**Manual do programador**

O projecto esta dividido em 4 arquivos, onde:

1. Estados.c responsavel por analisar os caracteres lidos ou sequencia de caracteres lidos e valiar de acordo aos automatos criados.
2. Header.h o meu cabeçalho responsavel por definir as funcoes;
3. Main.c responsavel por rodar o projecto e fazer a leitura do arquivo.
4. Tokens.h o arquivo responsavel por guardar o significado de cada token.

Funções encontradas no ficheiro estados.c

1. Função do “analex” tipo “Tipo” responsavel por analisar os caracteres lidos ou sequencia de caracteres lidos e valiar de acordo aos automatos criados;

Funções encontradas no ficheiro main.c

1. Função main do tipo int que é o corpo do projecto.
2. Função zerarVetor que é reposavel por zerar o vector.
3. Função lerFicheiro, que é resposavel por ler o ficheiro caractere a caractere e retorna esse caracatere.
4. Função voltarCaractere, que é responsavel por voltar o ponteiro um bit antes da posicao atual, é usado depois de executado a funcao ler caractere para verificar o carcatere que vem a seguir e poder escolher o estado seguinte e a funcao voltarcaractere retorna o ponteiro na posicao anterior.

**Listas dos tokens utilizados**

#define TK\_DIRE                             "Directiva\_de\_Processamento"

#define TK\_PRV                              "Palavra\_Reservada"

#define TK\_ID                               "identificador"

#define TK\_CM                               "comentario"

#define TK\_TEXT                             "Texto"

#define TK\_AP                               "abre\_parenteses"

#define TK\_FP                               "fecha\_parenteses"

#define TK\_AC                               "abre\_chaves"

#define TK\_FC                               "fecha\_chaves"

#define TK\_INI                              "Inicio\_arquivo"

#define TK\_END                              "final\_arquivo"

#define TK\_PV                               "ponto\_virgula"

#define TK\_VG                               "virgula"

#define TK\_AD                               "adicao"

#define TK\_SUB                              "subtracao"

#define TK\_MUL                              "multiplicacao"

#define TK\_DIV                              "divisao"

#define TK\_MOD                              "modulo"

#define TK\_MENOR                            "menor"

#define TK\_MAIOR                            "maior"

#define TK\_MENORIG                          "menor\_igual"

#define TK\_MAIORIG                          "maior\_igual"

#define TK\_IG                               "igual"

#define TK\_DIF                              "diferente"

#define TK\_AND                              "E\_logico"

#define TK\_OR                               "OU\_logico"

#define TK\_NEG                              "negacao"

#define TK\_DES\_ES                           "Deslocamento a Esquerda"

#define TK\_DES\_D                            "Deslocamento a Direita"

#define TK\_BOU\_EX                           "OPerador Bitwise OU exclusivo"

#define TK\_BE                               "Operador Bitwise E"

#define TK\_BOU                              "Operador Bitwise OU"

#define TK\_BCOMP                            "Operador Complemento Bitwise"

#define TK\_INT                              "int"

#define TK\_FLOAT                            "float"

#define TK\_CHAR                             "char"

#define TK\_DOUBLE                           "double"

#define TK\_INCREMENT                        "Incremento"

#define TK\_DECREMENT                        "Decremento"

#define TK\_ATR                              "Atribuicao Simples"

#define TK\_ATR\_AD                           "Adicao e Atribuicao"

#define TK\_ATR\_SUB                          "Subtra��o e Atribuicao"

#define TK\_ATR\_MUL                          "Multiplica��o e Atribuicao"

#define TK\_ATR\_DIV                          "DIV e Atribuicao"

#define TK\_ATR\_MOD                          "M�dulo e Atribuicao"

#define TK\_ATR\_DES\_ES                       "Deslocamento a Esquerda e Atribuicao"

#define TK\_ATR\_DES\_D                        "Deslocamento a Direita e Atribuicao"

#define TK\_ATR\_BOU\_EX                           "OPerador Bitwise OU exclusivo e atribuicao"

