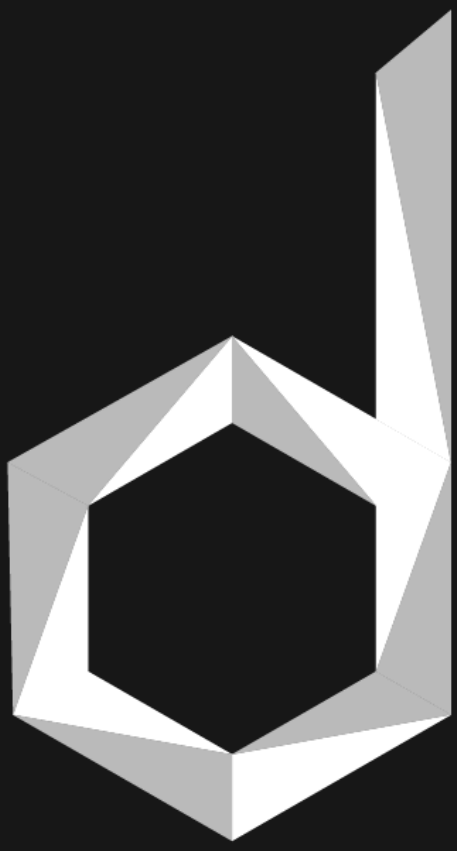
INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

DATA SCIENCE

PYTHON 3.9.7, C# & LABVIEW

Graficar un Vector y Almacenar sus Datos en un Archivo

Contenido

Instrucciones – Graficar y Almacenar sus Datos en un Archivo:

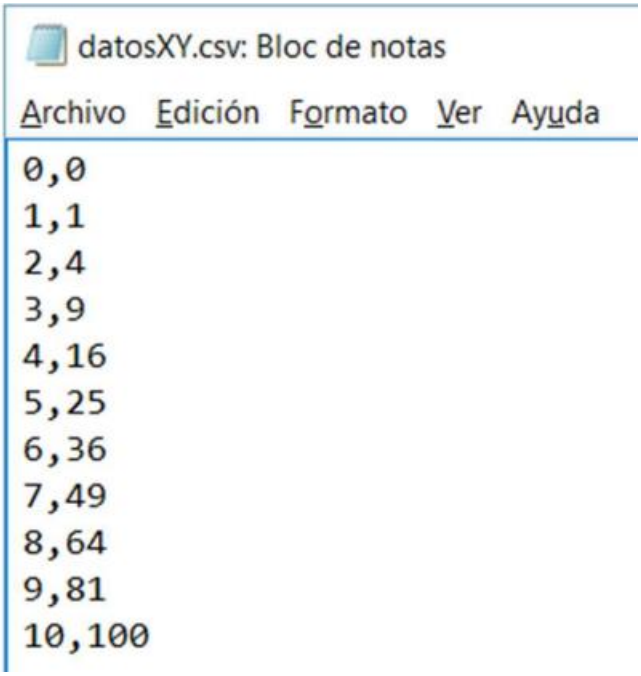
Escriba un programa que haga lo siguiente:

1. Genere el vector x y por medio de la ecuación $y = x^2$ genere el vector y.

$$x = [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10]$$

$$y = [0 \quad 1 \quad 4 \quad 9 \quad 16 \quad 25 \quad 36 \quad 49 \quad 64 \quad 81 \quad 100]$$

2. Escriba en un archivo en formato de Excel .csv (comma−separated values) los vectores anteriores. En la primera columna anote los datos del vector x y en la segunda los valores del vector y. Llame el archivo DatosXY.csv y sus datos se verán de la siguiente manera:



3. Cierre el archivo.
4. Lea estos datos y grafíquelos.

Pseudocódigo.

1. Generar por medio de un ciclo for los vectores x y y.
2. Crear el archivo llamado DatosXY.csv y escribir en él los datos separados con una coma, como se muestra en la Fig. 1.
3. Abrir el archivo y leer cada uno de sus renglones, eliminando durante este proceso los espacios en blanco y los saltos de línea (“\n”).
    a. Guardar en el vector vect1 de tipo string los datos leídos, el número de elementos del vector, corresponde al número de valores del vector x.
4. Generar el vector que contiene la posición de la coma en cada uno de los elementos de vect1.
    a. Por ejemplo, en el primer elemento, que es 0,0, la coma se ubica en el índice 1.
5. Usar la posición de la coma para truncar el string (slice the string), para así obtener el valor de los vectores x y y.

a. El substring que contiene todos los índices hasta antes al que corresponde a la coma, es el valor de x[i], el que contiene todos los índices posteriores al que corresponde a la coma, es el valor y[i].

6. Graficar los valores de los vectores x y y.

Código Python:

```
# -*- 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 ángulo saldrá raro en consola, la línea debe ponerse tal cual como aparece y justo al inicio.



#VECTOR X^2 HECHO CON DOS BUCLES FOR Y UNA FUNCIÓN (LISTAS DE COMPRENSIÓN):
#Función f(x) = x^2
def f(x):
    return x**2
#Las listas en Python pueden ser usadas como vector.
#Declaración de un vector que represente los valores del eje x, de 0 a 10, el último valor del bucle no es alcanzado.
x_l_c = [i for i in range (11)]
#Declaración de un vector que represente los valores del eje y, usando los valores del eje x, yendo igual de 0 a 10
#pero elevando dichos valores al cuadrado.
y_l_c = [f(x_l_c[i]) for i in range (11)]
#LISTAS DE COMPRENSIÓN: Las listas de comprensión son listas que se crean por medio de ciclos for pero que se declaran
#dentro de la misma lista, el problema de usar este método es que si no sabemos si los datos están bien organizados o
#si pueden venir de varios tipos de datos, algunos numéricos, algunos strings y así, puede ocurrir un error en la
#recopilación de datos, por lo que es mejor entonces usar el método tradicional en esos casos.
print("El vector horizontal creado con lista es: \n", x_l_c)
print("El vector vertical creado con lista es: \n", y_l_c)



#VECTOR X^2 HECHO CON UNA FUNCIÓN Y UN BUCLE FOR (MÉTODO TRADICIONAL):
#Función f(x) = x^2
def f(x):
    return x**2
#Método tradicional (listas)
x_lista = []
y_lista = []
for i in range(11):
    x_lista.append(i)
    y_lista.append(f(i))
```

```
#Arreglo hecho con método tradicional, que es más útil para cuando no sabemos la estructura en la que se reciben
#los datos.
print("El vector horizontal creado con método tradicional es: \n", x_lista)
print("El vector vertical creado con el método tradicional es: \n", y_lista)



#VECTOR X^2 HECHO CON ARREGLOS PERTENECIENTES A LA LIBRERÍA NUMPY:
import numpy as np #Librería numpy: Realiza operaciones matemáticas complejas (matriciales).
#El método arrange(inicio, final, paso) sirve para indicarle al vector de donde empieza, de cuanto en cuanto avanza
#y en donde termina:
x_np = np.arange(0, 11, 1) #El dato se llama numpy array, es distinto a las listas
y_np_v1 = f(x_np)
y_np_v2 = (x_np)**2 #Vectorizado: No utiliza bucles sino que se hace operación directa con vectores
#Arreglo hecho con el objeto numpy array, que es perteneciente a la librería numpy.
print("El vector horizontal creado con numpy es: \n", x_np)
print("El vector vertical creado con numpy versión 1 es: \n", y_np_v1)
print("El vector vertical creado con numpy versión 2 es: \n", y_np_v2)



#ABRIR y ESCRIBIR EN UN ARCHIVO:
#Variable que guarda el directorio y el nombre del archivo creado, se deben reemplazar los guiones\ por /
#Para leer una imagen o cualquier otro archivo se usa la dirección relativa o absoluta de un directorio:
# - Dirección relativa: Es una dirección que busca un archivo desde donde se encuentra la carpeta del
#   archivo python actualmente, esta se debe colocar entre comillas simples o dobles.
# - Dirección absoluta: Es una dirección que coloca toda la ruta desde el disco duro C o cualquier otro
#   que se esté usando hasta la ubicación del archivo, la cual se debe colocar entre comillas simples o
#   dobles.
#   ..      : Significa que nos debemos salir de la carpeta donde nos encontramos actualmente.
#   /       : Sirve para introducirnos a alguna carpeta cuyo nombre se coloca después del slash.
#   .ext    : Se debe colocar siempre el nombre del archivo + su extensión.
#Es un archivo de Excel el que se va a crear, por eso tiene extensión .csv
nombreArchivo = "Ejercicios Python con Archivos y Carpetas/13.-Graficar y Almacenar sus Datos en un Archivo/DatosXY.csv"
#open(): Método que sirve para abrir un archivo cualquiera, para ello es necesario indicar dos parámetros,
#el primero se refiere a la ruta relativa o absoluta del archivo previamente creado y la segunda indica qué
#es lo que se va a realizar con él, el contenido del archivo se asigna a una variable.
#   - w: Sirve para escribir en un archivo, pero borrará la información que previamente contenía el archivo.
#   - a: Sirve para escribir en un archivo sin que se borre la info anterior del archivo, se llama append.
nuevo_archivo = open(nombreArchivo, 'w')
#var_file_open.write(): Método para colocar un string en un archivo previamente abrierto con el método open().
#str(): Lo que hace el método es convertir cada elemento de la lista a un string, esto es porque el método
#write no permite escribir datos de otro tipo en un archivo.
#Se concatenan los datos de un string por medio del símbolo + y poniendo los strings concatenados entre comillas.
for i in range(len(x_lista)):
```

```
    nuevo_archivo.write(str(x_lista[i])+","+str(y_lista[i])+"\n")
#var_file_open.close(): Método para cerrar un archivo previamente abrierto con el método open(), es peligroso
