Projeto de Implementação de um Compilador para a Linguagem **TPP**: Análise Léxica (Trabalho – 1ª parte)

Prof. Rogério Aparecido Gonçalves *Universidade Tecnológica Federal do Paraná (UTFPR)*

Este documento apresenta a especificação da 1ª parte do trabalho de implementação da disciplina. O objetivo nessa etapa é projetar e implementar a a fase de *Análise Léxica* do compilador para a linguagem **TPP**.

Contents

1 Análise Léxica

A **Análise Léxica** é a fase do compilador que lê o código-fonte do arquivo de entrada como um fluxo de caracteres, e nesse processo de varredura reconhece os *tokens* ou marcas da linguagem. As denominações Sistema de Varredura, Analisador Léxico e *Scanner* são equivalentes.

Devem ser reconhecidas as marcas presentes na linguagem `TPP`, como `se`, `repita` e outras que são palavras chave, palavras reservadas. Precisam ser reconhecidos os nomes de variáveis e funções que são os identificadores, símbolos e operadores aritméticos, lógicos e relacionais.

O processo de reconhecimento das marcas, a identificação de padrões pode ser feito de duas formas: utilizando-se *expressões regulares* ou implementando o analisador com *autômatos finitos*.

Para implementar o sistema de varredura (scanner) para a linguagem `TPP`, é necessário tomar nota das classes de **tokens** apresentadas na Tabela 1.

Table 1: **Tokens** da linguagem **TPP**

palavras reservadas	símbolos
`se`	+ soma
`então`	- subtração
`senão`	* multiplicação
`fim`	/ divisão
`repita`	= igualdade
`flutuante`	, vírgula
`retorna`	:= atribuição
`até`	< menor
`leia`	> maior
`escreva`	<= menor-igual
`inteiro`	>= maior-igual
	(abre-par
	(fecha-par
	: dois-pontos

palavras reservadas	símbolos
	[abre-col
	] fecha-col
	&& e-logico
	|| ou-logico
	! negação

Ainda podem ser definidos os **tokens**:

- **número**: 1 ou mais dígitos que podem ser *inteiro* ou *flutuante* (representação em notação científica ou não);
- **identificador**: começa com uma letra e precede com *N* letras e números sem limite de tamanho;
- **comentários**: cercados de chaves da seguinte forma: `{...}`. É possível inclusive comentários que cubram múltiplas linhas.

Os **tokens** estão definidos na Gramática da Linguagem: SE, ENTAO, SENAO, REPITA, ATE, FIM, ADICAO, SUBTRACAO, MULTIPLICACAO, DIVISAO, ...

Obs.: Para a construção da linguagem, utilize palavras reservadas em português brasileiro como indicado na tabela.

1.1 Instruções Gerais

1. Faça download do arquivo do modelo de estrutura do trabalho e relatório disponível na página da disciplina no moodle. Descompacte e trabalhe nos arquivos e estrutura fornecida, pois será a mesma estrutura que deverá ser entregue ao final do projeto.
2. Siga a estrutura fornecida para desenvolver o trabalho.
3. O relatório deve ter a descrição do trabalho e da implementação e exemplos de entrada e saída gerada pelo Analisador Léxico.
4. Deverão ser entregues:
   a) O código fonte da implementação.
   b) Relatório em **pdf** que pode ser feito no formato do `LibreOffice` ou no `Latex`.
5. O projeto deve seguir a estrutura de diretórios e arquivos, disponível no formato. A estrutura do projeto é apresentada na Figura 1.

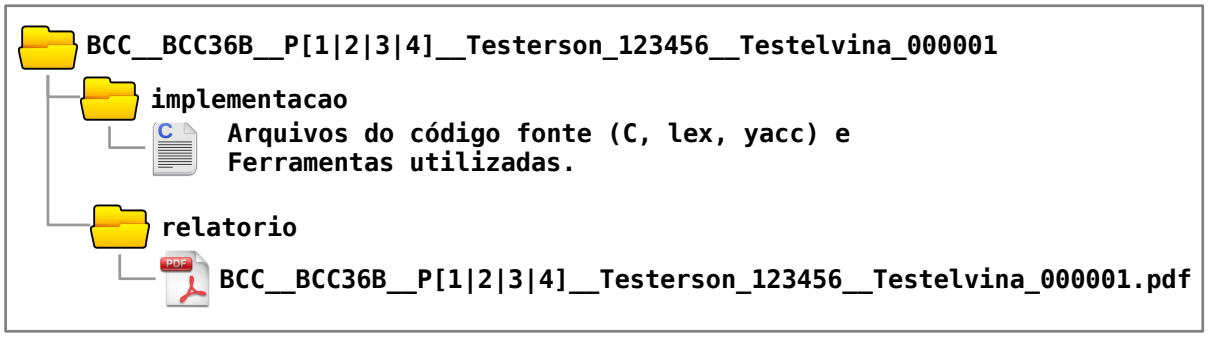

Figure 1: Formato de Entrega

1.2 Linguagens de programação para a implementação

Para a implementação do compilador, pode ser utilizado qualquer linguagem de programação[1]. É recomendado que seja utilizado uma linguagem que dê suporte à estruturas de dados de alto nível e preferencialmente que exista bibliotecas para a construção da varredura e gramática. Algumas LPs/ferramentas conhecidas são:

- `C/C++ - Flex/Bison` (Paxson et al. 1995), (GNU 2021)
- `Python - PLY` (que possui ferramenta Léxica e Sintática) (Beazley 2020)
- `Java - JFlex/Jacc` (Gerwin Klein and Décampsy 2020), (Jones 2016)

Obs.: A implementação de referência será apresentada na linguagem `C`.

