INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

TEORÍA DE LA COMPUTACIÓN

ING. EN SISTEMAS COMPUTACIONALES

Práctica 6: Analizador de Texto

Integrantes:

Espinoza León Jaqueline Viridiana

García Zacarías Angel Emmanuel

Hurtado Morales Emiliano

Profesora: Luz María Sánchez García

Fecha de entrega: 01/06/22

Ciclo Escolar: 2022 - 1

Índice

1. Planteamiento del problema

Se diseñará un programa en lenguaje C, capaz de leer un archivo y detectar lo siguiente:

- Letras
    - Minúsculas
    - Mayúsculas
- Dígitos (0 - 9)
- Signos de puntuación
- Operadores
    - Aritméticos
    - Lógicos
- Paréntesis
- Corchetes
- Llaves
- Caracteres

Después de eso, indicará la frecuencia con la que se repite cada uno y mostrará el porcentaje que ocupa en el archivo.

2. Actividades

1. Se diseñó un programa en lenguaje C, capaz de leer un archivo y detectar lo siguiente:

- Letras
    - Minúsculas
    - Mayúsculas
- Dígitos (0 - 9)
- Signos de puntuación
- Operadores
    - Aritméticos
    - Lógicos
- Paréntesis
- Corchetes
- Llaves
- Caracteres

Después de eso, indicó la frecuencia con la que se repite cada uno y mostró el porcentaje que ocupa en el archivo.

2. El programa es capaz de leer un archivo fuente, además de detectar los tokens antes mencionados, y mostrar su frecuencia y su porcentaje.

3. Mostramos los resultados obtenidos.

```
( se encuentra 128 veces, es un 1.1814 %
) se encuentra 128 veces, es un 1.1814 %
[ se encuentra 202 veces, es un 1.8643 %
] se encuentra 202 veces, es un 1.8643 %
{ se encuentra 13 veces, es un 0.1200 %
} se encuentra 13 veces, es un 0.1200 %
< se encuentra 14 veces, es un 0.1292 %
> se encuentra 9 veces, es un 0.0831 %
& se encuentra 3 veces, es un 0.0277 %
| se encuentra 2 veces, es un 0.0185 %
+ se encuentra 201 veces, es un 1.8551 %
- se encuentra 3 veces, es un 0.0277 %
* se encuentra 983 veces, es un 9.0725 %
/ se encuentra 364 veces, es un 3.3595 %
% se encuentra 8 veces, es un 0.0738 %
= se encuentra 208 veces, es un 1.9197 %
, se encuentra 139 veces, es un 1.2829 %
; se encuentra 147 veces, es un 1.3567 %
: se encuentra 20 veces, es un 0.1846 %
. se encuentra 33 veces, es un 0.3046 %
# se encuentra 11 veces, es un 0.1015 %
^ se encuentra 2 veces, es un 0.0185 %
~ se encuentra 2 veces, es un 0.0185 %
" se encuentra 11 veces, es un 0.1015 %
? se encuentra 2 veces, es un 0.0185 %
¿ se encuentra 0 veces, es un 0.0000 %
! se encuentra 4 veces, es un 0.0369 %
¡ se encuentra 0 veces, es un 0.0000 %
_ se encuentra 13 veces, es un 0.1200 %
$ se encuentra 2 veces, es un 0.0185 %
```

```
y se encuentra 22 veces, es un 0.2030 %
z se encuentra 20 veces, es un 0.1846 %
A se encuentra 36 veces, es un 0.3323 %
B se encuentra 5 veces, es un 0.0461 %
C se encuentra 21 veces, es un 0.1938 %
D se encuentra 19 veces, es un 0.1754 %
N se encuentra 18 veces, es un 0.1661 %
O se encuentra 15 veces, es un 0.1384 %
P se encuentra 14 veces, es un 0.1292 %
Q se encuentra 2 veces, es un 0.0185 %
R se encuentra 19 veces, es un 0.1754 %
S se encuentra 19 veces, es un 0.1754 %
T se encuentra 14 veces, es un 0.1292 %
U se encuentra 13 veces, es un 0.1200 %
V se encuentra 6 veces, es un 0.0554 %
W se encuentra 2 veces, es un 0.0185 %
X se encuentra 6 veces, es un 0.0554 %
Y se encuentra 2 veces, es un 0.0185 %
Z se encuentra 3 veces, es un 0.0277 %
\ se encuentra 1 veces, es un 0.0092 %
' se encuentra 358 veces, es un 3.3041 %
```

```
0 se encuentra 28 veces, es un 0.2584 %
1 se encuentra 31 veces, es un 0.2861 %
2 se encuentra 28 veces, es un 0.2584 %
3 se encuentra 27 veces, es un 0.2492 %
4 se encuentra 26 veces, es un 0.2400 %
5 se encuentra 21 veces, es un 0.1938 %
6 se encuentra 23 veces, es un 0.2123 %
7 se encuentra 23 veces, es un 0.2123 %
8 se encuentra 23 veces, es un 0.2123 %
9 se encuentra 22 veces, es un 0.2030 %
a se encuentra 428 veces, es un 3.9502 %
b se encuentra 54 veces, es un 0.4984 %
c se encuentra 335 veces, es un 3.0918 %
d se encuentra 89 veces, es un 0.8214 %
e se encuentra 755 veces, es un 6.9682 %
f se encuentra 243 veces, es un 2.2427 %
g se encuentra 121 veces, es un 1.1168 %
h se encuentra 54 veces, es un 0.4984 %
i se encuentra 498 veces, es un 4.5962 %
j se encuentra 11 veces, es un 0.1015 %
k se encuentra 3 veces, es un 0.0277 %
l se encuentra 305 veces, es un 2.8150 %
m se encuentra 55 veces, es un 0.5076 %
n se encuentra 280 veces, es un 2.5842 %
o se encuentra 281 veces, es un 2.5934 %
p se encuentra 44 veces, es un 0.4061 %
q se encuentra 11 veces, es un 0.1015 %
r se encuentra 502 veces, es un 4.6331 %
s se encuentra 351 veces, es un 3.2395 %
t se encuentra 176 veces, es un 1.6244 %
u se encuentra 190 veces, es un 1.7536 %
v se encuentra 39 veces, es un 0.3599 %
w se encuentra 4 veces, es un 0.0369 %
x se encuentra 10 veces, es un 0.0923 %
```