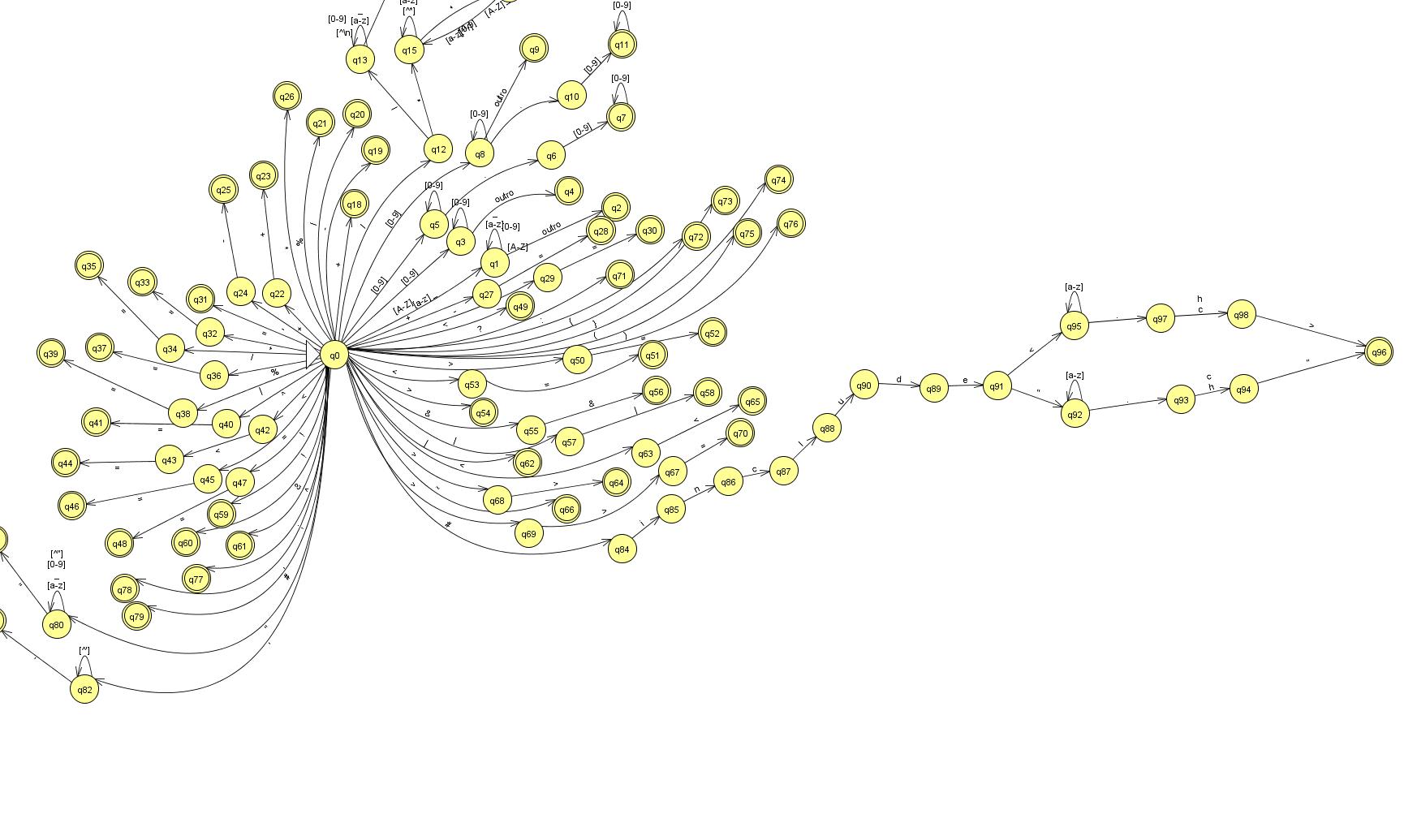
#define TK\_ATR\_BE                               "Operador Bitwise E e Atribuicao"

#define TK\_ATR\_BOU                              "Operador Bitwise OU e Atribuicao"

#define TK\_ADDRESS                          "Endereço"

#define TK\_POINTER                      "Ponteiro"

