Matheus Floriano Saito Da Silva

Analisador Léxico para a Linguagem C-: Projeto de Implementação utilizando Máquina de Moore

Relatório técnico de atividade prática solicitado pelo professor Rogerio Aparecido Gonçalves na disciplina de Teoria da computação do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR

Departamento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Fevereiro / 2025

Resumo

Este relatório apresenta os desenvolvimentos e resultados da atividade prática realizada, cujo objetivo foi projetar e implementar um analisador léxico para a linguagem C- por meio de uma máquina de Moore e linguagem de programação (Python).

Palavras-chave: Moore. Python. analisador léxico. implementar.

Sumário

1 Introdução e objetivos 4
2 Fundamentação 4
3 Materiais 4
4 Projeto do Autômato 4
5 Descrição do funcionamento do Código 5
5.1 Máquina de Moore 5
5.2 Pré-processamento 6
5.3 Processamento 6
6 Testes e Resultados 6
6.1 Correção dos testes 6
6.2 Testes Finais 9
7 Conclusão 11
8 Referências 11

1 Introdução e objetivos

Neste relatório será descrito o trabalho 1 da disciplina de Teoria da computação, o qual visa construir um analisador léxico para a linguagem C- criado por meio de uma máquina de Moore e a linguagem Python. Este analisador léxico recebe por entrada um arquivo contendo um código C- e processa este código devolvendo tokens das operações.

2 Fundamentação

Para a realização deste trabalho foi utilizado como base as instruções, o autômato inicial e o código base inicial fornecido pelo professor, além do capitulo 2 do livro do (LOUDEN, 2004).

3 Materiais

Foi utilizado um Notebook Dell G15 com as especificações:

- Intel Core i5-10500H (@2.50GHz e 12 *thread*)
- 16GB RAM
- SSD NVMe ADATA 512GB
- Pop!_OS 22.04 LTS

  E ferramentas :

- JFLAP Versão 7.1
- Python

4 Projeto do Autômato

Foi construído e utilizado como base o seguinte autômato:

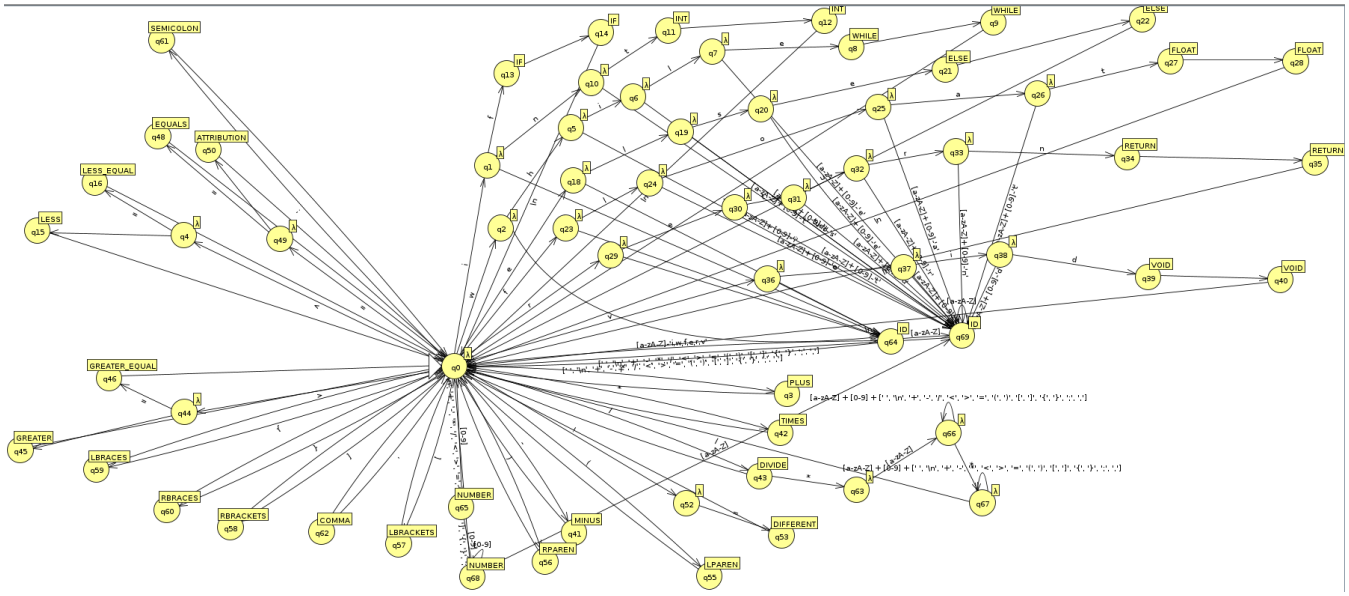

Figura 1 – Autômato Base

5 Descrição do funcionamento do Código

5.1 Máquina de Moore

A máquina de Moore do autômato apresentado (Figura 1) foi adaptado ao código por meio da automata-python. Esta é responsável por toda a análise léxica.
Seu comportamento base é:

- Todos os símbolos simples lidos retornam a respectiva saída
- Quando um caractere letra da tabela ascii não representa a inicial de uma palavra reservada ele é imediatamente considerado ID
- Quando o caractere letra faz parte de uma palavra reservada ele só se torna um ID se não completar a palavra reservada
- Dígitos são reconhecidos e quando detectam um símbolo retornam a q0, se aparecer um caractere no meio dos dígitos, o dígito termina retornando NUMBER e começa um ID
- Comentários são reconhecidos após ter um caractere "/" seguido de "*" e param de ser reconhecidos quando é detectado "*" seguido de "/"
- Quando uma palavra ou símbolo termina de processar e gera uma saída, a máquina volta para q0

5.2 Pré-processamento

```
def preprocess_input(input_string)
```

Esta função é responsável por tratar o código diretamente fornecido pelo arquivo, a ideia aqui é simplificar os dados passados para o autômato, neste caso o principal seria passar cada seção do código original em formato de: operação <quebra de linha>. Por exemplo:

"return(0);" seria transformado em :

```
return
(
0
)
;
```

Vale ressaltar que entre as palavras podem ser gerados vários espaços ou quebras de linha e o autômato está preparado para ignorá-los, portanto, esta função também foi utilizada para tratar um problema com operadores compostos, onde ao separar != por exemplo, causaria um problema de identificação.

5.3 Processamento

```
def process_input(input_string)
```

Esta função é responsável por lidar com os tokens sem si, obtendo as saídas da máquina de Moore e as colocando em uma lista de strings, onde cada string representa um token.

6 Testes e Resultados

6.1 Correção dos testes

Vale a pena ressaltar que alguns testes estavam errados, e um apenas foi ajustado para funcionar de acordo com a minha implementação como demonstrado a seguir:

Figura 2 – Teste 004

Na figura 2 podemos ver que a linha 32 descreve como INT quando na verdade é um ID.

Figura 3 – Teste 005

Na figura 3 vemos que estava faltando um RPAREN.