4. Finalmente responda a las siguientes preguntas

a) **¿Es fácil o difícil trabajar con archivos de texto? ¿Por qué?**

A nuestra consideración es fácil si el programa que se intenta diseñar es algo sencillo, pero se eleva la dificultad si usamos archivos grandes o tenemos un programa más complejos

b) **¿Qué dificultades se presentaron al realizar la práctica? ¿Se pudieron resolver?**

No encontramos ninguna dificultad.

c) **¿Qué aplicaciones puede tener la práctica?**

Podemos usarla en caso de que busquemos conocer que cantidad del archivo es algún carácter o número, o en caso de que busquemos llevar un control de determinados caracteres.

d) **¿Qué recomendaciones daría a los nuevos equipos?**

Consideramos que si se tienen bien diseñadas las prácticas anteriores, es posible que esta sea demasiado sencilla. Por lo tanto, el consejo sería que diseñen de manera correcta todo lo anterior.

e) **Investiga algoritmos que puedan resolver el mismo problema.**

Uno de los algoritmos que pueden aportar a la identificación de apariciones de carácteres es el Algoritmo de Huffman, el cual consiste en la creación de un árbol binario que tiene cada uno de los caracteres por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado. El proceso que toma es el siguiente:

1. Se crean varios árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.
2. Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.
3. Se repite el paso 2 hasta que sólo quede un árbol

Aunque no se emplee la parte del árbol para las apariciones, la parte más importante del algoritmo de Huffman para el contexto dado, es la creación de la lista, donde cada nodo guarda el carácter y su frecuencia. Es posible implementar una lista doblemente enlazada como estructura de datos, donde se declare en cada nodo un char, un int y los apuntadores para atrás y adelante.

3. Pruebas

1.- Programación de detección de los caracteres requeridos.

a) Detección de corchetes, llaves, paréntesis y paréntesis triangulares.

```
if(arreglo[i] == '(')
            frecuencias[0]++;
        else if(arreglo[i] == ')')
            frecuencias[1]++;
        else if(arreglo[i] == '[')
            frecuencias[2]++;
        else if(arreglo[i] == ']')
            frecuencias[3]++;
        else if(arreglo[i] == '{')
            frecuencias[4]++;
        else if(arreglo[i] == '}')
            frecuencias[5]++;
        else if(arreglo[i] == '<')
            frecuencias[6]++;
        else if(arreglo[i] == '>')
            frecuencias[7]++;
```