**Autómato Finito Determinístico**



**Expressão Regular**

([a-z]+[A-Z]+\_)([A-Z]+\_+[0-9]+[a-z])\*outro+[0-9]([0-9])\*outro+[0-9]([0-9])\*.[0-9]([0-9])\*+[0-9]([0-9])\*outro+[0-9]([0-9])\*.[0-9]([0-9])\*+//([A-Z]+\_+[0-9]+[a-z]+[^\n])\*\n+/\*([A-Z]+[^\*]+\_+[0-9]+[a-z])\*\*(\*+(\_+[^\*/]+[A-Z]+[a-z]+[0-9])([A-Z]+[^\*]+\_+[0-9]+[a-z])\*\*)\*/+++-+/+%+(+)(+)+--+\*+(+)=+-=+=+\*=+/=+%=+|=+^=+<<=+==+!=+<+>=+<=+>+&&+||+!+&+^+|+<<+~+>>+>>=+?+:+{+}+(+)+;+,+#+"([^"]+\_+[0-9]+[a-z])\*"+'([^'])\*'+#include"([a-z])\*.(c+h)"+#include<([a-z])\*.(c+h)>

**Pseudo-Código de cada função (Ficheiro estados.c)**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include "token.h"

#include "header.h"

typedef struct *Tipo*

{

    char \*token;

    char \*lexema;

    int linha;

} *Tipo*;

int cont = 0;

int contLinha = 1;

*Tipo* analex(*FILE* \**ficheiro*)

{

*Tipo* k;

    int estado = 0;

    char caractere;

    zerarVetor();

    while (!feof(*ficheiro*))

    {

        switch (estado)

        {

        case 0:

            caractere = lerCaractere();

            if (caractere == '#')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 84;

            }

            else if ((caractere >= 'A' && caractere <= 'Z') || (caractere >= 'a' && caractere <= 'z') || caractere == '\_')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 1;

            }

            else if (caractere >= '0' && caractere <= '9')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 3;

            }

            else if (caractere == ';')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 77;

            }

            else if (caractere == '=')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 45;

                    VoltarCaractere();

                }

                else

                {

                    estado = 31;

                }

            }

            else if (caractere == '+')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '+')

                {

                    estado = 22;

                    VoltarCaractere();

                }

                else if (prox == '=')

                {

                    estado = 27;

                    VoltarCaractere();

                }

                else

                {

                    estado = 18;

                }

            }

            else if (caractere == '-')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '-')

                {

                    estado = 24;

                    VoltarCaractere();

                }

                else if (prox == '=')

                {

                    estado = 29;

                    VoltarCaractere();

                }

                else

                {

                    estado = 19;

                }

            }

            else if (caractere == '\*')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 34;

                    VoltarCaractere();

                }

                else

                    estado = 26;

            }

            else if (caractere == '/')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '/')

                {

                    estado = 12;

                    VoltarCaractere();

                }

                else if (prox == '\*')

                {

                    estado = 15;

                    VoltarCaractere();

                }

                else if (prox == '=')

                {

                    estado = 36;

                    VoltarCaractere();

                }

                else

                {

                    estado = 20;

                }

            }

            else if (caractere == '%')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 36;

                    VoltarCaractere();

                }

                else

                {

                    estado = 21;

                }

            }

            else if (caractere == '<')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 53;

                    VoltarCaractere();

                }

                else if (prox == '<')

                {

                    estado = 63;

                    VoltarCaractere();

                }

                else

                {

                    estado = 49;

                }

            }

            else if (caractere == '>')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 50;

                    VoltarCaractere();

                }

                else if (prox == '>')

                {

                    estado = 68;

                    VoltarCaractere();

                }

                else

                {

                    estado = 54;

                }

            }

            else if (caractere == '(')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 75;

            }

            else if (caractere == ')')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 76;

            }

            else if (caractere == '{')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 73;

            }

            else if (caractere == '}')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 74;

            }

            else if (caractere == ',')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 78;

            }

            else if (caractere == '!')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 59;

            }

            else if (caractere == '?')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 71;

            }

            else if (caractere == '&')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                /\*if(prox == '='){

                }else \*/

                if (prox == '&')

                {

                    estado = 55;

                    VoltarCaractere();

                }

                else

                {

                    estado = 60;

                }

            }

            else if (caractere == '~')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 66;

            }

            else if (caractere == '|')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '|')

                {

                    estado = 57;

                    VoltarCaractere();

                }

                else

                {

                    estado = 62;

                }

            }

            else if (caractere == '^')

            {

                vetor[cont] = caractere;

                cont++;

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 40;

                    VoltarCaractere();

                }

                else

                {

                    estado = 61;

                }

            }

            else if (caractere == '.')