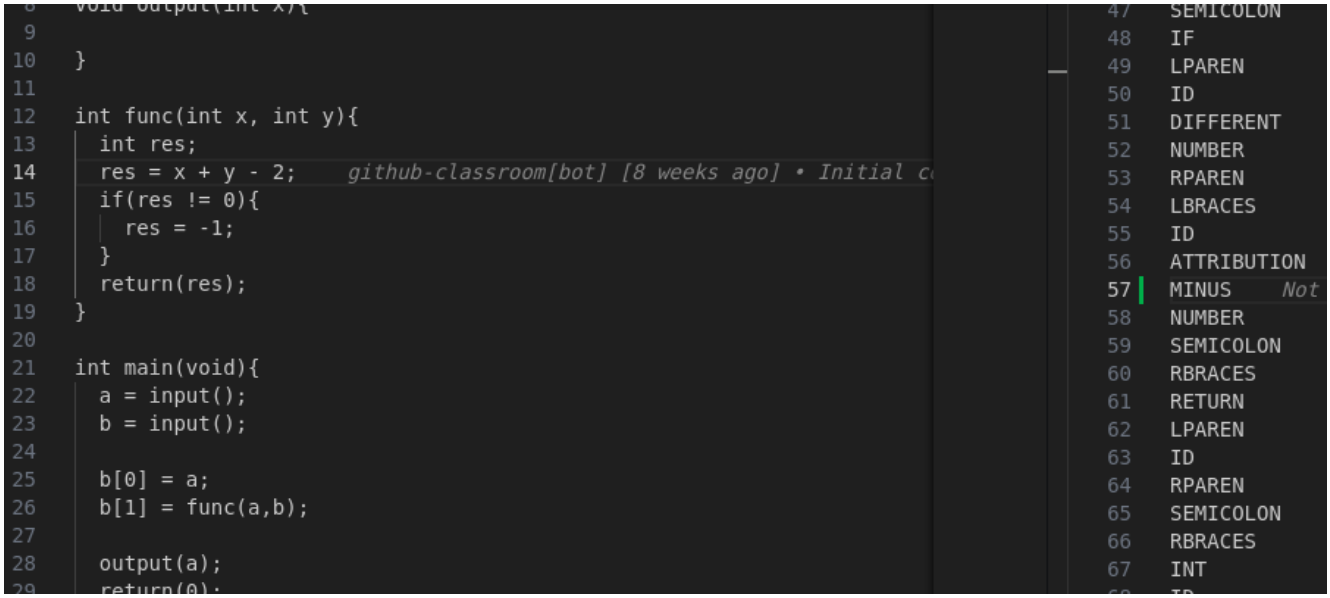

Figura 4 – Teste 006

Na figura 4 é possível ver que ocorre uma diferença entre implementações, no meu caso está sendo considerado MINUS, NUMBER, enquanto o teste original considera um número negativo como apenas NUMBER.

Figura 5 – Teste 007

Na figura 5 após o NUMBER tínhamos um SEMICOLON adicional após b = 10.

6.2 Testes Finais

Após todas as mudanças nos testes como descrito na seção 6.1 o resultado do teste final:

Figura 6 – Testes

Também fiz um arquivo de teste que demonstra todas as funções do lexer, e a comparação entre código e saída de tokens pode ser vista na Figura 7:

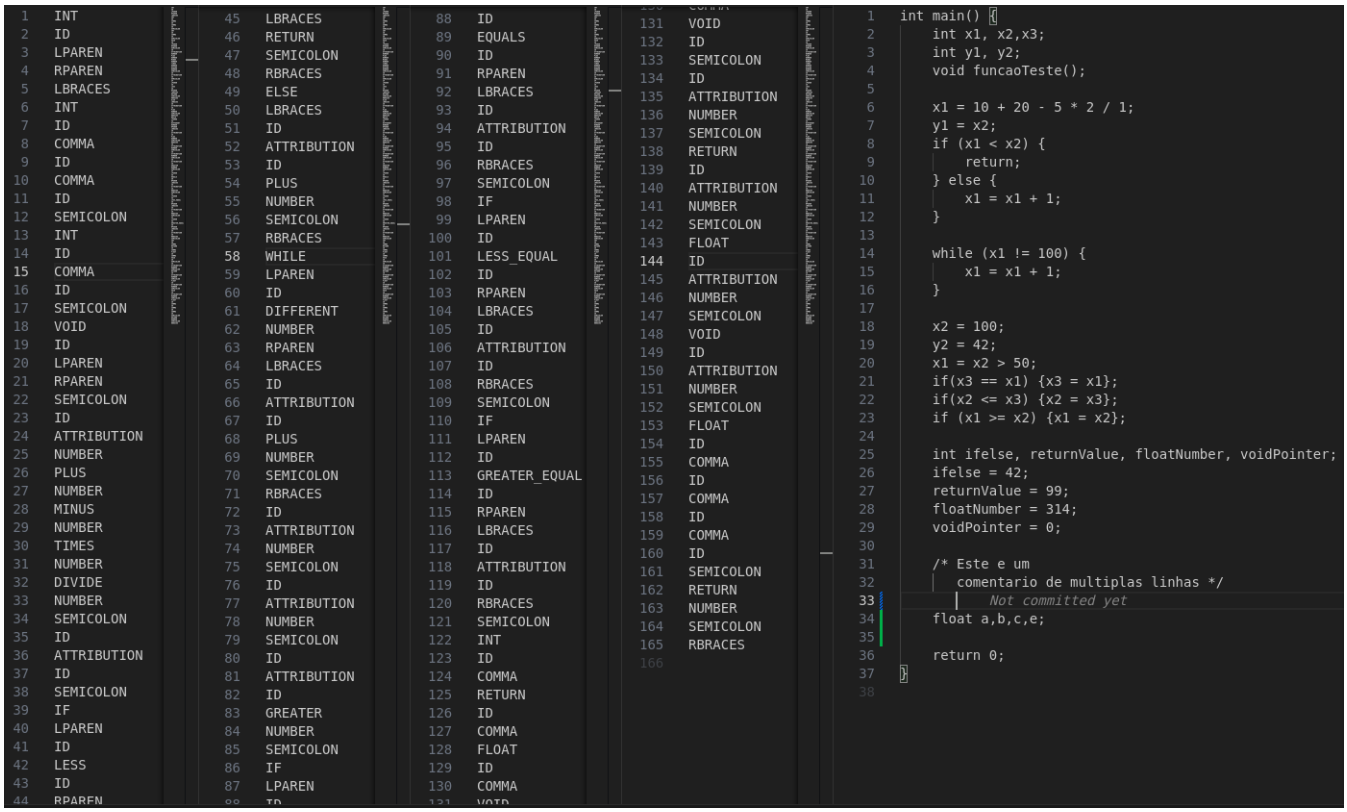

Figura 7 – MyTest

Saída do Teste 007 (Resultados começam na esquerda e terminam na direta):

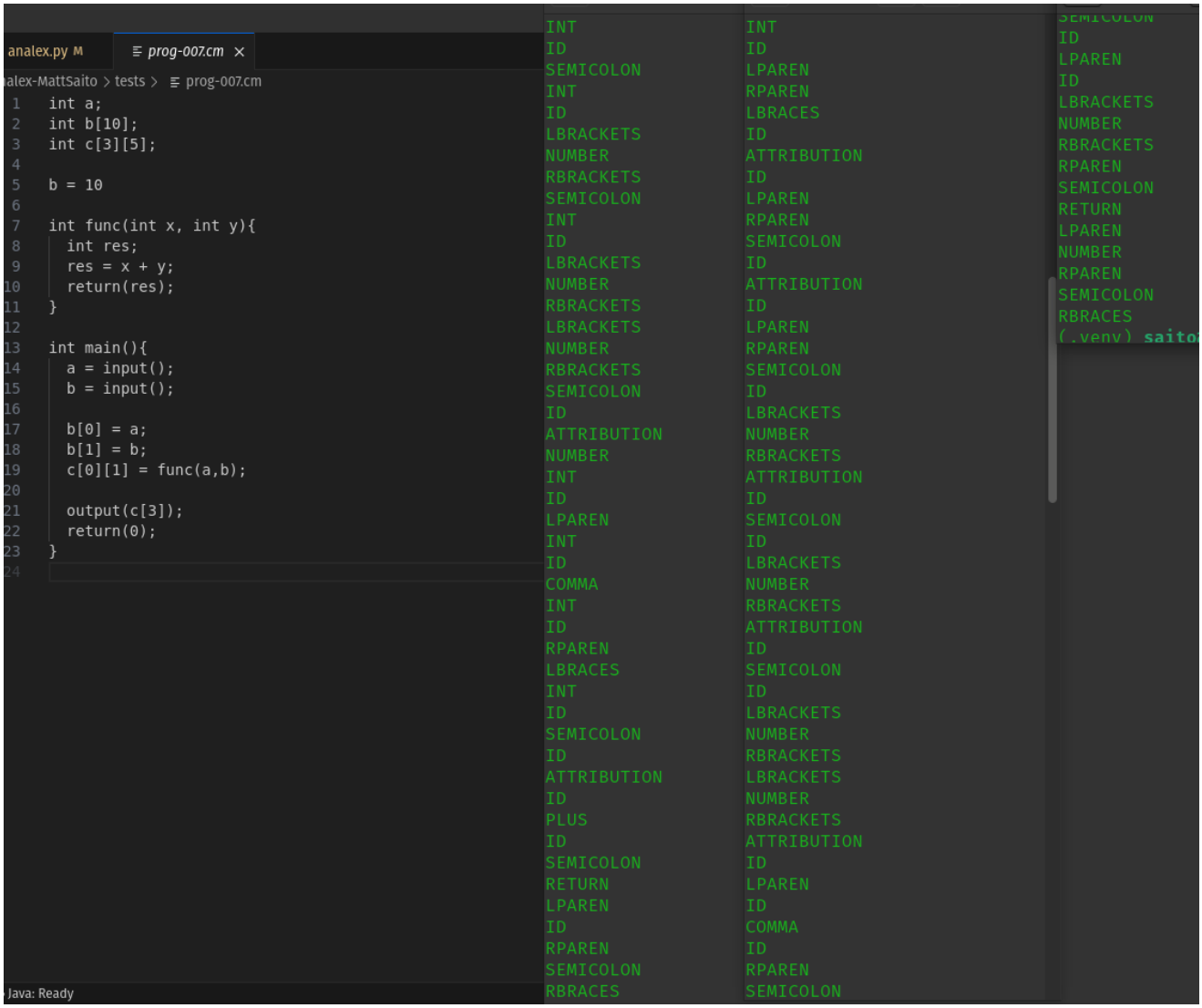

Figura 8 – Test-007

Saída do Teste 006 (Resultados começam na esquerda e terminam na direta):

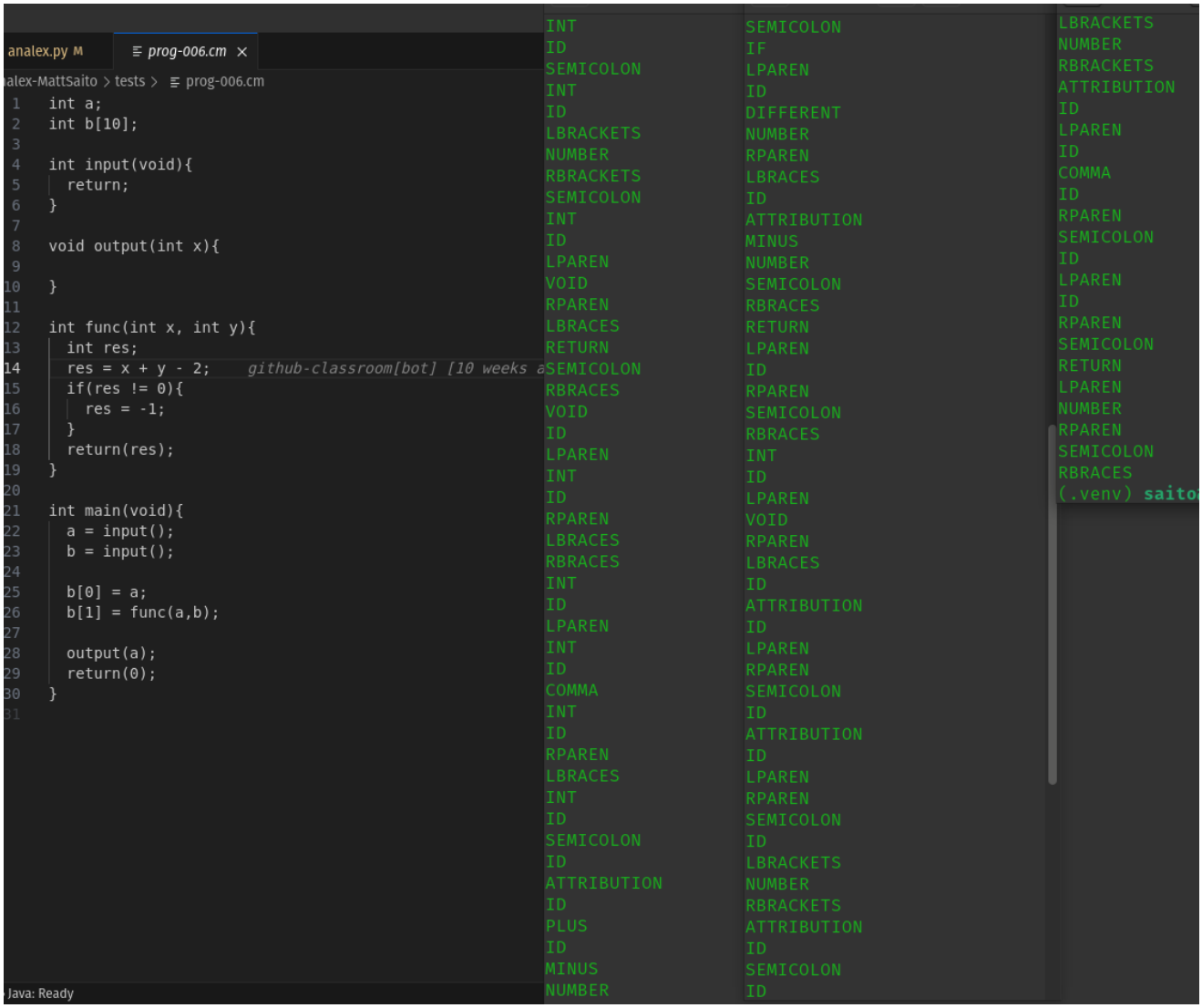

Figura 9 – Test-006

7 Conclusão

O projeto em si foi realizado com sucesso e os resultados finais foram favoráveis, durante a implementação do projeto foi possível obter mais conhecimento sobre um analisador léxico e sobre como lidar com autômatos em Python.

8 Referências

LOUDEN, K. C. *Compiladores: princípios e práticas*. 2004. [Accessed 27-01-2025]. Citado na página 4.