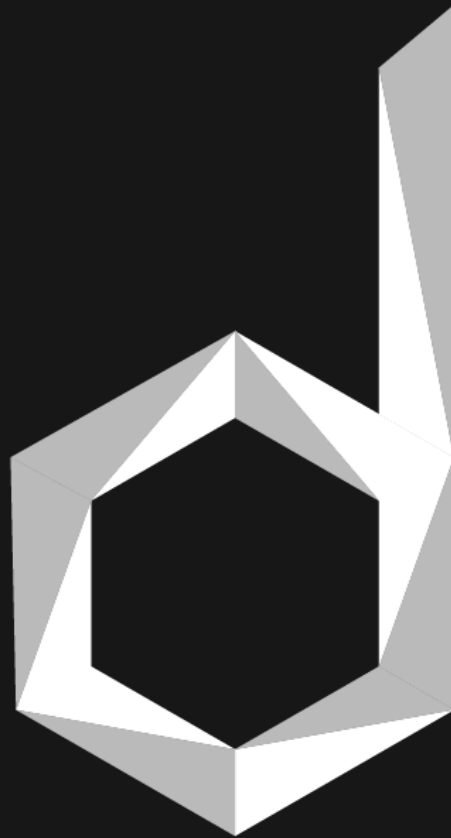


INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

INSTRUMENTACIÓN VIRTUAL

PYTHON 3.9.7, C# & LABVIEW

Conteo de Caracteres Repetidos

Contenido

| | |
|-----------------------------------------------------------------|----|
| Instrucciones – Conteo de Caracteres Repetidos: | 2 |
| Código Python – Visual Studio Code (Logo Azul): | 4 |
| Resultado del Código Python | 7 |
| Código C# (.Net Framework) – Visual Studio (Logo Morado): | 8 |
| Código con resultado en consola Sin uso de Diccionarios: | 8 |
| Resultado del Código C# | 9 |
| Código con resultado en consola Con uso de Diccionarios: | 10 |
| Diagrama LabVIEW: | 11 |



Instrucciones – Conteo de Caracteres Repetidos:

Escriba un programa que cuente la frecuencia de los caracteres de una cadena de texto. Utilice como ejemplo la cadena “google.com”.

Pseudocódigo.

1. Contar el número total de caracteres.
2. Identificar los índices de los caracteres que se repiten.
 - a. Por ejemplo, en google.com se tiene 9 caracteres, donde el índice 0 corresponde a la letra g, el índice 1 a la primera letra o, y así sucesivamente hasta el índice 9 que corresponde a la letra m, entonces el vector con los índices de los caracteres que se repiten queda como [0, 1, 1, 0, 4, 5, 6, 7, 1, 9], donde el 0 corresponde a la letra g (la cual aparece dos veces), el 1 corresponde a la letra o, que aparece tres veces, el índice 4 corresponde a la letra l, y así sucesivamente.

Inicialmente se puede definir un vector `chIndex` con una longitud igual al número de caracteres que contenga la cadena de texto. Es decir:

`chIndex = [0, ..., 0]`

Luego por medio de dos ciclos **for**, se genera un vector que contenga el valor del índice en donde por primera vez apareció un carácter, el cual pertenece al paso 2.

- b. En el caso de `st = “google.com”` el vector generado en este proceso es de nueve elementos: `chIndex = [0, 1, 1, 0, 4, 5, 6, 7, 1, 9]`. Note que la longitud obtenida de este vector es: `strLen = 9`.

```
for (int i = 0; i < strLen; i++) {  
    temp = st[i]; //Esta variable temporal guarda cada carácter i-ésimo de la palabra st.  
  
    /*En este segundo ciclo for se recorre elemento a elemento el string. Si st[i] = st[j] se  
    rompe el ciclo anidado for (con índice j) y se guarda el valor de j, pasando así al siguiente  
    elemento de st.*/  
  
    for (int j = 0; j < strLen; j++) {  
        if (temp == st[j]) {  
            chIndex[i] = j;  
            break;  
        }  
    }  
}
```

3. Reordenar el vector generado en el proceso 2, de valor menor a mayor, por medio de la función `sort()`.



- a. Por ejemplo, si el vector generado en el proceso 2 es [0, 1, 1, 0, 4, 5, 6, 7, 1, 9], se reordena como [0, 0, 1, 1, 1, 4, 5, 6, 7, 9].
4. Generar un nuevo vector que elimina los índices repetidos.
 - a. Es decir, si se tiene el vector [0, 0, 1, 1, 1, 4, 5, 6, 7, 9], el vector generado en este proceso es [0, 1, 4, 5, 6, 7, 9].