b) Detección de operadores aritméticos y lógicos.

```
else if(arreglo[i] == '&')
    frecuencias[8]++;
else if(arreglo[i] == '|')
    frecuencias[9]++;
else if(arreglo[i] == '+')
    frecuencias[10]++;
else if(arreglo[i] == '-')
    frecuencias[11]++;
else if(arreglo[i] == '*')
    frecuencias[12]++;
else if(arreglo[i] == '/')
    frecuencias[13]++;
else if(arreglo[i] == '%')
    frecuencias[14]++;
else if(arreglo[i] == '=')
    frecuencias[15]++;
```

c) Detección de caracteres

```
else if(arreglo[i] == ',')
    frecuencias[16]++;
else if(arreglo[i] == ';')
    frecuencias[17]++;
else if(arreglo[i] == ':')
    frecuencias[18]++;
else if(arreglo[i] == '.')
    frecuencias[19]++;
else if(arreglo[i] == '#')
    frecuencias[20]++;
else if(arreglo[i] == '^')
    frecuencias[21]++;
else if(arreglo[i] == '~')
    frecuencias[22]++;
else if(arreglo[i] == '"')
    frecuencias[23]++;
else if(arreglo[i] == '?')
    frecuencias[24]++;
else if(arreglo[i] == 168)
    frecuencias[25]++;
else if(arreglo[i] == '!')
    frecuencias[26]++;
else if(arreglo[i] == 173)
    frecuencias[27]++;
else if(arreglo[i] == '_')
    frecuencias[28]++;
else if(arreglo[i] == '$')
    frecuencias[29]++;
else if(arreglo[i] == 92)
    frecuencias[92]++;
else if(arreglo[i] == 39)
    frecuencias[93]++;
```

d) Detección de números

```
else if(arreglo[i] == '0')
    frecuencias[30]++;
else if(arreglo[i] == '1')
    frecuencias[31]++;
else if(arreglo[i] == '2')
    frecuencias[32]++;
else if(arreglo[i] == '3')
    frecuencias[33]++;
else if(arreglo[i] == '4')
    frecuencias[34]++;
else if(arreglo[i] == '5')
    frecuencias[35]++;
else if(arreglo[i] == '6')
    frecuencias[36]++;
else if(arreglo[i] == '7')
    frecuencias[37]++;
else if(arreglo[i] == '8')
    frecuencias[38]++;
else if(arreglo[i] == '9')
    frecuencias[39]++;
```

e) Detección de minúsculas

```
else if(arreglo[i] == 'a')
    frecuencias[40]++;
else if(arreglo[i] == 'b')
    frecuencias[41]++;
else if(arreglo[i] == 'c')
    frecuencias[42]++;
else if(arreglo[i] == 'd')
    frecuencias[43]++;
else if(arreglo[i] == 'e')
    frecuencias[44]++;
else if(arreglo[i] == 'f')
    frecuencias[45]++;
else if(arreglo[i] == 'g')
    frecuencias[46]++;
else if(arreglo[i] == 'h')
    frecuencias[47]++;
else if(arreglo[i] == 'i')
    frecuencias[48]++;
else if(arreglo[i] == 'j')
    frecuencias[49]++;
else if(arreglo[i] == 'k')
    frecuencias[50]++;
else if(arreglo[i] == 'l')
    frecuencias[51]++;
else if(arreglo[i] == 'm')
    frecuencias[52]++;
```

```
else if(arreglo[i] == 'n')
    frecuencias[53]++;
else if(arreglo[i] == 'o')
    frecuencias[54]++;
else if(arreglo[i] == 'p')
    frecuencias[55]++;
else if(arreglo[i] == 'q')
    frecuencias[56]++;
else if(arreglo[i] == 'r')
    frecuencias[57]++;
else if(arreglo[i] == 's')
    frecuencias[58]++;
else if(arreglo[i] == 't')
    frecuencias[59]++;
else if(arreglo[i] == 'u')
    frecuencias[60]++;
else if(arreglo[i] == 'v')
    frecuencias[61]++;
else if(arreglo[i] == 'w')
    frecuencias[62]++;
else if(arreglo[i] == 'x')
    frecuencias[63]++;
else if(arreglo[i] == 'y')
    frecuencias[64]++;
else if(arreglo[i] == 'z')
    frecuencias[65]++;
```