            {

            }

            else if (caractere == '"')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 80;

            }

            break;

        case 1:

            caractere = lerCaractere();

            while ((caractere >= 'A' && caractere <= 'Z') || (caractere >= 'a' && caractere <= 'z') || caractere == '\_' || (caractere >= '0' && caractere <= '9'))

            {

                vetor[cont] = caractere;

                cont++;

                caractere = lerCaractere();

            }

            estado = 2;

            break;

        case 2:

            k.lexema = vetor;

            char \*palavras\_reservadas[] = {"auto", "break", "case", "char", "const", "continue", "default", "do","double", "else", "enum", "extern", "float", "for", "goto", "if","int", "long", "register", "return", "short", "signed", "sizeof", "static","struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"};

            for (int i = 0; i < 32; i++)

            {

                if (strcmp(palavras\_reservadas[i], k.lexema) == 0)

                {

                    k.token = TK\_PRV;

                    k.linha = contLinha++;

                    return k;

                }

            }

            k.token = TK\_ID;

            k.linha = contLinha++;

            return k;

            break;

        case 3:

            caractere = lerCaractere();

            while (caractere >= '0' && caractere <= '9')

            {

                vetor[cont] = caractere;

                cont++;

                caractere = lerCaractere();

            }

            if (caractere == '.')

            {

                vetor[cont] = caractere;

                cont++;

                estado = 6;

            }

            else

                estado = 4;

            break;

        case 4:

            k.lexema = vetor;

            k.token = TK\_INT;

            k.linha = contLinha++;

            return k;

            break;

        case 5:

            k.lexema = vetor;

            k.token = TK\_CM;

            k.linha = contLinha++;

            return k;

            break;

        case 6:

            caractere = lerCaractere();

            while (caractere >= '0' && caractere <= '9')

            {

                vetor[cont] = caractere;

                cont++;

                caractere = lerCaractere();

            }

            estado = 7;

            break;

        case 7:

            k.lexema = vetor;

            k.token = TK\_FLOAT;

            k.linha = contLinha++;

            return k;

            break;

        case 8:

            break;

        case 9:

            break;

        case 10:

            break;

        case 11:

            break;

        case 12:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere = '/')

            {

                estado = 13;

            }

            break;

        case 13:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (isascii(caractere) && caractere != '\n')

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            estado = 14;

            break;

        case 14:

            k.lexema = vetor;

            k.token = TK\_CM;

            k.linha = contLinha++;

            return k;

            break;

        case 15:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (isascii(caractere) && caractere != '\*')

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            estado = 16;

            break;

        case 16:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (!isascii(caractere))

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            if (caractere == '/')

            {

                estado = 17;

            }

            else

            {

                estado = 15;

            }

            break;

        case 17:

            k.lexema = vetor;

            k.token = TK\_CM;

            k.linha = contLinha++;

            return k;

            break;

        case 18:

            k.lexema = vetor;

            k.token = TK\_AD;

            k.linha = contLinha++;

            return k;

            break;

        case 19:

            k.lexema = vetor;

            k.token = TK\_SUB;

            k.linha = contLinha++;

            return k;

            break;

        case 20:

            k.lexema = vetor;

            k.token = TK\_DIV;

            k.linha = contLinha++;

            return k;

            break;

        case 21:

            k.lexema = vetor;

            k.token = TK\_MOD;

            k.linha = contLinha++;

            return k;

            break;

        case 22:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '+')

                estado = 23;

            break;

        case 23:

            k.lexema = vetor;

            k.token = TK\_INCREMENT;

            k.linha = contLinha++;

            return k;

            break;

        case 24:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '-')

                estado = 25;

            break;

        case 25:

            k.lexema = vetor;

            k.token = TK\_DECREMENT;

            k.linha = contLinha++;

            return k;

            break;

        case 26:

            k.lexema = vetor;

            k.token = TK\_MUL;

            k.linha = contLinha++;

            return k;

            break;

        case 27:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 28;

            break;

        case 28:

            k.lexema = vetor;

            k.token = TK\_ATR\_AD;

            k.linha = contLinha++;

            return k;

            break;

        case 29:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 30;