El vector de índices se reordena en el proceso 3, es decir que del vector del proceso 2 (chIndex = [0, 1, 1, 0, 4, 5, 6, 7, 1, 9]), se produce el vector chIndex = [0, 0, 1, 1, 1, 4, 5, 6, 7, 9] con la función sort().

- b. Este arreglo, por medio de los dos ciclos **for** previamente explicados, se reduce al vector **chNewIndex** = [0, 1, 4, 5, 6, 7, 9], en el cual se eliminan los índices repetidos y además se encuentran ordenados de menor a mayor.

```
for (int i = 0; i < chIndex.Length; i++) {
    hold = chIndex[i];

    comp = i;

    for (int comp = 0; j < chIndex.Length; comp++) {
        if (hold != chIndex[comp]) {
            i = comp;

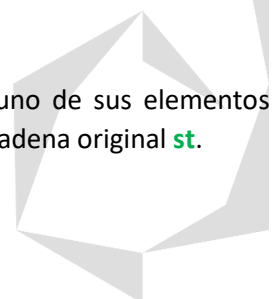
            //Agrega un elemento al vector de chNewIndex
            Array.Resize(ref chNewIndex, chNewIndex.Length+1)
            chNewIndex[chNewIndex.Length-1] = chIndex[comp]
            break;
        }
    }
}
```

5. Determinar la frecuencia para cada uno de los caracteres, en base al vector creado en el proceso número 3.
 - a. En este caso, el carácter con índice 0 es g, el cual aparece 2 veces, el carácter asociado al índice 1 es o y aparece 3 veces, todos los demás caracteres aparecen una sola vez. El vector generado en este proceso es [2, 3, 1, 1, 1, 1, 1].

Inicialmente se puede definir un vector freq con una longitud igual al que tiene el vector **chNewIndex** = [0, 1, 4, 5, 6, 7, 9]. Es decir, un vector de 7 elementos dado por:

freq = [0, 0, 0, 0, 0, 0, 0]

Luego por medio de dos ciclos **for**, se genera un vector que en cada uno de sus elementos guarde la frecuencia con la que aparece cada uno de los caracteres de la cadena original **st**.



b. En nuestro ejemplo, se creará el vector `freq = [2, 3, 1, 1, 1, 1, 1]` de la cadena original.

```
for (int i = 0; i < chNewIndex.Length; i++) {  
    sumfreq = 0;  
    for (int j = 0; j < chIndex.Length; j++) {  
        if (chIndex[j] == chNewIndex[i]) {  
            sumfreq += 1;  
        }  
        freq[i] = sumfreq;  
    }  
}
```

6. Imprimir en pantalla los caracteres, sin que se repitan utilizando el vector del paso 4, y su frecuencia con el vector del paso 5.

Código Python – Visual Studio Code (Logo Azul):

```
# -*- coding: utf-8 -*-  
  
#Comentario de una sola linea con el simbolo #, en Python para nada se deben poner acentos sino el programa  
#puede fallar o imprimir raro en consola, la siguiente línea de código es para que no tenga error, pero aún  
#así al poner un acento, la palabra saldrá rara en consola.  
#La línea debe ponerse tal cual como aparece y justo al inicio.  
  
#Con cls se borra el contenido de la consola y con el botón superior izquierdo de Play se corre el programa.  
#Para comentar en Visual Studio Code varias líneas de código se debe usar las líneas:  
#[CTRL] + K (VSCode queda a la espera). Después pulsa [CTRL] + C para comentar y [CTRL] + U para descomentar.  
  
#CÓDIGO PARA VER LA FRECUENCIA DE LETRAS QUE EXISTEN EN UNA SOLA PALABRA:  
st = "google.com" #Palabra de ejemplo a analizar  
  
#PROCESO:  
#1.- CONTAR EL NÚMERO TOTAL DE CARACTERES.  
#len(): El método len() sirve para calcular el tamaño de una cadena de caracteres, lista o array  
stLenght = len(st)  
#Variable auxiliar que guarda cada caracter de forma individual para  
lastChar = ""  
#Vector con una longitud igual al número de caracteres de la palabra de ejemplo, donde se guardan los índices en  
#donde por primera vez apareció un carácter en la palabra analizada.
```

```

chIndex = []

#2.- IDENTIFICAR LOS ÍNDICES DE LOS CARACTERES QUE SE REPITEN. POR EJEMPLO, en google.com SE TIENEN 9 CARACTERES,
#DONDE EL ÍNDICE 0 CORRESPONDE A LA LETRA g, EL ÍNDICE 1 A LA PRIMERA LETRA o Y ASÍ SUCEATIVAMENTE HASTA EL ÍNDICE 9
#QUE CORRESPONDE A LA LETRA m. ENTONCES EL VECTOR CON LOS ÍNDICES DE LOS CARACTERES QUE SE REPITEN QUEDA COMO
#[0, 1, 1, 0, 4, 5, 6, 7, 1, 9], DONDE EL 0 CORRESPONDE A LA LETRA g (la cual aparece dos veces), EL 1 CORRESPONDE A
#LA LETRA o, QUE APARECE TRES VECES, EL ÍNDICE 4 CORRESPONDE A LA LETRA l Y ASÍ SUCEATIVAMENTE. ESTO SE REALIZA A
#TRAVÉS DE DOS BUCLES FOR ANIDADOS DONDE SE COMPARAN LAS LETRAS UNA A UNA CON TODAS LAS LETRAS DEL STRING Y EL ÍNDICE
#GUARDADO EN EL VECTOR ES EL DE LA PRIMERA VEZ DONDE APARECIÓ LA LETRA ANALIZADA.

#Bucle for con rango definido: Cuando en un bucle for se ponen dentro del paréntesis del range un primer y segundo
#valor separados por comas, se le está indicando al bucle de qué valor a que valor va a contar para repetir la
#ejecución del bucle.

for i in range(0, stLenght):
    #Guarda de uno en uno los caracteres correspondientes a los índices de 0 hasta el tamaño de la palabra de ejemplo.
    lastChar = st[i]

    #En el bucle for anidado se recorre caracter a caracter el string. Si st[i] = st[j], osea que la letra identificada
    #en el primer bucle for sea igual a la reconocida en el segundo bucle for, se rompe el ciclo anidado for con
    #índice j y se guarda el valor numérico del índice j en el vector chIndex, pasando así a analizar el siguiente
    #caracter de st.
    for j in range(0, stLenght):
        if(lastChar == st[j]): #st[i] = st[j], osea que la letra analizada en el primer y segundo for sean iguales
            chIndex.append(j) #append(): El método append() sirve para agregar valores a una lista, array o diccionario
            break #Se rompe el ciclo anidado for con índice j

print("1 y 2.- Los índices de los caracteres que se repiten en el string son: \n", chIndex)

#3.- REORDENAR EL VECTOR GENERADO EN EL PROCESO 2 DE VALOR MENOR A MAYOR POR MEDIO DE LA FUNCIÓN SORT.
#sort(): El método sort() sirve para ordenar de menor a mayor los elementos numéricos de una lista o diccionario.
chIndexSort = sorted(chIndex)

print("3.- Los índices de los caracteres que se repiten ordenados de mayor a menor son: \n", chIndexSort)

#4.- GENERAR UN NUEVO VECTOR QUE ELIMINA LOS ÍNDICES REPETIDOS, ESTO PARA PODER AL FINAL ASOCIAR A CADA LETRA DE LA
#PALABRA ANALIZADA LA FRECUENCIA CON LA QUE APARECE EN LA PALABRA.

index = 0

#Vector donde se guardan los índices no repetidos de la lista chIndexSort.
chNewIndex = []

#Se agrega el primer elemento de la lista chIndexSort que contiene los índices ordenados de menor a mayor a la lista
#chNewIndex que incluye los mismos índices pero sin que estos se repitan.
chNewIndex.append(chIndexSort[0])

#Bucle for exterior: En este se recorre toda la palabra inicial y se crean dos variables auxiliares:
#hold: Variable auxiliar para analizar individualmente cada valor de la lista chIndexSort y compararlo con todos sus
#demás valores.