f) Detección de mayúsculas

```
else if(arreglo[i] == 'A')
    frecuencias[66]++;
else if(arreglo[i] == 'B')
    frecuencias[67]++;
else if(arreglo[i] == 'C')
    frecuencias[68]++;
else if(arreglo[i] == 'D')
    frecuencias[69]++;
else if(arreglo[i] == 'E')
    frecuencias[70]++;
else if(arreglo[i] == 'F')
    frecuencias[71]++;
else if(arreglo[i] == 'G')
    frecuencias[72]++;
else if(arreglo[i] == 'H')
    frecuencias[73]++;
else if(arreglo[i] == 'I')
    frecuencias[74]++;
else if(arreglo[i] == 'J')
    frecuencias[75]++;
else if(arreglo[i] == 'K')
    frecuencias[76]++;
else if(arreglo[i] == 'L')
    frecuencias[77]++;
else if(arreglo[i] == 'M')
    frecuencias[78]++;
else if(arreglo[i] == 'N')
    frecuencias[79]++;
else if(arreglo[i] == 'O')
    frecuencias[80]++;
else if(arreglo[i] == 'P')
    frecuencias[81]++;
else if(arreglo[i] == 'Q')
    frecuencias[82]++;
else if(arreglo[i] == 'R')
    frecuencias[83]++;
else if(arreglo[i] == 'S')
    frecuencias[84]++;
else if(arreglo[i] == 'T')
    frecuencias[85]++;
else if(arreglo[i] == 'U')
    frecuencias[86]++;
else if(arreglo[i] == 'V')
    frecuencias[87]++;
else if(arreglo[i] == 'W')
    frecuencias[88]++;
else if(arreglo[i] == 'X')
    frecuencias[89]++;
else if(arreglo[i] == 'Y')
    frecuencias[90]++;
else if(arreglo[i] == 'Z')
    frecuencias[91]++;
```

g) Mostrar frecuencia y porcentaje

```
float obtenerPorcentaje(float num, struct stat sb){
    return (num * 100 /sb.st_size);
}

    for(i = 0; i<94; i++){
        printf("%c se encuentra %i veces, es un %0.4f %% \n", dato[i],
               frecuencias[i], obtenerPorcentaje((float)frecuencias[i],sb));
    }
```

2.- Lectura de archivo fuente

```
void entradaDatos(char *nombreArchivo, char *arreglo, struct stat sb){
    char bufer[1];
    FILE *archivo;
    size_t bytesLeidos;
    int i;

    i = 0;

    archivo = fopen(nombreArchivo, "rb"); // Abrir en modo read binario
    // Si es NULL, entonces no existe, o no se pudo abrir
    if (!archivo) {
        printf("¡No se pudo abrir el archivo %s!", nombreArchivo);
    }

    // Mientras no alcancemos el EndOfLine del archivo...
    while (i<sb.st_size) {
        // Leer dentro del búfer; fread regresa el número de bytes leídos
        bytesLeidos = fread(bufer, sizeof(char), sizeof(bufer), archivo);
        arreglo[i] = bufer[0];
        i++;

    }
    // Al final, se cierra el archivo
    fclose(archivo);
}
```

Para ejecutar el programa, se debe escribir el nombre del archivo a leer al lado del .exe y este debe estar en el mismo directorio.

3.- Salida en pantalla de los resultados

```
( se encuentra 128 veces, es un 1.1814 %
) se encuentra 128 veces, es un 1.1814 %
[ se encuentra 202 veces, es un 1.8643 %
] se encuentra 202 veces, es un 1.8643 %
{ se encuentra 13 veces, es un 0.1200 %
} se encuentra 13 veces, es un 0.1200 %
< se encuentra 14 veces, es un 0.1292 %
> se encuentra 9 veces, es un 0.0831 %
& se encuentra 3 veces, es un 0.0277 %
| se encuentra 2 veces, es un 0.0185 %
+ se encuentra 201 veces, es un 1.8551 %
- se encuentra 3 veces, es un 0.0277 %
* se encuentra 983 veces, es un 9.0725 %
/ se encuentra 364 veces, es un 3.3595 %
% se encuentra 8 veces, es un 0.0738 %
= se encuentra 208 veces, es un 1.9197 %
, se encuentra 139 veces, es un 1.2829 %
; se encuentra 147 veces, es un 1.3567 %
: se encuentra 20 veces, es un 0.1846 %
. se encuentra 33 veces, es un 0.3046 %
# se encuentra 11 veces, es un 0.1015 %
^ se encuentra 2 veces, es un 0.0185 %
~ se encuentra 2 veces, es un 0.0185 %
" se encuentra 11 veces, es un 0.1015 %
? se encuentra 2 veces, es un 0.0185 %
¿ se encuentra 0 veces, es un 0.0000 %
! se encuentra 4 veces, es un 0.0369 %
¡ se encuentra 0 veces, es un 0.0000 %
_ se encuentra 13 veces, es un 0.1200 %
$ se encuentra 2 veces, es un 0.0185 %
```

```
y se encuentra 22 veces, es un 0.2030 %
z se encuentra 20 veces, es un 0.1846 %
A se encuentra 36 veces, es un 0.3323 %
B se encuentra 5 veces, es un 0.0461 %
C se encuentra 21 veces, es un 0.1938 %
D se encuentra 19 veces, es un 0.1754 %
N se encuentra 18 veces, es un 0.1661 %
O se encuentra 15 veces, es un 0.1384 %
P se encuentra 14 veces, es un 0.1292 %
Q se encuentra 2 veces, es un 0.0185 %
R se encuentra 19 veces, es un 0.1754 %
S se encuentra 19 veces, es un 0.1754 %
T se encuentra 14 veces, es un 0.1292 %
U se encuentra 13 veces, es un 0.1200 %
V se encuentra 6 veces, es un 0.0554 %
W se encuentra 2 veces, es un 0.0185 %
X se encuentra 6 veces, es un 0.0554 %
Y se encuentra 2 veces, es un 0.0185 %
Z se encuentra 3 veces, es un 0.0277 %
\ se encuentra 1 veces, es un 0.0092 %
' se encuentra 358 veces, es un 3.3041 %
```

```
0 se encuentra 28 veces, es un 0.2584 %
1 se encuentra 31 veces, es un 0.2861 %
2 se encuentra 28 veces, es un 0.2584 %
3 se encuentra 27 veces, es un 0.2492 %
4 se encuentra 26 veces, es un 0.2400 %
5 se encuentra 21 veces, es un 0.1938 %
6 se encuentra 23 veces, es un 0.2123 %
7 se encuentra 23 veces, es un 0.2123 %
8 se encuentra 23 veces, es un 0.2123 %
9 se encuentra 22 veces, es un 0.2030 %
a se encuentra 428 veces, es un 3.9502 %
b se encuentra 54 veces, es un 0.4984 %
c se encuentra 335 veces, es un 3.0918 %
d se encuentra 89 veces, es un 0.8214 %
e se encuentra 755 veces, es un 6.9682 %
f se encuentra 243 veces, es un 2.2427 %
g se encuentra 121 veces, es un 1.1168 %
h se encuentra 54 veces, es un 0.4984 %
i se encuentra 498 veces, es un 4.5962 %
j se encuentra 11 veces, es un 0.1015 %
k se encuentra 3 veces, es un 0.0277 %
l se encuentra 305 veces, es un 2.8150 %
m se encuentra 55 veces, es un 0.5076 %
n se encuentra 280 veces, es un 2.5842 %
o se encuentra 281 veces, es un 2.5934 %
p se encuentra 44 veces, es un 0.4061 %
q se encuentra 11 veces, es un 0.1015 %
r se encuentra 502 veces, es un 4.6331 %
s se encuentra 351 veces, es un 3.2395 %
t se encuentra 176 veces, es un 1.6244 %
u se encuentra 190 veces, es un 1.7536 %
v se encuentra 39 veces, es un 0.3599 %
w se encuentra 4 veces, es un 0.0369 %
x se encuentra 10 veces, es un 0.0923 %
```

4. Anexos

Instrucciones de compilación.

Estando en el símbolo de sistema para desarrolladores (Windows) o en la terminal (Linux), se escribe en la línea de comando lo siguiente:

- Windows
    - Compilación: cl Practica6.c
    - Ejecución: Practica5.c codigoFuente.c
- Linux
    - Compilación: gcc Practica6.c
    - Ejecución: ./Practica6.c codigoFuente.c

Código fuente

- cola.h

```c
#ifndef cola_h
#define cola_h

typedef struct NodoC{
   char dato;
   struct NodoC *sig;
}*ApNodo;

typedef struct Cnodo{
   ApNodo prim;
   ApNodo ulti;
}*Cola;

Cola nueva(){
   Cola t = (Cola)malloc(sizeof(struct Cnodo));
   t->prim=t->ulti=NULL;
   return t;
}

int esnueva(Cola q){return ((q->prim==NULL)&&(q->ulti==NULL));}

char primero(Cola q){return q->prim->dato;}

Cola formar(Cola q, char e){
   ApNodo t = (ApNodo)malloc(sizeof(struct NodoC));
   t->dato=e;
   t->sig=NULL;
   if(esnueva(q)){
      q->prim=q->ulti=t;
   } else{
      q->ulti->sig=t;
      q->ulti=t;
   }
   return q;
}
```

```
Cola desformar(Cola q){
   ApNodo t;
   if(q->prim==q->ulti){
      free(q->prim);
      free(q);
      return nueva( );
   } else{
      t = q->prim;
      q->prim=q->prim->sig;
      free(t);
      return q;
   }
}

int comparar(Cola q, char *cadena, int size, int pos){
   if(esnueva(q) && cadena[pos] == '\0'){
      return 1;
   } else if(esnueva(q) && cadena[pos] != '\0'){
      return 0;
   } else if(!esnueva(q) && cadena[pos] == '\0'){
      return 0;
   } else if(primero(q) != cadena[pos]){
      return 0;
   } else if(primero(q) == cadena[pos]){
      return comparar(desformar(q),cadena,size,pos + 1);
   }
}
#endif
```