            break;

        case 30:

            k.lexema = vetor;

            k.token = TK\_ATR\_SUB;

            k.linha = contLinha++;

            return k;

            break;

        case 31:

            k.lexema = vetor;

            k.token = TK\_ATR;

            k.linha = contLinha++;

            return k;

            break;

        case 32:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 35;

            break;

        case 33:

            k.lexema = vetor;

            k.token = TK\_IG;

            k.linha = contLinha++;

            return k;

            break;

        case 34:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 35;

            break;

        case 35:

            k.lexema = vetor;

            k.token = TK\_ATR\_MUL;

            k.linha = contLinha++;

            return k;

            break;

        case 36:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 37;

            break;

        case 37:

            k.lexema = vetor;

            k.token = TK\_ATR\_MOD;

            k.linha = contLinha++;

            return k;

            break;

        case 38:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 39;

            break;

        case 39:

            k.lexema = vetor;

            k.token = TK\_ATR\_BOU;

            k.linha = contLinha++;

            return k;

            break;

        case 40:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 41;

            break;

        case 41:

            k.lexema = vetor;

            k.token = TK\_ATR\_BOU\_EX;

            k.linha = contLinha++;

            return k;

            break;

        case 42:

            break;

        case 43:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 44;

            break;

        case 44:

            k.lexema = vetor;

            k.token = TK\_ATR\_DES\_ES;

            k.linha = contLinha++;

            return k;

            break;

        case 45:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 46;

            break;

        case 46:

            k.lexema = vetor;

            k.token = TK\_IG;

            k.linha = contLinha++;

            return k;

            break;

        case 47:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 48;

            break;

        case 48:

            k.lexema = vetor;

            k.token = TK\_DIF;

            k.linha = contLinha++;

            return k;

            break;

        case 49:

            k.lexema = vetor;

            k.token = TK\_MENOR;

            k.linha = contLinha++;

            return k;

            break;

        case 50:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 52;

            break;

        case 51:

            k.lexema = vetor;

            k.token = TK\_MENORIG;

            k.linha = contLinha++;

            return k;

            break;

        case 52:

            k.lexema = vetor;

            k.token = TK\_MAIORIG;

            k.linha = contLinha++;

            return k;

            break;

        case 53:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 51;

            break;

        case 54:

            k.lexema = vetor;

            k.token = TK\_MAIOR;

            k.linha = contLinha++;

            return k;

            break;

        case 55:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '&')

                estado = 56;

            break;

        case 56:

            k.lexema = vetor;

            k.token = TK\_AND;

            k.linha = contLinha++;

            return k;

            break;

        case 57:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '|')

                estado = 58;

            break;

        case 58:

            k.lexema = vetor;

            k.token = TK\_OR;

            k.linha = contLinha++;

            return k;

            break;

        case 59:

            k.lexema = vetor;

            k.token = TK\_NEG;

            k.linha = contLinha++;

            return k;

            break;

        case 60:

            k.lexema = vetor;

            k.token = TK\_ADDRESS;

            k.linha = contLinha++;

            return k;

            break;

        case 61:

            k.lexema = vetor;

            k.token = TK\_BOU\_EX;

            k.linha = contLinha++;

            return k;

            break;

        case 62:

            k.lexema = vetor;

            k.token = TK\_BOU;

            k.linha = contLinha++;

            return k;

            break;

        case 63:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '<')

            {

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 43;

                    VoltarCaractere();

                }

                else

                    estado = 65;

            }

            break;

        case 64:

            k.lexema = vetor;

            k.token = TK\_DES\_D;

            k.linha = contLinha++;

            return k;

            break;

        case 65:

            k.lexema = vetor;

            k.token = TK\_DES\_ES;

            k.linha = contLinha++;

            return k;

            break;

        case 66:

            k.lexema = vetor;

            k.token = TK\_BCOMP;

            k.linha = contLinha++;

            return k;

            break;

        case 67:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '=')

                estado = 70;

            break;

        case 68:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '>')

            {

                char prox = lerCaractere();

                if (prox == '=')

                {

                    estado = 67;

                    VoltarCaractere();

                }

                else

                    estado = 64;