```

```

#comp: Variable auxiliar para empezar el segundo bucle for anidado desde donde no se haya encontrado valores repetidos.
#Recordemos que el tamaño de la palabra original, el vector chIndex y chIndexSort es el mismo ya que en él se analizaron
#los índices de todas las letras de la palabra donde se repitiera cada letra.
for i in range(0, stLenght): #stLenght = len(st) = len(chIndex) = len(chIndexSort)
    hold = chIndexSort[index] #Variable que guarda uno a uno los valores contenidos en la lista chIndexSort.
    comp = index #Variable que guarda el índice de la lista chIndexSort donde no se encontraron valores repetidos.
    #Bucle for anidado: En este se recorre toda la palabra inicial, partiendo desde el índice guardado en la variable
    #comp, que cambia de valor cuando no encuentra valores repetidos en la lista chIndexSort.
    for j in range(comp, stLenght):
        #En el bucle anidado solo se ejecuta algo cuando el valor del
        if(hold != chIndexSort[j]):
            #Cambio de valor de la variable index y comp por el índice donde no se encontró valores repetidos
            index = j #index = comp = j, cuando el valor actual de la lista es distinto al que le sigue
            #Cuando no se repiten valores en la lista chIndexSort se agregan a la lista chNewIndex y se sale del
            #bucle for anidado j
            chNewIndex.append(chIndexSort[j])
            break

print("4.- Los índices de los caracteres que se repiten ordenados de mayor a menor y sin índices repetidos son: \n", chNewIndex)

#5.- DETERMINAR LA FRECUENCIA DE CADA UNO DE LOS CARACTERES, EN BASE AL VECTOR CREADO EN EL PROCESO 4.
freq = [] #Vector donde se guarda la frecuencia de cada letra analizada perteneciente a la palabra original.
#Bucle for exterior: Este parte de 0 hasta el tamaño de la lista chNewIndex, que contiene todos los índices no repetidos
#y ordenados de menor a mayor de la lista que indica la posición en donde por primera vez apareció un carácter en la
#palabra analizada, en este se crea una variable auxiliar que indicará la frecuencia con la que aparece cada letra en
#la palabra original.
#stLenght != len(chNewIndex)
for i in range(0, len(chNewIndex)):
    #Variable auxiliar que indica la frecuencia de cada letra, esto se logra comparando los valores de los vectores
    #chNewIndex y chIndexSort, ya que cada vez que un valor de la lista chNewIndex sea igual a un valor la lista
    #chIndexSort es porque la letra aparece una vez en la palabra original.
    sumFreq = 0
    #Bucle anidado: En este se compara cada índice de la lista chNewIndex con cada índice de la lista chIndexSort, cada
    #vez que sean iguales los valores, se le sumará un 1 a la variable sumFreq para saber la frecuencia de esa letra en
    #específico, al ya no cumplirse esa condición, el programa se saldrá del bucle anidado y agregará un valor a la lista
    #freq, el segundo bucle for parte desde 0 hasta el tamaño de la palabra original porque:
    #stLenght = len(st) = len(chIndex) = len(chIndexSort)
    for j in range(0, stLenght):
        #Comparación de los índices de la lista chIndexSort y chNewIndex para saber la frecuencia de cada letra.
        if(chIndexSort[j] == chNewIndex[i]):
            sumFreq += 1 #Se suma un 1 a la variable sumFreq cada que sea igual un índice de ambas listas
    freq.append(sumFreq) #append(): El método append() sirve para agregar valores a una lista, array o diccionario

print("5.- Las frecuencias de las letras que se repiten en la palabra son: \n", freq)

