências significativas, chamadas *lexemas*. Para cada lexema, então, o analisador léxico gera um *token*, que consiste num par <nome-token, valor-atributo>.

2. Implementação e ferramentas de apoio

O analisador léxico foi implementado com o auxílio do gerador de analisadores léxicos LEX. Essa ferramenta permite que, de maneira simples, sejam descritos os padrões de cada *token* presente na linguagem-fonte e, então, um compilador LEX gera um analisador léxico em outra linguagem (nesse trabalho, C/C++). A execução desse analisador léxico gerado toma como entrada um programa-fonte em TIGER e tenta casar, um a um, os lexemas encontrados com os padrões de *tokens* descritos.

Além dessa tarefa primária, o analisador léxico também trata comentários ignorando o que está entre `/*` e `*/` e não gera *tokens* com esse conteúdo, mesmo que eventualmente alguma subcadeia de um comentário case com um padrão léxico. Tratamento parecido ocorre com as strings, mas essas geram um *token* de *string*.

A saída dessa fase do compilador é a sequência de *tokens* gerados a partir do código-fonte, que tomam a forma:

```
nome-token -> valor-atributo
```

3. Execução

Para gerar o analisador léxico, foi feito um Makefile, logo, basta abrir um terminal Linux no diretório com os arquivos e executar o comando:

```
make
```

Assim, será gerado um arquivo `scanner.out` que pode ser executado com o seguinte comando:

```
./scanner.out < arquivo-de-entrada > arquivo-de-saída
```

O redirecionamento de entrada e saída padrão é opcional.

Caso deseje apagar os arquivos gerados, digite:

```
make clean
```

4. Trabalho futuro

Essa é apenas a primeira fase de um compilador. A saída do analisador léxico deve, nas próximas etapas, alimentar um analisador sintático e continuar o processo de compilação, como descrito brevemente na introdução. À medida em que essas fases forem implementadas, elas deverão substituir as respectivas partes de um compilador TIGER já existente até que, ao final, ele seja substituído por completo.

Nas próximas etapas, os testes do compilador mostrar-se-ão provavelmente bem mais complicados, uma vez que exigirão corretude sintática e semântica nos programas-fonte que, até agora, não se fazia necessária.

5. Conclusão

A construção de um analisador léxico se torna muito menos laboriosa com o auxílio de ferramentas como o LEX. Apesar de uma confusão inicial com a estrutura padrão de um programa desse tipo e a dificuldade de entender certas dependências externas e funções, foi possível compreender mais a fundo como se dá essa análise.

A descrição de padrões em expressões regulares foi facilitada pela simplicidade da linguagem TIGER e conjunto limitado de símbolos, mas ainda se mostrou desafiadora quando se tratava de *strings* e comentários, por exemplo.

6. Referências

[1] BIGONHA, Mariza A. S. **Um compilador para a linguagem TIGER usando a linguagem JAVA como ferramenta de desenvolvimento.** 10 set. 2021. Disponível em: https://homepages.dcc.ufmg.br/~mariza/Cursos/CompiladoresI/Geral/Projeto2021-2/EspComp2021-2/trabComp2021-2.pdf. Acesso em: 4 nov. 2021.

[2] BERK, Elliot Joel; ANANIAN, C. Scott. **JLex: A lexical analyzer generator for Java**. 7 fev. 2003. Disponível em: https://www.cs.princeton.edu/~appel/modern/java/JLex/. Acesso em: 4 nov. 2021.

[3] **The Flex manual page**. Disponível em: http://dinosaur.compilertools.net/flex/manpage.html. Acesso em: 4 nov. 2021.