            }

            break;

        case 69:

            break;

        case 70:

            k.lexema = vetor;

            k.token = TK\_ATR\_DES\_D;

            k.linha = contLinha++;

            return k;

            break;

        case 71:

            break;

        case 72:

            break;

        case 73:

            k.lexema = vetor;

            k.token = TK\_AC;

            k.linha = contLinha++;

            return k;

            break;

        case 74:

            k.lexema = vetor;

            k.token = TK\_FC;

            k.linha = contLinha++;

            return k;

            break;

        case 75:

            k.lexema = vetor;

            k.token = TK\_AP;

            k.linha = contLinha++;

            return k;

            break;

        case 76:

            k.lexema = vetor;

            k.token = TK\_FP;

            k.linha = contLinha++;

            return k;

            break;

        case 77:

            k.lexema = vetor;

            k.token = TK\_PV;

            k.linha = contLinha++;

            return k;

            break;

        case 78:

            k.lexema = vetor;

            k.token = TK\_VG;

            k.linha = contLinha++;

            return k;

            break;

        case 79:

            break;

        case 80:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (caractere != '"')

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            estado = 81;

            break;

        case 81:

            k.lexema = vetor;

            k.token = TK\_TEXT;

            k.linha = contLinha++;

            return k;

            break;

        case 82:

            caractere = lerCaractere();

            if (isascii(caractere))

            {

                vetor[cont] = caractere;

                cont++;

                estado = 83;

            }

            break;

        case 83:

            k.lexema = vetor;

            k.token = TK\_CHAR;

            k.linha = contLinha++;

            return k;

            break;

        case 84:

            caractere = lerCaractere();

            while (strstr(vetor, "#include") == NULL)

            {

                vetor[cont] = caractere;

                cont++;

                estado = 91;

                caractere = lerCaractere();

            }

            break;

        case 91:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '"')

            {

                estado = 92;

            }

            else if (caractere == '<')

            {

                estado = 95;

            }

            break;

        case 92:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (caractere >= 'a' && caractere <= 'z')

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            if (caractere == '.')

            {

                estado = 93;

            }

            break;

        case 93:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == 'h' || caractere == 'c')

            {

                estado = 94;

            }

            break;

        case 94:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '"')

            {

                estado = 96;

            }

            break;

        case 95:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            while (caractere >= 'a' && caractere <= 'z')

            {

                caractere = lerCaractere();

                vetor[cont] = caractere;

                cont++;

            }

            if (caractere == '.')

            {

                estado = 97;

            }

            break;

        case 96:

            k.lexema = vetor;

            k.token = TK\_DIRE;

            k.linha = contLinha++;

            return k;

            break;

        case 97:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == 'h' || caractere == 'c')

            {

                estado = 98;

            }

            break;

        case 98:

            caractere = lerCaractere();

            vetor[cont] = caractere;

            cont++;

            if (caractere == '>')

            {

                estado = 96;

            }

            break;

        }

    }

    k.lexema = "";

    k.token = TK\_END;

    k.linha = contLinha++;

    return k;

}

**Pseudo-Código de cada função (Ficheiro main.c)**

#include "estados.c"

char lerCaractere()

{

    char caractere;

    fread(&caractere, 1, 1, ficheiro);

    return caractere;

};

void VoltarCaractere()

{

    fseek(ficheiro, -1, SEEK\_CUR); // posiciona o cursor uma posi??o anterior

};

void zerarVetor()

{

    int conte;

    for (conte = 0; conte < MAX; conte++)

    {

        vetor[conte] = '\0';

    }

    cont = 0;

}

int main()

{

    ficheiro = fopen("arquivo.txt", "rb");

*Tipo* t;

    printf("LEXAMA\t\t\t\t\tTOKEN\t\t\t\t LINHA\n");

    do

    {

        t = analex(ficheiro);

        printf("%s\t\t\t\t\t %s\t\t\t\t%d\n", t.lexema, t.token, t.linha);

        // zerarVetor();

    } while (t.token != TK\_END);

    return 0;

}

**Pseudo-Código de cada função (Ficheiro header.h)**

#define MAX 1000

*FILE* \*ficheiro;

char vetor[MAX];

typedef struct *Tipo* *Tipo*;

*Tipo* analex(*FILE* \**ficheiro*);

char lerCaractere();

void VoltarCaractere();

void zerarVetor();