André Luiz Belmiro de Albuquerque Junior

Compiladores

Orientador: Leonardo

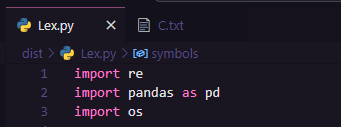
AV3 – Compilador

Conforme solicitado pelo professor Leonardo, o código criado tem como proposta a criação de um analisador Léxico que faça a leitura do código em C, como não foi solicitado que o código fosse criado em alguma linguagem em especifica, decidi criar o meu em Python.

1- Analisador Léxico

O Analisador Léxico é responsável por ler o código-fonte escrito em uma linguagem de programação e convertê-lo em uma sequência de tokens. Cada token identificado é classificado e atribuído a uma categoria específica, como operador, identificador ou número. Essa classificação é essencial, pois com ela geramos uma tabela de símbolos a qual será utilizada na próxima fase do processo de compilação, que é a análise sintática.

* 1. Bibliotecas utilizadas



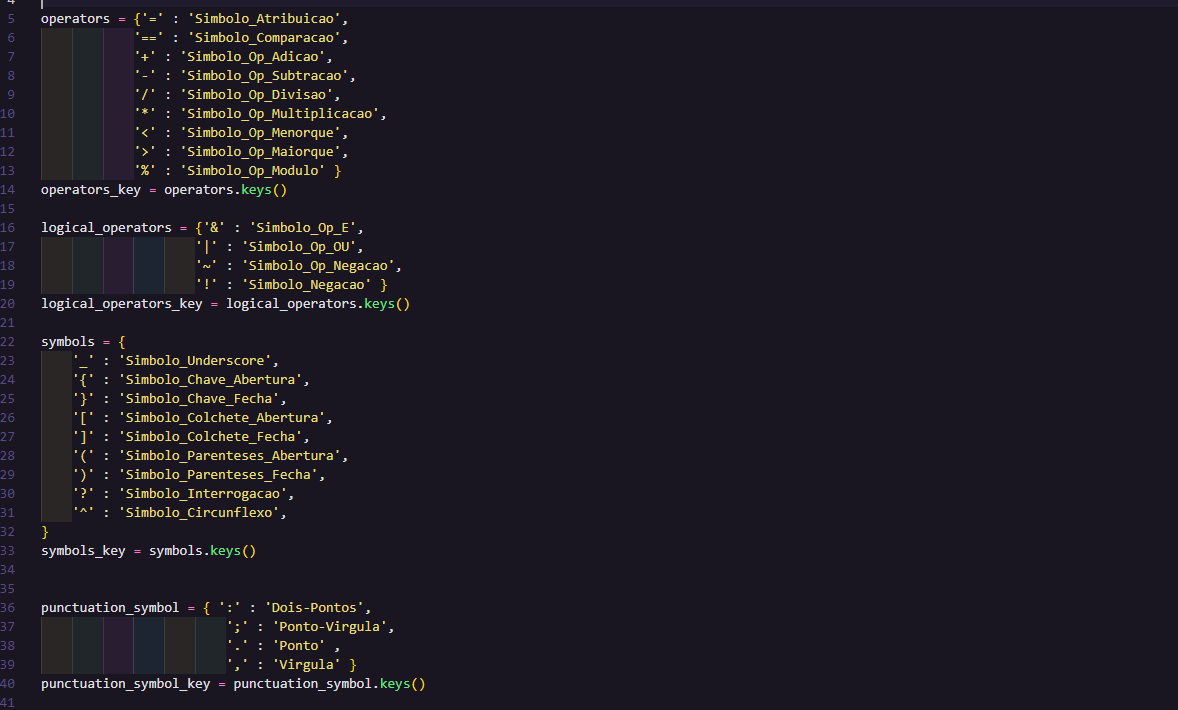
Nesse projeto foram utilizadas bibliotecas como o

