Introdução

O código feito na linguagem micro pascal tem uma analisador léxico para ler a partir de um arquivo de entrada saber e classificar diferentes tokens em um arquivo de saída, assim capaz de reportar erros e armazenar informações sobre cada token reconhecido e `TabelaSimbolos` para manter um registro dos identificadores.

Como funciona:

Abre um terminal e escreva: gcc Analizador_lexico.c -o Analizador_Lexico

Isso serve pra compilar.

Depois escreva: ./Analizador_lexico

Gera o arquivo executável

Irá aparecer uma mensagem na tela, escreva o nome do arquivo que será analisado no caso: program.mpas

O código apresenta 6 exemplos de teste certos e 6 exemplos de testes errados.

Exemplo de testes certos:

Exemplo-1

```
var
   a: integer;
begin
   a := 10;
end.
```

Exemplo-2

```
var
   x, y: real;
begin
   x := 5.0;
   y := x + 3.5;
end.
```

Exemplo-3

```
begin
```

```
    if x > y then
        writeln('x é maior');
end.
```

Exemplo-4

```
var
    i: integer;
begin
    for i := 1 to 10 do
        writeln(i);
end.
```

Exemplo-5

```
var
    a, b: integer;
begin
    a := 5;
    b := a * 2;
    if b > 10 then
        writeln('Maior que 10');
end.
```

Exemplo-6

```
var
    nome: string;
begin
    nome := 'MicroPascal';
    writeln('Olá, ', nome);
end.
```

Exemplo de testes errados:

Exemplo-1 O erro é o “1a” invalido.

```
var
    1a: integer;
begin
```

```
    a := 10;
end.
```

Exemplo-2 O erro é o :=

```
var
    b: integer;
begin
    b = 20;
end.
```

Exemplo-3 O erro é que não tem atribuição o operador.

```
var
    c: real;
begin
    c + 5;
end.
```

Exemplo-4 o erro é o “;” faltando.

```
var
    x: integer;
begin
    x := 10
end.
```

Exemplo-5 O operador está errado na atribuição.

```
var
    y: real;
begin
    y = 3.14;
end.
```

Exemplo-6 o erro é o sem um operando um operador invalido.

```
var
    z: integer;
begin
    z := 5 + * 3;
```

end.

Explicações do proposito de todas as structs e funções.

Exemplo de tokem com o proposito de ser reconhecido pelo analisador léxico.

```
typedef struct {
   char nome[MAX_LEN];
   char lexema[MAX_LEN];
   int linha;
   int coluna;
} Token;
```

Exemplo de TabelaSimbolos utilizado na análise semântica ou na geração de código.

```
typedef struct {
   char simbolo[MAX_LEN];
} TabelaSimbolos;
```

Exemplo de ePalavraReservada verificar se é lexema se sim retorna 1 se não retorna 0.

```
int ePalavraReservada(const char *lexema);
```

exemplo de esimbolos verificar se o lexema é reconhecido pelo programa se sim retorna 1 se não retorna 0.

```
int eSimbolo(const char *lexema);
```

exemplo de adicionarTabelaSimbolos colocar um lexema na tabela de símbolos.

```
void adicionarTabelaSimbolos(TabelaSimbolos *ts, int *tamanhoTS, const char *lexema);
```

exemplo de detectarerro inclui em saber em qual linha o erro está.

```
 void detectarErro(const char *erro, int linha, int coluna);
```

exemplo de reconhecerTokens faz a análise léxica do código fonte, identificar diferentes tipos de tokens e registrando erros quando necessário.

```
void reconhecerTokens(FILE *entrada, FILE *saida, TabelaSimbolos *ts, int *tamanhoTS);
```

exemplo de main ponto de entrada do programa.

```
int main();
```

exemplos de saída do analixador léxico do código:

exemplo-1

<PALAVRA_RESERVADA, program> [linha: 1, coluna: 1]

<ID, exemplo> [linha: 1, coluna: 9]

<PALAVRA_RESERVADA, var> [linha: 2, coluna: 1]

<ID, x> [linha: 2, coluna: 5]

<SIMBOLO, :> [linha: 2, coluna: 6]

<PALAVRA_RESERVADA, integer> [linha: 2, coluna: 8]

<PALAVRA_RESERVADA, begin> [linha: 3, coluna: 1]

<ID, x> [linha: 4, coluna: 1]

<SIMBOLO, :=> [linha: 4, coluna: 3]

<NUMERO, 10> [linha: 4, coluna: 5]

<PALAVRA_RESERVADA, end> [linha: 5, coluna: 1]

<SIMBOLO, .> [linha: 5, coluna: 4]

Exemplo -2

<PALAVRA_RESERVADA, program> [linha: 1, coluna: 1]

<ID, teste> [linha: 1, coluna: 9]

<PALAVRA_RESERVADA, var> [linha: 2, coluna: 1]

<ID, a> [linha: 2, coluna: 5]

<SIMBOLO, ,> [linha: 2, coluna: 6]

<ID, b> [linha: 2, coluna: 8]

<SIMBOLO, :> [linha: 2, coluna: 9]

<PALAVRA_RESERVADA, real> [linha: 2, coluna: 11]

<PALAVRA_RESERVADA, begin> [linha: 3, coluna: 1]

<PALAVRA_RESERVADA, if> [linha: 4, coluna: 1]

<ID, a> [linha: 4, coluna: 3]

<SIMBOLO, < > [linha: 4, coluna: 5]

<ID, b> [linha: 4, coluna: 7]

<PALAVRA_RESERVADA, then> [linha: 4, coluna: 9]

<ID, a> [linha: 5, coluna: 5]

<SIMBOLO, :=> [linha: 5, coluna: 7]

<ID, a> [linha: 5, coluna: 10]

<SIMBOLO, +> [linha: 5, coluna: 12]

<NUMERO, 1> [linha: 5, coluna: 14]

<PALAVRA_RESERVADA, end> [linha: 6, coluna: 1]

<SIMBOLO, .> [linha: 6, coluna: 4]

Exemplo-3

<PALAVRA_RESERVADA, program> [linha: 1, coluna: 1]

<ID, erro1> [linha: 1, coluna: 9]

<PALAVRA_RESERVADA, var> [linha: 2, coluna: 1]

<ID, a> [linha: 2, coluna: 5]

Então o código feito implementa um analisador léxico para o microPascal conseguindo identificar palavras reservadas, números, símbolos e identificadores dando origem a uma lista de tokens também podendo identificar erros de tokens inválidos e podendo avisa-los quando dão erro e onde, além de identificar e armazenar os tokens e exibi-los no console junto com a tabela de símbolos, desenvolvendo uma base solida para compiladores mais complicados aumentando a eficiência.