
**UNIVERSIDADE FEDERAL DA FRONTEIRA SUL**
**CAMPUS DE CHAPECÓ**
**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

**IGOR RONSONI**

# CIÊNCIA DA COMPUTAÇÃO
# TRABALHO PRÁTICO DE CONSTRUÇÃO DE COMPILADORES

**CHAPECÓ**
**2021**

Universidade Federal da Fronteira Sul

Igor Ronsoni

# Construção de um compilador

Chapecó
2021

# Compilador

Igor Ronsoni

Universidade Federal da Fronteira Sul, Rodovia SC 484 - Curso de Ciência da Computação

igor.ronsoni@estudante.uffs.edu.br

**Resumo**:

O presente artigo tem como objetivo, detalhar o processo de implementação do trabalho prático da disciplina de Construção de Compiladores. O trabalho consiste em implementar um compilador completo. O compilador apresentado não foi implementado por completo, apenas as etapas léxica e sintática foram criadas.

## 1. Introdução

Como continuação do conteúdo da matéria de Linguagens Formais e Autômatos, na disciplina de Construção de Compiladores, nosso projeto final será de implementar um compilador, onde o código fonte deverá passar por 5 etapas: Análise Léxica, Análise Sintática, Análise Semântica, Geração de Código intermediário e Otimização de Código. Serão apresentadas no artigo apenas as 2 primeiras etapas que foram implementadas.

## 2. Implementação

A aplicação foi desenvolvida em Python pois é uma linguagem que possui uma maior facilidade para se manusear strings vindas de arquivos.

O presente trabalho prático teve como principal ferramenta o trabalho desenvolvido na disciplina de Linguagens Formais e Autômatos que consistia em desenvolver a leitura de gramática e tokens de um arquivo e criar uma tabela de transições. A tabela gerada foi usada para o reconhecimento dos tokens de entrada na etapa léxica do compilador.

A etapa léxica teve como objetivo identificar os tokens e criar uma pilha contendo o símbolo terminal de determinado token, a linha e coluna que o token se encontra e por último o token reconhecido.

A análise sintática teve como objetivo reconhecer a erros sintáticos como ordem dos tokens em uma linha, estrutura no geral.

## 3. Estrutura

A etapa lexica tem como estrutura basica um dicionario (dict(Python)) onde cada estado não terminal que dá nome a gramática é reconhecido como key e suas transições como uma value, dentro deste value temos os símbolos terminais como keys e os estados de transição como values.

Na etapa sintática a fita de saída da léxica é vista como uma pilha e a gramática gerada pelo GoldParser é vista como uma estrutura de dicionário de dicionário.

## 4. Análise Léxica

Após a geração da tabela do autômato finito livre de estados mortos e inalcançáveis com estados de erro, o analisador léxico realiza a leitura de um arquivo contendo o código fonte digitado pelo programador e analisa cada caracter do arquivo fonte com o as transições da tabela do autômato determinístico. Quando encontrado uma transição para o estado de erro, é analisado se o erro ocasionado é gerado pela falta de um espaço em branco entre um operador e um token qualquer ou um delimitador e um token qualquer, então é feito o tratamento de tal tirando um caracter ou levantando uma exceção dizendo qual token gerou o erro na linha e na coluna especificada. Cada token reconhecido é salvo pelo estado final que veio do autômato finito em pilha junto com a linha e a coluna do código fonte onde o token foi encontrado e também é salvado o token. O reconhecimento de cada token é feito com o encontro de um espaço em branco por isso a verificação de delimitadores e operadores.

## 5. Análise Sintática

Para a verificação sintática é utilizado um gerador de gramática chamado goldparser onde denominamos a estrutura no qual nosso compilador reconhecerá a linguagem. Após gerar o arquivo do goldparser é analisado o arquivo e gerado um dicionário contendo os estados e produções geradas. Com esse dicionário é realizada a análise da pilha vinda do analisador léxico. Analisamos então cada estado final vindo na pilha e analisando a estrutura e ao mesmo tempo passando pelo dicionário do gold parser para verificar possíveis erros. Se não houver erros levantados, então o sintático somente retorna uma compilação feita com sucesso.

## 6. Conclusão

Não fui capaz de implementar todas as etapas necessárias a tempo, mas o projeto está vivo para ser terminado e possivelmente melhorado a um ponto na qual uma pessoa que não entende nada de linguagens de programação possa poder aprender a programar usando esta ferramenta ou até mesmo para os iniciantes de programação poder aprender sobre estruturas de dados usando este compilador. O projeto estará como público no GitHub para futuramente qualquer um possa melhorar o projeto do seu jeito.