- Practica5.c

```c
/*
NOMBRE DE PROGRAMA: Analizador de Texto
DESCRIPCIÓN: Partiendo de cualquier tipo de archivo, se lee de forma binaria y
se guarda cada byte
en un arreglo del tamaño del archivo. Posteriormente, se hace un análisis del
arreglo con el objetivo
de contabilizar el numero de letras, numeros y signos, para que al final
muestre la cantidad de
veces que se repite junto con el porcentaje.
FECHA: mayo 2022
VERSIÓN: 2.0
AUTOR(ES):
    Espinoza León Jaqueline Viridiana
    García Zacarías Angel Emmanuel
    Hurtado Morales Emiliano
*/

//*****************************************************************
//LIBRERIAS INCLUIDAS
//*****************************************************************

#include <stdio.h>// Todas las funciones como fread, fwrite, fopen, fclose y
printf
#include <stdlib.h>// EXIT_FAILURE y EXIT_SUCCESS
#include <inttypes.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include "cola.h"

#define TAM 100

//*****************************************************************************
****
//DECLARACION DE ESTRUCTURAS
//*****************************************************************************
***

//*****************************************************************
//DECLARACIÓN DE FUNCIONES
//*****************************************************************

void entradaDatos(char*, char*, struct stat);
void analisisLexico(char*, struct stat);
float obtenerPorcentaje(float, struct stat);
```

```c
//*****************************************************************
//PROGRAMA PRINCIPAL
//*****************************************************************

/*Descripcion de la función:
Funcion main, se ingresa el archivo a analizar y realiza todas las llamadas a
funciones que permiten
su análisis léxico.
Input: Archivo de entrada.
Output: Ninguno.
*/

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

    //*****************************************************************
    //Variables del main
    //*****************************************************************

    FILE *archivo;
    char *nombreArchivo;
    char *arreglo;
    int *numC;
    int alt;
    int i;
    int k;
    float tamor;
    float tamcom;

    nombreArchivo = argv[1];

    //*****************************************************************
    //Algoritmo
    //*****************************************************************

    // Checa el tamaño del archivo y manda un error si no logra realizarlo
    struct stat sb;
    if (stat(nombreArchivo, &sb) == -1){
        perror("stat");
        exit(EXIT_FAILURE);
    }
    tamor = sb.st_size;

    // Se crea el arreglo dinámico
    arreglo = malloc(sizeof(char)*sb.st_size);

    entradaDatos(nombreArchivo, arreglo, sb);

    analisisLexico(arreglo, sb);

    return EXIT_SUCCESS;
}
```

```c
//*************************************************************************
//DEFINICIÓN DE FUNCIONES
//*************************************************************************

//
////////////////////////////////////////////////////////////////////////////
//////////////

/*Descripción de función:
Funcion entradaDatos, permite la inserción de cada byte del archivo en el
arreglo previamente creado.
Input: Puntero al archivo, puntero al arreglo y struct stat sb, que habilita
saber de que tamaño es
el archivo.
Output: Arreglo llenado completamente.
Observaciones: La inserción es muy rápida y no conlleva
mucho tiempo de procesamiento.
*/

void entradaDatos(char *nombreArchivo, char *arreglo, struct stat sb){
    char bufer[1];
    FILE *archivo;
    size_t bytesLeidos;
    int i;

    i = 0;

    archivo = fopen(nombreArchivo, "rb"); // Abrir en modo read binario
    // Si es NULL, entonces no existe, o no se pudo abrir
    if (!archivo) {
        printf("¡No se pudo abrir el archivo %s!", nombreArchivo);
    }

    // Mientras no alcancemos el EndOfLine del archivo...
    while (i<sb.st_size) {
        // Leer dentro del búfer; fread regresa el número de bytes leídos
        bytesLeidos = fread(bufer, sizeof(char), sizeof(bufer), archivo);
        arreglo[i] = bufer[0];
        i++;

    }
    // Al final, se cierra el archivo
    fclose(archivo);
}
```

```
////////////////////////////////////////////////////////////////////////////
///////////////

/*Descripción de función:
Funcion obtenerPorcentaje, se obtiene el porcentaje.
Input: un float que representa la frecuencia con la que se repite, struct de
tipo stat,
Output: retorna un porcentaje.
*/

float obtenerPorcentaje(float num, struct stat sb){
    return (num * 100 /sb.st_size);
}

////////////////////////////////////////////////////////////////////////////
///////////////

/*Descripción de función:
Funcion analisisLexico, se realiza todo el análisis del archivo y se
contabilizan los caracteres, numeros y
        simbolos.
Input: puntero de char, struct de tipo stat,
Output: no retorna nada.
*/
```

```
void analisisLexico(char *arreglo, struct stat sb){
    //Variables
    int frecuencias[TAM];
    int i;
    // Banderas para enteros y decimales
    int f_dec = 0;
    int sum;

    char dato[94] = {'(',')','[',']','{','}','<','>','&','|','+','-
','*','/','%','=',',',';
',':','.','#','^','~',34,'?',168,'!',173,'_','$','0','1','2','3','4','5','6','
7','8','9','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q'
,'r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','J','
K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',92,39};

    //Estadarización de las frecuencias
    for(i = 0 ; i < TAM ; i++)
        frecuencias[i] = 0;

    sum  = 0;
```

```
//Análisis
for(i = 0 ; i < sb.st_size ; i++)
{
    if(arreglo[i] == '(')
        frecuencias[0]++;