```

```
#6.- IMPRIMIR EN CONSOLA LOS CARACTERES SIN QUE SE REPITAN Y SU FRECUENCIA, ADEMÁS DE ASOCIARLOS TODOS DENTRO DE UN
#DICCIONARIO.

freq_caract = {} #Diccionario que contiene cada letra de la palabra junto con la frecuencia con la que aparece.
letras = [] #Vector donde se guardarán las letras de los índices no repetidos de la palabra original.

#Bucle for para crear la lista letras que contiene todas las letras no repetidas en la palabra
for i in range(0, len(chNewIndex)):
    letras.append(st[chNewIndex[i]])

#zip(): El método lo que hace es crear un tipo de dato llamado tupla, una tupla es un conjunto de datos del mismo o
#diferente tipo, si se quisiera .
tupla = zip(letras, freq)

#tuple(): El método crea un objeto tuple para que se le puedan aplicar los métodos correspondientes o en este caso se
#pueda imprimir en consola
lista2D = tuple(tupla)
print("6.- Tupla: ", lista2D)

#Convertir dos listas en un diccionario Python: Python tiene una función incorporada llamada zip(), que agrega tipos de
#datos iterables en una tupla y la función dict() crea un diccionario a partir de la colección dada, para lograr esto se
#debe realizar todo en una misma línea de código o usar el método tuple como paso intermedio.
freq_caract = dict(lista2D) # = freq_caract = dict(zip(letras, freq))
print("6.- Diccionario: ",freq_caract)

#Imprimir en consola el key y value de un diccionario
#Operación para imprimir de una por una las ciudades del diccionario
for letra in freq_caract:
    print("6.- Letra: ", letra , ", Frecuencia: ", freq_caract[letra])
```

