

FACULDADE: CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB

CURSO: CIÊNCIA DA COMPUTAÇÃO

DISCIPLINA: CONSTRUÇÃO DE COMPILADORES

CARGA HORÁRIA: 75 H. A.

ANO/SEMESTRE: 2017/01

PROFESSOR: EDUARDO FERREIRA DOS SANTOS

HORÁRIOS: Quartas às 09h40 e Sextas às 07h40

LABORATÓRIO 02 – ANALISADOR LÉXICO E SINTÁTICO

RESUMO

A construção de compiladores é uma disciplina da computação que tem como objetivo transformar linguagem de programação de alto nível em linguagem de máquina. O ANTLR é uma ferramenta de *parsing* construída em Java que pode ser utilizada para ler, processar, executar ou traduzir texto estruturado em arquivos binários. A utilização da ferramenta ANTLR vai facilitar a implementação de uma gramática de forma a construir um compilador.

OBJETIVOS

Objetivo Geral

Utilizar a ferramenta ANTLR para realizar a análise sintática.

Objetivos Específicos

1. Construir uma gramática eliminando a ambiguidade;
2. Integrar o ANTLR a um programa Java;
3. Realizar a etapa de análise sintática.

EXERCÍCIO 01 – INICIALIZAÇÃO DO PROJETO

Para iniciar o laboratório baixe todos os materiais necessários diretamente da Internet no seguinte endereço: <https://owncloud.eduardosan.com/index.php/s/mqB7S4YzGEQ4cbb>

A infra-estrutura fornecida contém os seguintes diretórios:

```
.
|-- bin
|-- build.xml
|-- lib
|   |
|   |-- antlr.jar
|-- src
|   |
|   |-- decaf
|       |-- DecafLexer.g4
```

EXERCÍCIO 01 – INICIALIZAÇÃO DO PROJETO

```
| |-- DecafParser.g4
| |-- Main.java
|-- java6035
|   |-- tools
|       |-- CLI
|           |-- CLI.java
```

Os arquivos importantes para o projeto estão no diretório `provided/skeleton/src/decaf/DecafParser.g4` e `provided/skeleton/src/decaf/DecafLexer.g4`. Cada um deles contém uma gramática para fazer especificamente um analisador léxico e o parser para um conjunto de caracteres definido na gramática.

Para compilar o projeto vamos utilizar a ferramenta `ant`¹ para compilar projetos em Java. Para iniciar o trabalho com a ferramenta, realize os seguintes passos:

1. Crie um repositório público para o projeto ou adiciona a um já existente;
2. Importe o código para dentro do Projeto.

Agora que o projeto já existe e está versionado, é possível compilar os elementos. Para fazê-lo, utilize a seguinte sequência de comandos:

```
# Baixe o repositório git
cd /home/aluno
git clone <url_do_repositorio>

# Copie o código para dentro do repositório recém criado
cp -rp skeleton/* /home/aluno/<diretorio_do_projeto>

# Adicione os arquivos ao git
cd /home/aluno/<diretorio_do_projeto>
git add .
git commit -m "Import do projeto"
git push origin master

# Agora compile o projeto
ant

# Execute o scanner
java -jar dist/Compiler.jar -target scan -debug ../scanner/char1
```

Verifique a saída para ter certeza que o programa foi compilado com sucesso.

1 Mais informações em <http://ant.apache.org/>

EXERCÍCIO 02 – EXECUTANDO SCANNER

A gramática fornecida fornece as ferramentas necessárias para realizar a identificação dos tokens primários, mas ainda não é suficiente para a análise léxica e sintática. O scanner deve ser capaz de identificar os tokens da linguagem DECAF, descrita no seguinte documento: <http://www.eduardosan.com/wp-content/uploads/2016/07/decaf-manual-description.pdf>

Altere o analisador léxico fornecido para obter as seguintes restrições:

1. O scanner deve ser capaz de reconhecer caracteres ilegais;
2. Identificar aspas simples e duplas faltando;
3. Apresentar mensagens de erro descritivas em caso de falhas na análise léxica;
4. Ainda que encontre erros, deve continuar analisando o arquivo.

Para completar as restrições, altere o arquivo `provided/skeleton/src/decaf/DecafLexer.g4` para adicionar as regras que estão faltando na gramática. Ao terminar a alteração, execute novamente o comando do `ant` e recompile a gramática para validar o resultado.

```
# Agora compile o projeto  
ant
```

```
# Execute o scanner  
java -jar dist/Compiler.jar -target scan -debug ../scanner/char1
```

Para efeito de comparação, no início seu programa deve produzir a seguinte saída:

```
2 'a'  
2 'b'  
2 'c'  
3 'R'  
3 'i'  
3 'n'  
3 'a'  
3 'r'  
3 'd'  
4 '6'  
4 '0'  
4 '3'  
4 '5'
```

Ao final das alterações na gramática, o programa deve produzir a seguinte saída:

```
2 CHARLITERAL 'a'  
2 CHARLITERAL 'b'  
2 CHARLITERAL 'c'  
3 CHARLITERAL 'R'  
3 CHARLITERAL 'i'  
3 CHARLITERAL 'n'  
3 CHARLITERAL 'a'  
3 CHARLITERAL 'r'
```

EXERCÍCIO 02 – EXECUTANDO SCANNER

3 CHARLITERAL 'd'
4 CHARLITERAL '6'
4 CHARLITERAL '0'
4 CHARLITERAL '3'
4 CHARLITERAL '5'

BIBLIOGRAFIA

PARR, Terence. **The definitive ANTLR 4 reference**. Pragmatic Bookshelf, 2013. Disponível em <http://www4.di.uminho.pt/~gepl/GQE/documents/books/Pragmatic.The.Definitive.ANTLR.4.Reference.Jan.2013.pdf>

AHO, Alfred V. E Outros. **Compiladores: princípios, técnicas e ferramentas**. PEARSON, 2007.

PRICE, Ana Maria de Alencar. **Implementação de linguagens de programação: compiladores**. SAGRA-LUZZATTO, 2005.

Material inspirado no disponível em <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-035-computer-language-engineering-spring-2010/>