    else if(arreglo[i] == ')')
        frecuencias[1]++;
    else if(arreglo[i] == '[')
        frecuencias[2]++;
    else if(arreglo[i] == ']')
        frecuencias[3]++;
    else if(arreglo[i] == '{')
        frecuencias[4]++;
    else if(arreglo[i] == '}')
        frecuencias[5]++;
    else if(arreglo[i] == '<')
        frecuencias[6]++;
    else if(arreglo[i] == '>')
        frecuencias[7]++;
    else if(arreglo[i] == '&')
        frecuencias[8]++;
    else if(arreglo[i] == '|')
        frecuencias[9]++;
    else if(arreglo[i] == '+')
        frecuencias[10]++;
    else if(arreglo[i] == '-')
        frecuencias[11]++;
    else if(arreglo[i] == '*')
        frecuencias[12]++;
    else if(arreglo[i] == '/')
        frecuencias[13]++;
    else if(arreglo[i] == '%')
        frecuencias[14]++;
    else if(arreglo[i] == '=')
        frecuencias[15]++;
    else if(arreglo[i] == ',')
        frecuencias[16]++;
    else if(arreglo[i] == ';')
        frecuencias[17]++;
    else if(arreglo[i] == ':')
        frecuencias[18]++;
    else if(arreglo[i] == '.')
        frecuencias[19]++;
    else if(arreglo[i] == '#')
        frecuencias[20]++;
    else if(arreglo[i] == '^')
        frecuencias[21]++;
    else if(arreglo[i] == '~')
        frecuencias[22]++;
    else if(arreglo[i] == '"')
        frecuencias[23]++;
    else if(arreglo[i] == '?')
        frecuencias[24]++;
    else if(arreglo[i] == 168)
        frecuencias[25]++;
    else if(arreglo[i] == '!')
        frecuencias[26]++;
    else if(arreglo[i] == 173)
        frecuencias[27]++;
    else if(arreglo[i] == '_')
        frecuencias[28]++;
    else if(arreglo[i] == '$')
        frecuencias[29]++;
```

```
    else if(arreglo[i] == '0')
        frecuencias[30]++;
    else if(arreglo[i] == '1')
        frecuencias[31]++;
    else if(arreglo[i] == '2')
        frecuencias[32]++;
    else if(arreglo[i] == '3')
        frecuencias[33]++;
    else if(arreglo[i] == '4')
        frecuencias[34]++;
    else if(arreglo[i] == '5')
        frecuencias[35]++;
    else if(arreglo[i] == '6')
        frecuencias[36]++;
    else if(arreglo[i] == '7')
        frecuencias[37]++;
    else if(arreglo[i] == '8')
        frecuencias[38]++;
    else if(arreglo[i] == '9')
        frecuencias[39]++;
    else if(arreglo[i] == 'a')
        frecuencias[40]++;
    else if(arreglo[i] == 'b')
        frecuencias[41]++;
    else if(arreglo[i] == 'c')
        frecuencias[42]++;
    else if(arreglo[i] == 'd')
        frecuencias[43]++;
    else if(arreglo[i] == 'e')
        frecuencias[44]++;
    else if(arreglo[i] == 'f')
        frecuencias[45]++;
    else if(arreglo[i] == 'g')
        frecuencias[46]++;
    else if(arreglo[i] == 'h')
        frecuencias[47]++;
    else if(arreglo[i] == 'i')
        frecuencias[48]++;
    else if(arreglo[i] == 'j')
        frecuencias[49]++;
    else if(arreglo[i] == 'k')
        frecuencias[50]++;
    else if(arreglo[i] == 'l')
        frecuencias[51]++;
    else if(arreglo[i] == 'm')
        frecuencias[52]++;
    else if(arreglo[i] == 'n')
        frecuencias[53]++;
    else if(arreglo[i] == 'o')
        frecuencias[54]++;
    else if(arreglo[i] == 'p')
        frecuencias[55]++;
    else if(arreglo[i] == 'q')
        frecuencias[56]++;
    else if(arreglo[i] == 'r')
        frecuencias[57]++;
    else if(arreglo[i] == 's')
        frecuencias[58]++;
    else if(arreglo[i] == 't')
        frecuencias[59]++;
    else if(arreglo[i] == 'u')
        frecuencias[60]++;
    else if(arreglo[i] == 'v')
        frecuencias[61]++;
    else if(arreglo[i] == 'w')
        frecuencias[62]++;
    else if(arreglo[i] == 'x')
        frecuencias[63]++;
    else if(arreglo[i] == 'y')
        frecuencias[64]++;
    else if(arreglo[i] == 'z')
        frecuencias[65]++;
```

```
    else if(arreglo[i] == 'A')
        frecuencias[66]++;
    else if(arreglo[i] == 'B')
        frecuencias[67]++;
    else if(arreglo[i] == 'C')
        frecuencias[68]++;
    else if(arreglo[i] == 'D')
        frecuencias[69]++;
    else if(arreglo[i] == 'E')
        frecuencias[70]++;
    else if(arreglo[i] == 'F')
        frecuencias[71]++;
    else if(arreglo[i] == 'G')
        frecuencias[72]++;
    else if(arreglo[i] == 'H')
        frecuencias[73]++;
    else if(arreglo[i] == 'I')
        frecuencias[74]++;
    else if(arreglo[i] == 'J')
        frecuencias[75]++;
    else if(arreglo[i] == 'K')
        frecuencias[76]++;
    else if(arreglo[i] == 'L')
        frecuencias[77]++;
    else if(arreglo[i] == 'M')
        frecuencias[78]++;
    else if(arreglo[i] == 'N')
        frecuencias[79]++;
    else if(arreglo[i] == 'O')
        frecuencias[80]++;
    else if(arreglo[i] == 'P')
        frecuencias[81]++;
    else if(arreglo[i] == 'Q')
        frecuencias[82]++;
    else if(arreglo[i] == 'R')
        frecuencias[83]++;
    else if(arreglo[i] == 'S')
        frecuencias[84]++;
    else if(arreglo[i] == 'T')
        frecuencias[85]++;
    else if(arreglo[i] == 'U')
        frecuencias[86]++;
    else if(arreglo[i] == 'V')
        frecuencias[87]++;
    else if(arreglo[i] == 'W')
        frecuencias[88]++;
    else if(arreglo[i] == 'X')
        frecuencias[89]++;
    else if(arreglo[i] == 'Y')
        frecuencias[90]++;
    else if(arreglo[i] == 'Z')
        frecuencias[91]++;
    else if(arreglo[i] == 92)
        frecuencias[92]++;
    else if(arreglo[i] == 39)
        frecuencias[93]++;

}
```

```
    for(i = 0; i<94; i++){
        sum += frecuencias[i];
    }

    for(i = 0; i<94; i++){
        printf("%c se encuentra %i veces, es un %0.4f %% \n", dato[i],
frecuencias[i], obtenerPorcentaje((float)frecuencias[i],sb));
    }
}
```