Resultado del Código Python

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

PS C:\Users\diego\OneDrive\Documents\Aprendiendo\Python\1.-Instrumentación Virtual> & D:/Users/diego/AppData/Local/Programs/Python/Python39/python.exe
"c:/Users/diego/OneDrive/Documents/Aprendiendo/Python/1.-Instrumentación Virtual/6.-Conteo de Caracteres Repeditos.py"
1 y 2.- los índices de los caracteres que se repiten en el string son:
[0, 1, 1, 0, 4, 5, 6, 7, 1, 9]
3.- los índices de los caracteres que se repiten ordenados de mayor a menor son:
[0, 0, 1, 1, 1, 4, 5, 6, 7, 9]
4.- Los índices de los caracteres que se repiten ordenados de mayor a menor y sin índices repetidos son:
[0, 1, 4, 5, 6, 7, 9]
5.- Las frecuencias de las letras que se repiten en la palabra son:
[2, 3, 1, 1, 1, 1]
6.- Tupla: (('g', 2), ('o', 3), ('l', 1), ('e', 1), ('.', 1), ('c', 1), ('m', 1))
6.- Diccionario: {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
6.- Letra: g , Frecuencia: 2
6.- Letra: o , Frecuencia: 3
6.- Letra: l , Frecuencia: 1
6.- Letra: e , Frecuencia: 1
6.- Letra: . , Frecuencia: 1
6.- Letra: c , Frecuencia: 1
6.- Letra: m , Frecuencia: 1
PS C:\Users\diego\OneDrive\Documents\Aprendiendo\Python\1.-Instrumentación Virtual> 
```



Código C# (.Net Framework) – Visual Studio (Logo Morado):

Código con resultado en consola Sin uso de Diccionarios:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

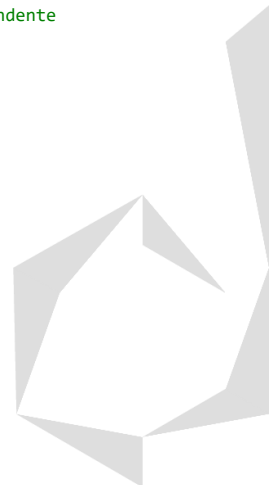
//El código se corre dando clic en CTRL + F5
namespace _5._Conteo_de_Caracteres_Repeditos
{
    //Namespace: Es una parte del código donde se puede crear una biblioteca con varias clases y cuando queramos crear otra
    //clase podemos importar el namespace del proyecto y eso va a importar todas las clases que lo conforman
    class ExtraerCaracteresDeString
    {
        static void Main(string[] args)
        {
            //El programa lo que hará es leer un string e indicar cuantas veces aparece cada una de sus letras.
            string st = "google.com";
            int n = st.Length; //Número de caracteres
            char temp;
            int[] chIndex = new int[0]; //Array con 0 elementos
            int count = 0;
            int[] chNewIndex = new int[0];
            int hold;
            int comp;
            int index;
            int[] freq;
            int sumFreq;
            for (int i = 0; i < n; i++)
            {
                temp = st[i];
                for (int j = 0; j < n; j++)
                {
                    if (temp == st[j])
                    {
                        count = count + 1;
                        //El Método resize lo que hace es aumentar el tamaño de un array, en específico aumenta
                        //el tamaño del array chIndex que ahora tiene tamaño de 0 elementos, se usa la palabra
                        //reservada ref para declarar a qué Array se le quiere modificar su tamaño
                        Array.Resize(ref chIndex, count);
                        chIndex[i] = j;
                        break;
                    }
                }
            }
            //Bucle for para la impresión en pantalla del arreglo chIndex
            Console.WriteLine("Arreglo chIndex");
            for (int i = 0; i < chIndex.Length; i++)
            {
                Console.Write(chIndex[i] + " ");
            }
            Console.WriteLine(); //Para imprimir en consola un salto de línea

            Array.Sort(chIndex); //Con esto se ordenan en forma ascendente los elementos del array chIndex

            //Bucle for para la impresión en pantalla del arreglo chIndex ordenado de forma ascendente
            Console.WriteLine("Arreglo chIndex ordenado en forma ascendente");
            for (int i = 0; i < chIndex.Length; i++)
            {
                Console.Write(chIndex[i] + " ");
            }
            Console.WriteLine(); //Para imprimir en consola un salto de línea

            index = 0;
            Array.Resize(ref chNewIndex, chNewIndex.Length + 1); //Aumenta el tamaño de un array
            chNewIndex[chNewIndex.Length - 1] = chIndex[0];

            for (int i = 0; i < chIndex.Length; i++)
            {
                int v = chIndex[index];
                hold = v;
                //comp = index;
            }
        }
    }
}
```



```

        //for (; comp < chIndex.Length; comp++){}
        //Esto lo que hace es lo mismo que el for que se define abajo
        for (comp = index; comp < chIndex.Length; comp++)
        {
            if (hold != chIndex[comp])
            {
                index = comp;
                Array.Resize(ref chNewIndex, chNewIndex.Length + 1);
                chNewIndex[chNewIndex.Length - 1] = chIndex[comp];
                break;
            }
        }
    }
}

//Bucle for para la impresión en pantalla del arreglo chNewIndex
Console.WriteLine("Arreglo chNewIndex");
for (int i = 0; i < chNewIndex.Length; i++)
{
    Console.Write(chIndex[i] + " ");
}
Console.WriteLine(); //Para imprimir en consola un salto de línea

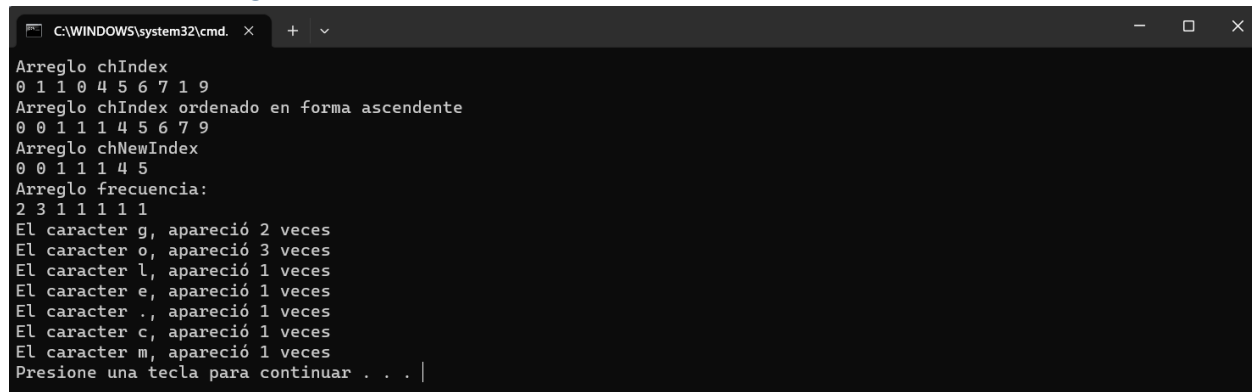
freq = new int[chNewIndex.Length];
for (int i = 0; i < chNewIndex.Length; i++)
{
    sumFreq = 0;
    for (int j = 0; j < n; j++)
    {
        if (chIndex[j] == chNewIndex[i])
        {
            sumFreq = sumFreq + 1;
        }
    }
    freq[i] = sumFreq; //Asignación de valor al array freq
}

Console.WriteLine("Arreglo frecuencia:");
for (int i = 0; i < freq.Length; i++)
{
    Console.Write(freq[i] + " ");
}
Console.WriteLine(); //Para imprimir en consola un salto de línea

//Impresión de resultado
for (int i = 0; i < freq.Length; i++)
{
    Console.WriteLine("El caracter {0}, apareció {1} veces", st[chNewIndex[i]], freq[i]);
}
}
} //Método Main
} //Clase del proyecto
} //Espacio de nombres
//El código se corre dando clic en CTRL + F5