1.3 Testes

Alguns casos de testes estão disponíveis no moodle institucional junto com essa especificação. Serão executados esses testes e outros testes que o professor julgar necessário durante a avaliação desta parte do trabalho.

1.4 Documentação

Durante toda a disciplina o aluno criará uma documentação formal da implementação do compilador para a linguagem. Sendo o relatório com conteúdo acumulativo, isto é, as fases subsequentes irão complementar o conteúdo existente das fases anteriores. Para a fase de Análise Léxica, a documentação deve apresentar:

- Especificação da linguagem de programação `TPP`;
- Especificação formal dos autômatos para a formação de cada classe de **token** da linguagem;
- Detalhes da implementação da varredura na LP e ferramenta (e/ou bibliotecas) escolhidas pelo projetista;
- Exemplos de saída do sistema de varredura (lista de *tokens*) para exemplos de entrada (código fonte).
- Implemente uma função que imprima a lista de **tokens**, não utilize a saída padrão da ferramenta de implementação de **Analisadores Léxicos**.

Utilize o formato de artigo da SBC[2] para fazer o relatório.

1.5 Avaliação

Será avaliado o funcionamento da varredura para a linguagem de programação `TPP`.

- Varredura: programa de exemplo `TPP` de **entrada** na linha de comando.
- **Saída** será o conjunto de pares (valor:token).

Para a avaliação inicial será considerado então (obrigatoriamente):

- Utilizar palavras reservadas em português (pt-BR);
- Construção da Análise Léxica;
- Inserção de comentários (para adicionar explicações futuras no código);
- Levantamento de erros (sugerindo classes de erros). Para isso: Contabilizar linhas (`\n`), colunas e lexema atual (as ferramentas fazem isso);
- Serão avaliados, dentre outros critérios:

[1] Que tenha suporte às ferramentas de especificação do sistema de varredura e da gramática e que também tenha suporte ao LLVM.
[2] Formato para publicação de artigos da SBC: http://www.sbc.org.br/documentos-da-sbc/summary/169-templates-para-artigos-e-capitulos-de-livros

a) Da implementação: + O funcionamento do programa. + O capricho e a organização na elaboração do projeto. + A corretude da implementação em relação ao que foi pedido no trabalho. + A colocação em prática dos conceitos que foram discutidos em sala de aula de forma correta. + A qualidade do projeto e da implementação (descrição e elaboração do projeto e o passo a passo da implementação).

b) Do relatório: + O conteúdo e a forma que foi apresentado, se o formato é o mesmo solicitado. + Organização das ideias e do processo de tradução. + O capricho na elaboração e na formatação do texto, bem como o conteúdo do texto.

- Não serão avaliados os trabalhos:

a) Que chegarem fora do prazo.
b) Que não forem feitos nas ferramentas solicitadas.
c) Que não estão no formato especificado.
d) Que não foram compactados em um só arquivo.
e) Que não tiverem identificação (nome e matrícula).
f) Que forem cópias de outros trabalhos ou materiais da internet.
g) Que não seguirem todas estas instruções.

- Não se esqueça que o trabalho vale **10,0** e contribui para o cálculo da nota final.

1.6 Entrega e Apresentação

O trabalho será **individual** e deverá ser entregue até o dia **30/08/2022** no moodle da disciplina em um pacote compactado. A estrutura do projeto com os arquivos do projeto (fonte e relatório) deve ser compactada (zipados) e o arquivo compactado deve ser enviado pelo moodle utilizando a opção de submissão **"Trabalho 1ª parte - Análise Léxica"**, o nome do arquivo compactado deve seguir o padrão de nomes do formato.

Deverá ser especificado na entrega o mecanismo de execução da varredura para a realização da correção.

Obs.: Favor utilizar ZIP como forma de compactação. O Relatório deve ser entregue impresso, no horário da aula, para o professor.

Referências

AHO, Alfred V., Monica S. LAM, Ravi SETHI, and Jeffrey D. ULLMAN. 2008. *Compiladores: Princípios, Técnicas e Ferramentas*. 2nd ed. São Paulo, SP: Pearson Addison-Wesley.

Beazley, David M. 2020. "PLY (Python Lex-Yacc)." https://www.dabeaz.com/ply/ply.html.

Gerwin Klein, Steve Rowe, and Régis Décampsy. 2020. "JFlex User's Manual, The Fast Lexical Analyser Generator)." https://jflex.de/manual.html.

GNU. 2021. "GNU Bison - The Yacc-compatible Parser Generator." https://www.gnu.org/software/bison/manual/.

JARGAS, Aurélio Marinho. 2012. *Expressões Regulares: Uma Abordagem Divertida*. 4th ed. São Paulo, SP: Novatec.

Johnson, Stephen C. 1975. "Yacc: Yet Another Compiler Compiler." *Bell Laboratories*.

Jones, Mark P. 2016. "jacc: just another compiler compiler for Java." http://web.cecs.pdx.edu/~mpj/jacc/.

Lesk, Michael E., and Eric Schmidt. 1975. "Lex: A Lexical Analyzer Generator." http://epaperpress.com/lexandyacc/download/lex.pdf.

LOUDEN, Kenneth C. 2004. *Compiladores: Princípios e Práticas*. 1st ed. São Paulo, SP: Thomson.

Paxson, Vern, and others. 1995. "Flex–Fast Lexical Analyzer Generator." *Lawrence Berkeley Laboratory*.