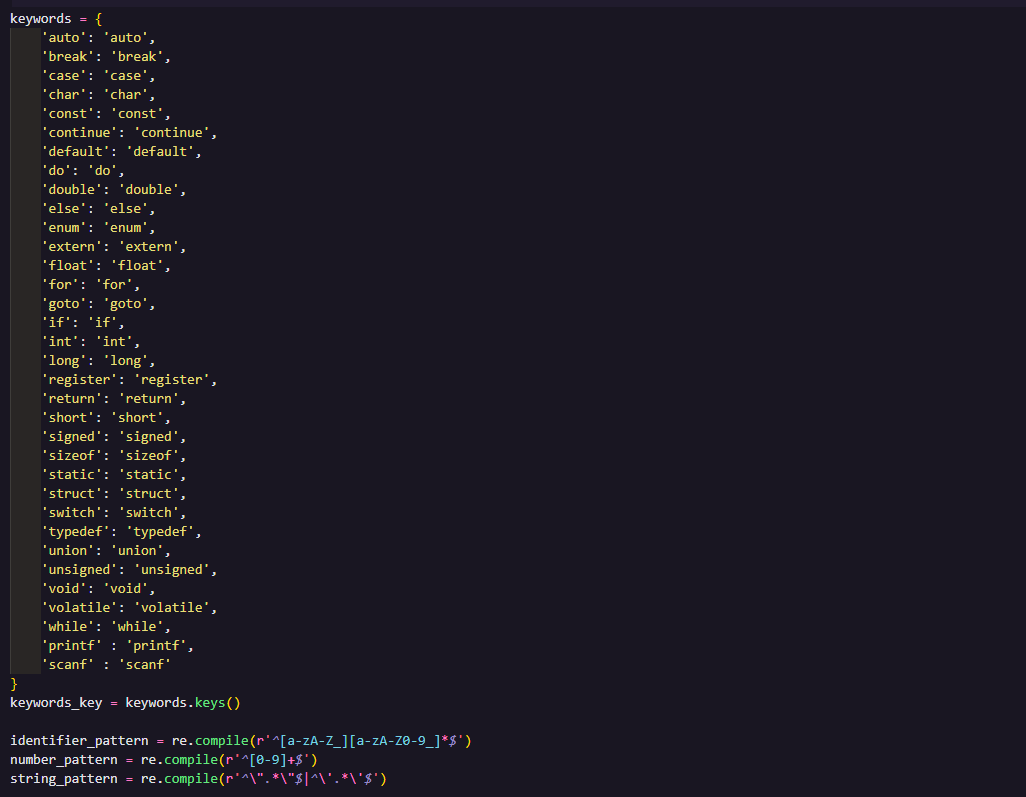
Re: um modulo do RegEx, que permitiu o uso de expressões regulares dentro do código.

Pandas: Utilizada para a manipulação e análise de dados, com ela foi possível gerar a tabela de símbolos dentro do excel.

OS: Permite a uma interação maior com o sistema operacional e suas funções, ela foi utilizada para verificar se a tabela de símbolos já foi gerada afim de evitar sobrescrever a existente.

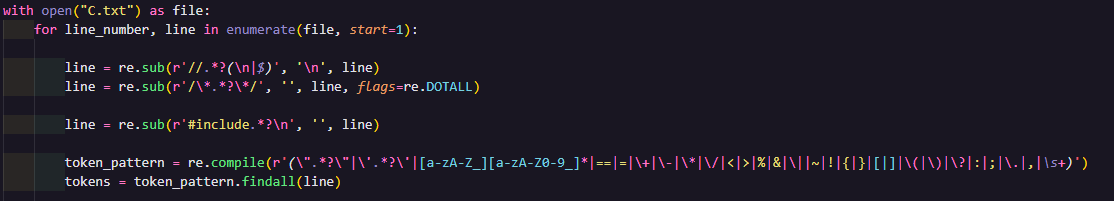
1.2 Tokens da linguagem C





Aqui foi criado o dicionário da linguagem C, definido suas palavras reservadas, operadores, símbolos e etc. Além disso, foram criadas variáveis as quais armazenam as chaves dos dicionários correspondentes, essas chaves são usadas para verificar se um token corresponde a alguma estrutura do dicionário ou não.

