

Analizador Léxico para MicroPascal

Descrição

Este projeto implementa um **analizador léxico** para a linguagem **MicroPascal**, responsável por ler um arquivo de código-fonte, identificar tokens e categorizá-los de acordo com a gramática da linguagem. Os tokens reconhecidos incluem:

- **Identificadores**
- **Palavras reservadas**
- **Operadores** (+, -, *, /, >, <, =, etc.)
- **Símbolos** (parênteses, chaves, ponto e vírgula, etc.)
- **Números** (inteiros e reais)

O analisador lê o arquivo de entrada, realiza a análise léxica e gera um arquivo de saída contendo a lista dos tokens identificados, suas respectivas posições no código e a tabela de símbolos.

Diagrama do AFD

O diagrama abaixo ilustra o Automato Finito Determinístico (AFD) usado para reconhecer tokens na linguagem MicroPascal. Ele mostra como o analisador léxico identifica diferentes tokens a partir de seus estados e transições.

?

Funcionalidades

- **Análise léxica completa:** Reconhecimento de tokens válidos conforme as regras da linguagem MicroPascal.
- **Relatório de erros léxicos:** Detecção e notificação de caracteres inválidos, strings e comentários não fechados, com indicação precisa da linha e coluna.
- **Tabela de símbolos:** Armazena palavras reservadas e identificadores únicos.
- **Ignora espaços e comentários:** Espaços em branco, quebras de linha e comentários são ignorados durante a análise.
- **Arquivo de saída:** Geração de arquivo .lex com tokens reconhecidos e sua posição no código.

Estrutura do Código

Funções Principais

```
int main(int argc, char *argv[])
```

Coordena a execução do analisador léxico.

- **Parâmetros:**

- argc: Número de argumentos passados ao programa.
- argv[]: Lista de argumentos.
 - argv[1]: Arquivo de entrada.
- **Descrição:**
 - Verifica a presença de um arquivo de entrada.
 - Abre o arquivo de entrada e cria o arquivo de saída.
 - Realiza a análise chamando analisar_lexico().

*void exibir_ajuda(const char *nome_programa)*

Exibe instruções detalhadas sobre como usar o analisador.

- **Parâmetros:**
 - nome_programa: Nome do executável.

void exibir_lista_de_tokens()

Mostra a lista de tokens reconhecidos pelo analisador.

*void analisar_lexico(FILE *arquivo, FILE *saida)*

Realiza a análise léxica do arquivo de entrada e escreve os tokens no arquivo de saída.

- **Descrição:**
 - Processa o arquivo, identificando e categorizando os tokens.
 - Tokens são gravados no arquivo no formato <TOKEN_NOME, TOKEN_LEXEMA> na linha X, coluna Y.

*Token obter_token(FILE *arquivo)*

Identifica e retorna o próximo token do arquivo de entrada.

void iniciar_tabela_de_simbolos()

Inicializa a tabela de símbolos com as palavras reservadas da linguagem MicroPascal.

*void reportar_erro(char *mensagem, int linha, int coluna)*

Exibe uma mensagem de erro com a linha e a coluna do problema.

Estruturas

typedef struct Token

Estrutura que define um token.

- **Campos:**
 - char nome[MAX_TOKEN_LENGTH]: Nome do token.
 - char lexema[MAX_TOKEN_LENGTH]: Lexema do token.
 - int linha: Linha do token.

- int coluna: Coluna do token.

Arquivo de Saída

O arquivo de saída é gerado com o nome do arquivo de entrada acrescido da extensão `.lex`.
Contém uma linha para cada token reconhecido, no seguinte formato:

`<TOKEN_NOME, TOKEN_LEXEMA>` na linha `X`, coluna `Y`

Exemplo:

`<ID, program>` na linha 1, coluna 1
`<ID, exemplo>` na linha 1, coluna 9
`<SMB_OPA, (>` na linha 1, coluna 16
...

Executando o Programa

Sintaxe

`./analizador_lexico <arquivo_de_entrada>`

Opções

- `-h` ou `--help`: Exibe a ajuda.
- `--tokens`: Lista os tokens suportados.

Exemplos

1. Para analisar um arquivo de código MicroPascal:
`./analizador_lexico programa.pas`
2. Para listar os tokens reconhecidos:
`./analizador_lexico --tokens`

Tokens Suportados

Palavras Reservadas

- `program`, `var`, `integer`, `real`, `begin`, `end`, `if`, `then`, `else`, `while`, `do`, `write`, `read`

Operadores

- `+` (Adição)
- `-` (Subtração)
- `*` (Multiplicação)
- `/` (Divisão)
- `=` (Atribuição)
- `>` (Maior que)
- `>=` (Maior ou igual)
- `<` (Menor que)
- `<=` (Menor ou igual)

- <> (Diferente)
- := (Atribuição)

Símbolos

- ; (Ponto e vírgula)
- , (Vírgula)
- { (Chave de abertura)
- } (Chave de fechamento)
- ((Parêntese de abertura)
-) (Parêntese de fechamento)
- . (Ponto final)
- : (Dois pontos)

Outros Tokens

- ID (Identificadores)
- NUM (Números)

Relatório de Erros Léxicos

O analisador gera mensagens de erro como:

Erro léxico: Caractere inválido na linha X, coluna Y

Requisitos

- **Compilador C:** GCC ou equivalente para compilar o projeto.

Exemplo de comando:

```
gcc -o analisador_lexico lexer.c main.c
```