Sumário

Relatório - Analisador léxico

1. Introdução:

Com o objetivo de implementar os conhecimentos desenvolvidos na disciplina, a respeito do analisador léxico a partir do conceito de autômatos finitos determinísticos, escrevemos um código em C para fazer a análise léxica da linguagem Micro Pascal ou Mini Pascal. Nesta atividade utilizamos utilizamos o Jflap para construirmos o autômato finito determinístico que pudesse representar os estados e as transições das entradas e símbolos da linguagem mini Pascal. Após a construção do autômato, construímos um código capaz de ler um arquivo, reconhecer tokens e adicionar símbolos em uma tabela de símbolos, além de no final salvar um arquivo de saída na extensão .lex listando os tokens reconhecidos.

Link do github: https://github.com/Rodrooj/Compilador_C

2. Estruturas:

2.1 TokenType (enum):

Define os tipos de tokens que o analisador pode reconhecer, incluindo palavras-chave, operadores, pontuação e outros elementos léxicos.

Fizemos o uso de duas structs:

2.2 Token (Struct):

Representa um token individual com os seguintes campos:

- Type: O tipo de token (TokenType);
- Lexame: A sequência de caracteres que forma o token;
- Line: Linha onde o token foi encontrado;
- Column: Coluna onde o token foi encontrado.

2.3 HashTable (Struct):

- Implementa uma tabela hash para armazenar as palavras reservadas e identificadores da linguagem, facilitando a identificação rápida desses tokens.

3. Funções principais:

3.1 create_token:

Token *create_token(TokenType type, const char *lexeme, int line, int column)

- Cria os tokens, sendo do tipos definidos no enum com sua representação e a linha e coluna encontrada.

3.2 free_token:

void free_token(Token *token)

- Libera a memória alocada para um token.

3.3 get_next_token:

Token *get_next_token(FILE *file, int *line, int *column)

- Função principal do analisador léxico;
- Lê o próximo token do arquivo de entrada e gera uma saída.

3.4 init_keyword_table:

void init_keyword_table()

- Inicializa a tabela hash com a tabela de símbolos.

3.5 get_keyword_token:

TokenType get_keyword_token(const char *lexeme)

- Verifica se um lexema é uma palavra reservada ou identificador e retorna o tipo de token correspondente.

Função principal (main)

- Verifica se o usuário passou um arquivo de entrada para a análise léxica, se não, reportará o erro “Erro ao abrir arquivo”;
- Inicializa a tabela hash chamando a função init_keyword_token e também inicializa o número da linha e da coluna;
- Em um laço de repetição verifica se os token são reconhecidos até o fim do arquivo ou a geração de uma mensagem de erro caso não seja reconhecido;
- Fecha o arquivo e libera a memória alocada, criando por fim um arquivo de saída .lex.

4. Testes:

Submetemos seis códigos à análise léxica, sendo três códigos sem erro léxico e três com erros:

4.1 Testes certos:

Arquivos certo: certo1.pas, certoo2.pas e certo3.pas:

- O Analisador léxico se mostrou eficiente ao reconhecer palavras reservadas e identificadores e símbolos no geral;
- A localização, sendo a linha e a coluna, foi registrada com precisão.

4.2 Testes errados:

- errado1.pas: Detectou um comentário não fechado;
- errado2.pas: O caractere (“) foi reconhecido como um caractere inválido;
- errado3.pas: O Caractere (!) foi tido como desconhecido.

5. Conclusão

Concluímos, portanto, este trabalho que implementa um analisador léxico em linguagem C que analisa a linguagem micro Pascal. Desenvolvemos um autômato finito determinístico e desenvolvemos um código capaz de reconhecer lexemas e relacioná-los a tokens. Nos testes apresentou um resultado satisfatório cumprindo em grande parte o que foi especificado pelo professor.