```

Resultado del Código C#



```

C:\WINDOWS\system32\cmd. X + v
Arreglo chIndex
0 1 1 0 4 5 6 7 1 9
Arreglo chIndex ordenado en forma ascendente
0 0 1 1 1 4 5 6 7 9
Arreglo chNewIndex
0 0 1 1 1 4 5
Arreglo frecuencia:
2 3 1 1 1 1 1
El caracter g, apareció 2 veces
El caracter o, apareció 3 veces
El caracter l, apareció 1 veces
El caracter e, apareció 1 veces
El caracter ., apareció 1 veces
El caracter c, apareció 1 veces
El caracter m, apareció 1 veces
Presione una tecla para continuar . . . |

```

Código con resultado en consola Con uso de Diccionarios:

```
7 namespace CaracteresRepetidosDiccion
8 {
9     class CharFreqDemo
10    {
11        /*Método que genera un diccionario que tiene de parámetro el
12         * string al que se le obtendrá la frecuencia de sus caracteres*/
13        static void char_frequency(string str1)
14        {
15            /*Creación de un diccionario que contiene en sus llaves caracteres
16             * (char) y en sus valores la frecuencia (int)*/
17            Dictionary<char, int> dict= new Dictionary<char, int>();
18            dict.Clear();//Limpia el diccionario dict
19            foreach (char n in str1)
20            {
21                try
22                {
23                    //Agrega una llave,si ésta no existe y le asocia una
24                     frecuencia de 1
25                    dict.Add(n, 1);
26                }//Try
27                catch
28                {
29                    /*Si ya existía la llave, no la agrega; pero
30                     * aumenta en 1 el valor de su frecuencia*/
31                    dict[n] +=1;
32                }//Catch
33            }//Foreach
34
35            //Impresión del diccionario
36            foreach (KeyValuePair<char, int> kvp in dict)
37            {
38                Console.WriteLine("Key = {0}, Value = {1}",
39                    kvp.Key, kvp.Value);
40            }//Foreach
41        }//Fin del método char_frequency
42
43        static void Main(string[] args)
44        {
45            //Llamada al método con el string="google.com"
46            char_frequency("google.com");
47        }//Fin de Main
48    }//Fin de la clase CharFreqDemo
49 }//Fin del espacio de nombres CaracteresRepetidosDiccion
```

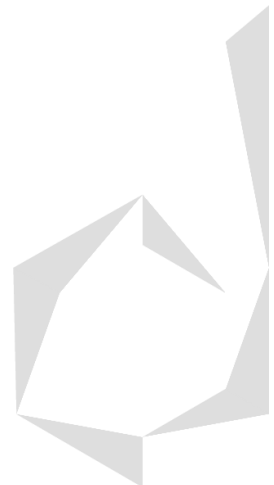


Diagrama LabVIEW:

