Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: Camilo Esteban Zapata O.
Corregido por (alumno ayudante): José I. Veloso Inzunza

Introducción al análisis de histograma

Una imagen digital es representada a partir de matrices numéricas que corresponden a las intensidades de colores asociadas a ellas. Es decir, lo que nosotros definimos como color, de forma digital no es más que la combinación de matrices numéricas. La mezcla de ellas genera variedades de colores interpretadas por nuestros ojos.

En esta actividad se propone analizar números aleatorios almacenados en un archivo de texto, los cuales son los valores de aquellas intensidades.

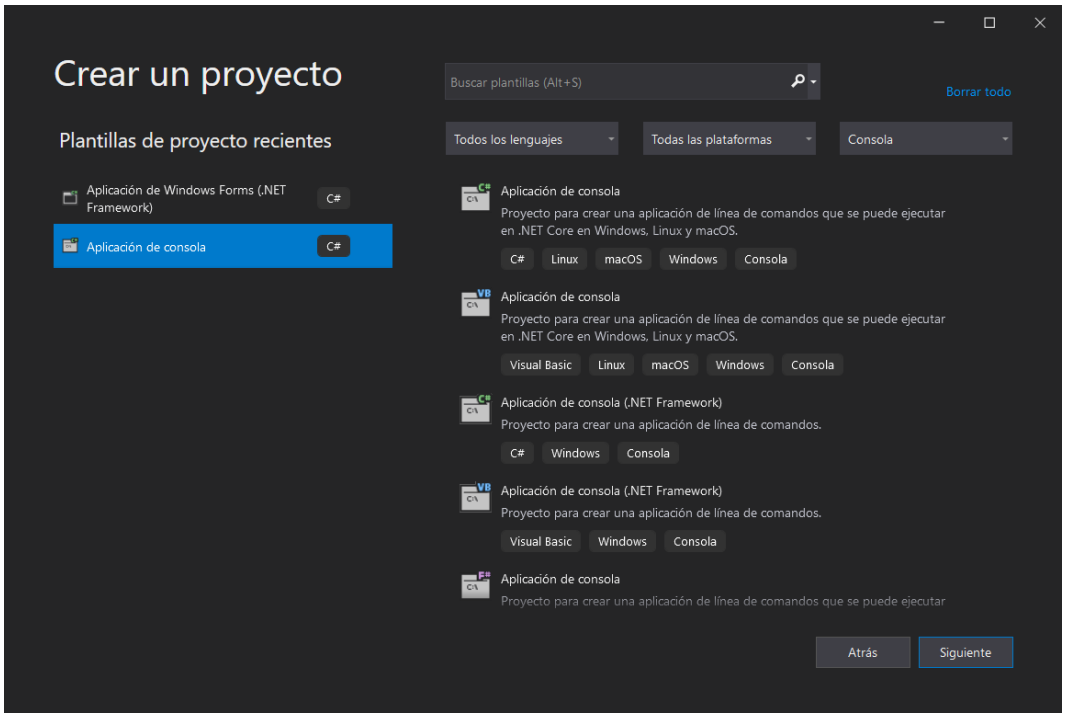
Objetivos:

1. Analizar valores de intensidades de una imagen y visualizarlos en pantalla.
2. Ordenar los datos obtenidos de acuerdo con su frecuencia.
3. Mostrar en pantalla la distribución de valores de intensidades de la imagen analizada.

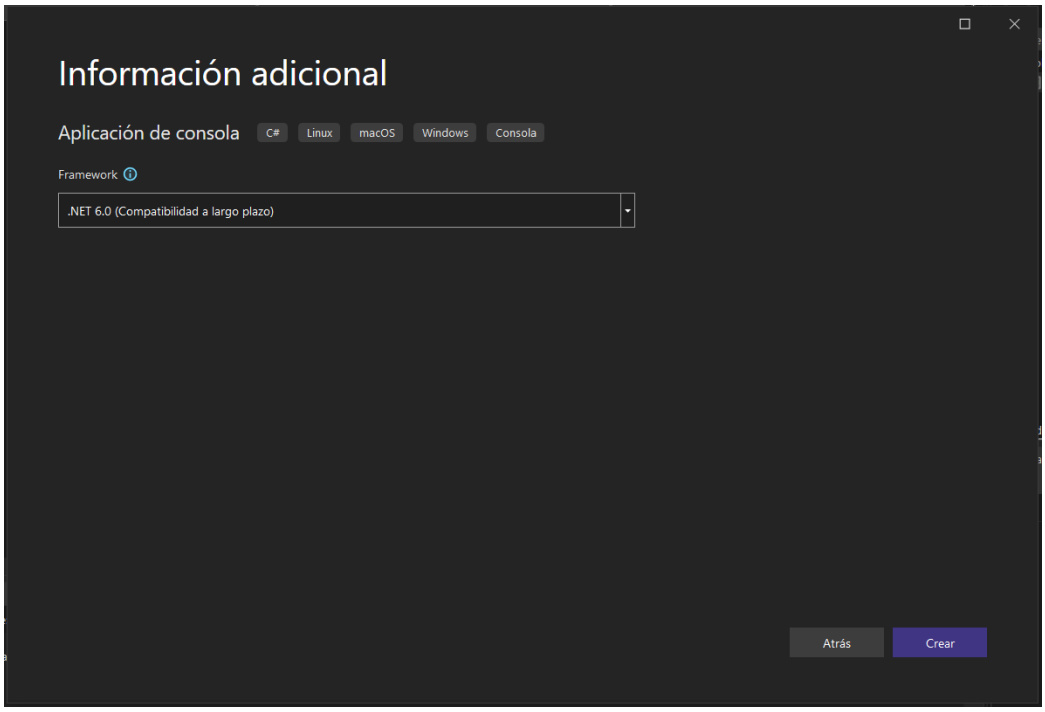
Nota: Si no dispone del archivo de texto, puede generarlo creando una matriz de 5x5 con números aleatorios separados por un espacio de valores entre 0 y 255, y almacenándolo en formato .txt.

Desarrollo

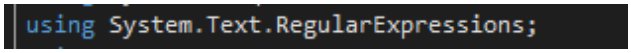
1. Cree una aplicación de consola en leguaje C# tal como se indica en la siguiente imagen.



2. Seleccione el framework .NET 6.0.



3. Agregue en el encabezado del programa la siguiente biblioteca:



4. Cree una variable del tipo *String* y asígnele la dirección de la ruta del archivo de texto en donde se encuentran los valores de intensidades, también cree una variable output del mismo tipo, pero con una cadena vacía:

```
string direccion = @"C:\Users\Camilo\Documents\2020\Prueba.txt";
string output = "";
```

4. El siguiente código permite mostrar en pantalla un menú de usuario, además de instanciar variables. Escríbalo a continuación del código anterior:

```
try
{
    using (StreamReader Linea = new StreamReader(direccion))
    {
        String Speach = Linea.ReadLine();

        Console.WriteLine("Histograma");
        Console.WriteLine();
        Console.WriteLine("1. mostrar histograma");
        Console.WriteLine("2. contenido txt");
        Console.WriteLine("3. salir");
        Console.WriteLine("Ingrese numero de opcion ");
        string choice = Console.ReadLine();
        int choiceInt = Int32.Parse(choice);

        String[] SpeachSplit = Speach.Split();
        Dictionary<string, int> Histogram = new Dictionary<string, int>();

        if (choiceInt == 1)
        {
            var result = Speach.Split(new[] { ' ' },
                StringSplitOptions.RemoveEmptyEntries).Select(x => Regex.Replace(x.ToLower(),
                "[^a-zA-Z0-9]", " ")).GroupBy(x => x).ToDictionary(x => x.Key, x => x.Count());
            Print(result.OrderByDescending(x => x.Value).ToList());

        }
        else if (choiceInt == 2)
        {
            if (!string.IsNullOrEmpty(Speach))
            {
                Console.WriteLine(Speach);
            }
        }

    }
}
catch (Exception e)
{
    output = "Error: " + e.ToString();
    Console.WriteLine("\n" + output + "\n");
}
```

Los valores que tiene *choiceInt* en el *if* y *else* anterior tienen las siguientes funciones:

- Si *choiceInt* tiene valor 1, recorre y ordena los valores del txt agrupando valores repetidos.
- Si *choiceInt* tiene valor 2, muestra por pantalla los valores del txt.

```
if (choiceInt == 1)
{
    var result = Speach.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Select(
        x => Regex.Replace(x.ToLower(), "[^a-zA-Z0-9]", " ")).GroupBy(x => x).ToDictionary(x => x.Key, x => x.Count());
    Print(result.OrderByDescending(x => x.Value).ToList());

}
else if (choiceInt == 2)
{
    if (!string.IsNullOrEmpty(Speach))
    {
        Console.WriteLine(Speach);
    }
}
```

5. Cree la siguiente función *Print* para darle diseño al histograma en la salida de la consola:

```
static void Print(List<KeyValuePair<string, int>> list)
{
    //obtener la longitud maxima de todas las palabras para que sea posible alinearlas
    var max = list.Max(x => x.Key.Length);
    foreach (var item in list)
    {
        //alinear a la derecha usando PadLeft y la longitud maxima
        Console.Write(item.Key.PadLeft(max));

        Console.Write(" ");
        //cambiar el color de la consola
        Console.BackgroundColor = ConsoleColor.DarkBlue;

        //agregar barras
        for (var i = 0; i < item.Value; i++)
        {
            Console.Write("#");
        }
        //cambiar de nuevo el color de la consola
        Console.BackgroundColor = ConsoleColor.Black;

        Console.Write(" ");
        //crea nuevas lineas
        Console.WriteLine(item.Value);
    }
}
```

6.Ejecute el programa, como resultado debería obtener algo similar a lo siguiente: