

CAMILA GOIS DE JESUS

DESCRIÇÃO DO CÓDIGO ANALISADOR LÉXICO

SALVADOR
14 de Dezembro de 2019

DESCRIÇÃO DO CÓDIGO ANALISADOR LÉXICO

Bibliotecas

```
#include <stdlib.h> // CONVERSÃO, MEMÓRIA, CONTROLE DE PROCESSO  
#include <stdio.h> // ENTRADA E SAÍDA (PRINTF, SCANF, FPRINT, FSCANF)  
#include <string.h> // CADEIAS DE CARACTERES E REGIÕES DE MEMÓRIA  
#include <windows.h> // ESPECÍFICO DO WINDOWS
```

Declaração de variáveis e seus respectivos tipos

Contador de linha do tipo inteiro ; Contador de Caractere do tipo inteiro
Contador de Aspas do tipo inteiro ; Contador de Chaves do tipo inteiro
Contador de Parênteses do tipo inteiro ; Contador do Erro do Caractere do tipo inteiro;
Erro do Código do tipo inteiro ; String (Vetor de Caracter) 550 posições;
String (Armazenamento) 550 bytes

typedef <nome do tipo de dado> <novo nome>:

Tipo Inteiro, Linha

typedef struct Redefinir struct (usando nomes mais curtos) // CÓDIGO ID e DICIONÁRIO

Inteiro , nElemento

String Vetor de caractere (palavra) 550 posições

typedef struct aux : Usando os nomes curtos para : AUXILIAR(aux) , REGISTRO(regt)

struct aux* prx; Struct Ponteiro (auxiliar apontando para o próximo)
ELEMENTO,* P; Ponteiro (P)

typedef struct: // LISTA , PONTEIRO INÍCIO (P inicio) , PONTEIRO FIM (P fim)

LISTA LL; (DECLARAÇÃO DE LISTA - VARIÁVEL LL)

void inicializarLISTA(LISTA* LL): INICIALIZANDO LISTA com Ponteiro (chamada da função)

int tamanho(LISTA* LL) : TAMANHO (TIPO INTEIRO) COM PONTEIRO (LL)

IGUALANDO PONTEIRO DO ENDEREÇO A LL NO INÍCIO DA LISTA

TAM DO TIPO INTEIRO

WHILE QUANDO ENDEREÇO FOR DIFERENTE DE ZERO

INCREMENTAÇÃO DE TAM

ENDEREÇO = ENDEREÇO APONTANDO PARA O PRÓXIMO
RETORNANDO A TAM

```
int exibirLISTA(LISTA* LL) // EXIBIÇÃO DE LISTA VAZIA
IGUALANDO PONTEIRO DO ENDEREÇO A LL NO INÍCIO DA LISTA
    SE LL FOR IGUAL A ZERO FAÇA:
        IMPRIMA LISTA VAZIA
        RETURN A ZERO
    SE NÃO
        IMPRIMA PALAVRAS ANALISADAS
        WHILE ENQUANTO O ENDEREÇO FOR DIFERENTE DE ZERO
            IMPRIMA PALAVRA (ENDEREÇO RECEBE REGISTRO DA PALAVRA)
            IMPRIMA NÚMEROS DE ELEMENTOS (ENDEREÇO RECEBE REGISTRO DO
            NÚMERO DE ELEMENTOS)
            ENDEREÇO = ENDEREÇO RECEBE APONTADOR PRÓXIMO
```

```
int salvarCodigo(char *token, LISTA* LL,int elementos) // SALVAR CÓDIGO(
TOKEN PONTEIRO DO TIPO CARACTER, PONTEIRO LISTA (LL), ELEMENTOS
DO TIPO INTEIRO
    IMPRIMA TOKEN DO TIPO STRING
    P NOVO = ALOCAÇÃO DE MEMÓRIA DO ELEMENTO
    NOVO RECEBE REGISTRO DA LINHA = CONTADOR DE LINHA
    CÓPIA DA STRING (ORIGEM, DESTINO)
    NOVO RECEBE REGISTRO DE NÚMERO DE ELEMENTOS = ELEMENTOS
    NOVO RECEBE PROXIMO = ZERO
        SE LL RECEBE INICIO == 0 E LL RECEBE INICIO = NOVO
    SENAO
        LL RECEBE FIM QUE RECEBE PROXIMO = NOVO
        LL RECEBE FIM = NOVO
    RETORNA 1
```

```
int verificaNumero(char *numero) // INTEIRO (VERIFICAR NÚMERO) COM
PONTEIRO DO TIPO CHAR
CONTADOR DE NÚMERO O TIPO INTEIRO = 0
INTEIRO VALOR = CONVERSÃO DE INTEIRO EM STRING (NUMERO)
SE COMPARAÇÃO DE DUAS STRING (NUMERO, "0") == 0
    INCREMENTA CONTADOR DE NÚMERO
SE VALOR ==0
    INCREMENTA CONTADOR DE NÚMERO
    RETORNE CONTADOR DE NÚMERO
```

```
void inicio(LISTA* LL){ // INICIO DA LISTA , COMPARANDO AS PALAVRAS SE
AS MESMAS ESTÃO PRESENTES NA LISTA
PONTEIRO ENDEREÇO = LL RECEBE INCIO
VETOR AUX (550 POSIÇÕES) DO TIPO CHAR
IMPRIMIR PALAVRAS VÁLIDAS
While (ENDEREÇO DIFERENTE DE ZERO)
FAÇA
AS STRING DE COMPARAÇÃO E IMPRIMA PALAVRA VÁLIDA (NUM, STRING,
PRINT, READ, RETURN, IF, ELSE, SQRT, STEP, FOR , TO , FECHAMENTO DE
CHAVE, PARENTESSES, ASPAS, OPERADORES RELACIONAIS, ARITMÉTICOS)
```

```
int verificaFechamento() // VERIFICAÇÃO DE FECHAMENTO DO CÓDIGO
APONTA LOCALIZAÇÃO DE UM REGISTRO NO CASO O ARQUIVO
ABERTURA DE ARQUIVO
IMPRIME CHAVES, PARÊNTESES, ASPAS, E OS CONTADORES DOS MESMO
DO TIPO INTEIRO
COMPARA ARQUIVO COM 0 E IMPRIME ERRO NA ABERTURA DO CÓDIGO
RETORNANDO A 1
    SE CONTADOR DE CHAVES , PARÊNTESES E ASPAS = 0
    IMPRIMA ARQUIVO DO TIPO STRING COM A INFORMAÇÃO “FALTA
    FECHAMENTO DO CÓDIGO”
    INCREMENTA ERROCODIGO
    FECHA ARQUIVO
```

```
int verificaErros(){ // VERIFICAÇÃO DE ERROS DO ARQUIVO
    APONTA LOCALIZAÇÃO DE UM REGISTRO NO CASO O ARQUIVO
    ABERTURA DE ARQUIVO
        SE ARQUIVO == 0
        IMPRIMA “ERRO NA ABERTURA DO ARQUIVO “
        RETORNANDO A 1;
```

```
SE ERROCODIGO ==0
IMPRIMA ARQUIVO DO TIPO STRING
IMPRIMA ARQUIVO DO TIPO STRING, FILE
IMPRIMA ARQUIVO DO TIPO STRING, “ANALISADO CORRETAMENTE”
    SENÃO
    IMPRIMA ARQUIVO DO TIPO STRING, “ O ARQUIVO”
    IMPRIMA ARQUIVO DO TIPO STRING, FILE
    IMPRIMA ARQUIVO DO TIPO STRING, “CONTÉM ERRO(s)”
    FECHA (ARQUIVO)
```

int verificaLexema(char *caracter, LISTA* LL, int countCaracter) // VERIFICAÇÃO DE LEXEMA , PALAVRA E CHARACTER INTEIRO N

APONTA LOCALIZAÇÃO DE UM REGISTRO NO CASO O ARQUIVO

SE COMPARAR 2 STRING (CHARACTER, "a") ==0

CONCATENAÇÃO DE 2 STRING(PALAVRA, CHARACTER)

IMPRIMA LETRA DO TIPO STRING - PALAVRA , CHARACTER , PALAVRA

SENAO

COMPARE COM OUTROS CHARACTER, PALAVRA, OPERADORES ARITMÉTICOS, FECHAMENTOS DE CHAVES E PARÊNTESES, OPERADORES RELACIONAIS, NÚMEROS, PONTO E VIRGULA, VIRGULA ,DOIS PONTOS

int verificaCadeia(char *palavra, char *caracter, int countCaracter, int countErroCaracter, LISTA *LL) // VERIFICAÇÃO DE CADEIA

APONTA LOCALIZAÇÃO DE UM REGISTRO NO CASO O ARQUIVO COMPARAÇÃO DE STRING DE PALAVRAS

void output() // ARQUIVO DE SAÍDA

IMPRIMA ARQUIVO DE SAÍDA : resultado.txt

TEMPO (5000)

void loading() // CARREGANDO ARQUIVO

IMPRIMA CARREGANDO, LENDO E COMPILANDO O ARQUIVO: STRING, file

TEMPO (5000)

void lerArquivo(LISTA *LL) // LER ARQUIVO

FILE *arq; APONTA LOCALIZAÇÃO DE UM REGISTRO NO CASO O ARQUIVO

char caracter[400]; VETOR DE CHARACTER (40 POSIÇÕES)

gets(file); RETORNA UM CHARACTER LIDO (file)

loading(); CARREGANDO

arq = fopen(file, "r"); ARQUIVO = ABERTURA (file, "r")

strcpy(palavra,""); CÓPIA DE 2 STRING

while (!feof(arq)) // while (DIFERENTE DA VERIFICAÇÃO DE FIM DE ARQUIVO)

fread(&caracter,1,1,arq); LER

countCaracter++; INVREMETA CONTADOR

//printf("\n[%s]",caracter); IMPRIME

//Sleep(10); TEMPO

verificaLexema(caracter,LL,countCaracter); VERIFICA

```
    }  
    printf("\nArquivo lido com sucesso!\n"); IMPRIME
```

```
int main() { // FUNCAO MAIN  
    char arquivo[550]; VETOR DE CARACTER 300 POSIÇÕES  
    inicializarLISTA(&LL); // INICIALIZAR LISTA  
    printf("|===== ANALISADOR LEXICO =====|\n"); IMPRIME  
    printf("|== INFORME O NOME DO ARQUIVO A SER ANALISADO: ==|\n");  
    IMPRIME  
    lerArquivo(&LL); // LER ARQUIVO  
    exibirLISTA(&LL); // EXIBIR ARQUIVO  
    Sleep(5000); TEMPO  
    system("cls"); PAUSA  
    inicio(&LL); // INICIO  
    verificaFechamento(); // VERIFICACAO FECHAMENTO DO ARQUIVO  
    verificaErros(); // VERIFICACAO DE ERROS DO ARQUIVO  
    output(); // SAÍDA DO ARQUIVO  
    return (0); RETORNAR 0
```