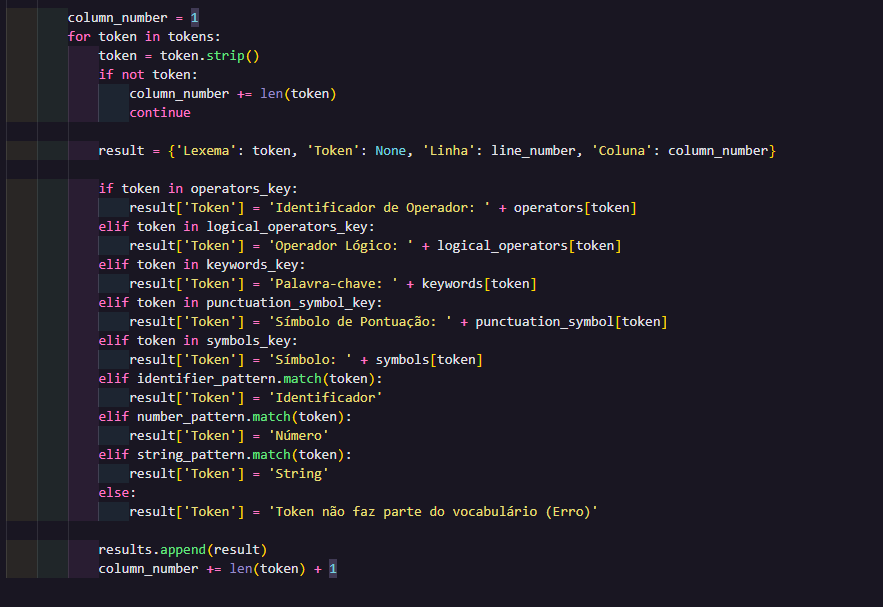
1.3 – Abertura do código C e organização



Nessa parte, realizamos a abertura do arquivo contendo o código C para leitura, e temos uma função **enumerate**, o qual nos permite identificar a linha original do código ao gerar a tabela de símbolos.

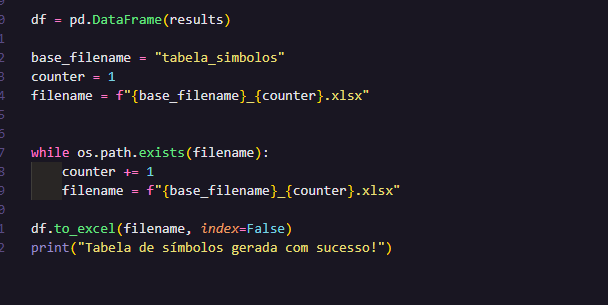
Os **line** abaixo são para a remoção de possíveis comentários ou bibliotecas que possam estar inclusas no arquivo contendo o código C. E por ultimo temos a padronização das expressões regulares para identificar diferentes tipos de tokens na linha, isso me permitiu realizar a captura de **strings**, a qual estava se mostrando uma dificuldade considerando que é possível ter espaçamento e diversos tipos de caracteres dentro de uma.

1.4 – Leitura do código C e organização



No início temos a utilização de um rastreamento da posição do token dentro da linha atual, junto de um loop que nos permite percorrer sobre todos os tokens obtidos da linha do código, assim removendo os espaçamentos em brancos. O **result** cria um dicionário o que será adicionado no fim a lista **results**, os dois permitirão a organização dos dados na tabela de símbolos depois. Logo em seguida temos a verificação dos elementos do código em C e sua checagem dentro do dicionário criado anteriormente.

1.5 Transferência dos resultados para o excel.



Utilização do **panda** e do **os** para a criação do arquivo excel contendo o resultado do analisador léxico, e por fim um print sinalizando o sucesso na criação da tabela.

1. Código C utilizado

Segue código C utilizado como teste, ele tem como função a soma de 2 variáveis inteiras, o código pode ser alterado dentro do arquivo txt.

#include <stdio.h>

int main() {

int num1, num2, soma;

// Solicita a entrada do primeiro número

printf("Digite o primeiro número: ");

scanf("%d", &num1);

// Solicita a entrada do segundo número

printf("Digite o segundo número: ");

scanf("%d", &num2);

// Realiza a soma dos dois números

soma = num1 + num2;

// Exibe o resultado

printf("A soma de %d e %d é: %d\n", num1, num2, soma);

return 0;

}

1. Tabela Símbolos

Está é gerada e repassada pro excel para melhor visualização, segue print abaixo com exemplo.

