Nome: Lucas Mateus Fernandes RA: 0035411

Analizador Lexico apresentado à disciplina de Compiladores, como requisito parcial de conclusão da matéria. Implementado por Lucas Mateus Fernandes RA:0035411

Linha de comando

main.py [-h] [-t PATH_OUT] [-s SIZEBUFFER] [-p] PATH_SOURCE

Comando	Significado
Argumentos posicionais:	
PATH_SOURCE	Caminho para o arquivo que vai ser analisado
Argumentos opcionais:	
[-h, - -help]	Mostra a mensagem de ajuda
[-t , - -table] PATH_OUT	Caminho para armazenar a tabela de simbolos
[-s , - -size]	Tamanho do Buffer de leitura
[-p, - -pretty]	Printa a saida na tela

Arquivos

main.py

Arquivo responsavel por analisar as flags usadas na linha de comando e executar o comando conforme os parametros.

Scan.py

main(self, fluxo , tamanho): Nucleo do analizador léxico responsavel por filtrar todos os tokens do arquivo fazendo uso de um buffer de tamanho

loadFile(self, nomeArquivo): Função que carrega um ponteiro para um arquivo que será tratado como um streamer de dados

stream(self, buffer, fileSource, num): Transfere caracteres do arquivo para o buffer

nextToken(self, buff, fluxo): Retorna um token

_renewBuff(self, buff, fluxo): Faz uma descarga parcial do buffer e o attualiza

Identifier.py

Arquivo responsavel por verificar se uma string é compativel com uma regex

O atributo typesRe é responsavel por armazenar as exprexões regulares correspondentes aos tokens

vetor(self,string): Verifica todas as expressões regulares que são compativeis com o lexema

finishToken(self, vetAnterior,vetPosterior): Um token termina quando um lexema para de ser compativel com uma expressão regular

Buffer.py

Classe para manipula o buffer, responsavel para garantir o estado do token, um token é limitado ao tamanho do buffer pois não pode existir um lexema que não caiba no buffer

getString(self): Retorna os caracteres que determinam um possivel lexema

getCaracter(self): Retorna um caracter para ser concatenado ao ultimo lexema

empty(self): Verifica se o Buffer esta vazio

renew(self): Faz uma descarga parcial do buffer, do inicio até a posição atual do ponteiro

add(self, item): Adiciona um elemento ao buffer

next(self): Avança o ponteiro dentro do buffer (tentativa gulosa de pegar o maior lexema possivel)

back(self): Retrocede o ponteiro dentro do buffer (tentativa de backtrack)

Token.py

Classe responsavel por cria uma estrutura que representa um token contendo "lexema" , "tipo","linha".

Funcionamento

O analizador lexico faz o uso de um buffer responsavel pela transição linear do conteudo do arquivo para o tratamento de um lexema.

Um lexema é composto de 'n' caracteres presente do inicio do buffer

Um token é composto pelo maior lexema que corresponda a alguma Regex anteriormente definida.

se o lexema passou no teste passa para o lexema + 1 caracter caso contrario cria um token com lexema -1

Caso o primeiro caracter do buffer seja um '#' específico de comentarios é analisado até o proximo "n e faz uma descarga do buffer e reinicia o lexema

Caso o primeiro caracter do buffer seja algum caracter de identação faz uma descarga do buffer e reinicia o lexema