#olvidar colocar este método, ya que la computadora lo considerará como si nunca hubiera sido cerrado, por lo
#cual no podré volver a abrirlo al dar clic sobre él.
nuevo_archivo.close()



#ABRIR y LEER UN ARCHIVO:
#open(): Método que sirve para abrir un archivo cualquiera, para ello es necesario indicar dos parámetros,
#el primero se refiere a la ruta relativa o absoluta del archivo previamente creado y la segunda indica qué
#es lo que se va a realizar con él, el contenido del archivo se asigna a una variable.
#   - r: Sirve para leer el contenido de un archivo.
#Es importante mencionar que no se podrá abrir el archivo si este se encuentra abierto por otro programa.
archivo_lectura = open(nombreArchivo, 'r')
data_str = [] #declaración de una lista
for linea in archivo_lectura:
    #replace(): Método que reemplaza un caracter que se encuentra en un string por otro declarado por nosotros, esto
    #se ejecutará todas las veces que dicho caracter aparezca en el string.
    linea = linea.replace("\n", "") #Método que reemplaza el salto de línea proveniente del archivo por un espacio.
    #append(): Lo que hace es agregar un elemento a la lista (array de python).
    #split(): Método que crea una lista nueva en el index de la lista actual donde nos encontramos, esto sucederá cuando
    #se encuentre el caracter indicado como parámetro, creando así una lista interna en cada una de las posiciones de la
    #lista principal con los elementos que estén separados entre comas.
    data_str.append(linea.split(","))
    #Con alguna restricción se le puede decir que solamente cree una lista con ciertos valores y no todos, por medio
    #de una constante count que se aumente por uno cada vez que el elemento entra en el bucle for y una condicional
    #que diga de dónde a donde quiero que agarre los elementos del archivo para crear la lista
    print(linea) #Impresión con salto de línea, pero se le quitó con el método replace()
#print(): Imprimir un mensaje en consola.
print("Datos extraídos del archivo: \n", data_str) #Impresión de la lista creada con los elementos leídos del archivo.
#var_file_open.close(): Método para cerrar un archivo previamente abrierto con el método open(), es peligroso
#olvidar colocar este método, ya que la computadora lo considerará como si nunca hubiera sido cerrado, por lo
#cual no podré volver a abrirlo al dar clic sobre él.
archivo_lectura.close()



#MÉTODO TRADICIONAL: Forma tradicional de extraer datos y crear una nueva lista vacía de ceros.
lista_ceros = []
for i in range(len(data_str)):
    lista_temp = []
    for i in range(len(data_str[0])):
        lista_temp.append(0)
    lista_ceros.append(lista_temp)
```

```
print("Lista vacía con ceros: \n", lista_ceros) #Lista llena de ceros pero con el tamaño de la lista que quiero obtener



#LISTAS DE COMPRENSIÓN: Las listas de comprensión son listas que se crean por medio de ciclos for pero que se declaran
#dentro de la misma lista, el problema de usar este método es que si no sabemos si los datos están bien organizados o
#si pueden venir de varios tipos de datos, algunos numéricos, algunos strings y así, puede ocurrir un error en la
#recopilación de datos, por lo que es mejor entonces usar el método tradicional en esos casos. Por lo cual las listas
#de compresión son una forma hardcore de obtener los datos del vector x,y del vector y.
lista_comprension_ceros = [[0 for i in range(len(data_str[0]))] for i in range(len(data_str))] #Lista llena de ceros
#Llenado de la lista de ceros
for i in range(len(data_str)):
    for j in range(len(data_str[0])):
        lista_comprension_ceros[i][j] = float(data_str[i][j])
print("Matriz original: \n", lista_comprension_ceros)



#MÉTODO TRADICIONAL: La forma tradicional de extraer datos de un archivo de Excel es buena para cuando no sabemos si
#los datos están bien organizados o pueden ser de varios tipos de datos, algunos numéricos, algunos strings y así.
x_trad_lista = [] #lista vacía para obtener los datos x de forma tradicional.
y_trad_lista = [] #lista vacía para obtener los datos y de forma tradicional.
for i in range(len(lista_comprension_ceros)):
    x_trad_lista.append(lista_comprension_ceros[i][0])
    y_trad_lista.append(lista_comprension_ceros[i][1])
#print(): Imprimir un mensaje en consola.
print(x_trad_lista)
print(y_trad_lista)



#MATRIZ TRANSPUESTA: Se debe obtener la transpuesta de los datos extraídos porque siempre de los archivos de Excel
#estos son extraídos de forma vertical y los necesitamos de forma horizontal para que puedan ser graficados con python.
#MATRIZ TRANSPUESTA (FORMA TRADICIONAL): Obtención de una matriz transpuesta de forma tradicional, en los elementos
#de las matrices transpuestas, lo que pasa es que se intercambian las coordenadas de sus elementos aij=aji.
def transpuesta(matriz):
    m = len(matriz) #Número de renglones de la matriz.
    n = len(matriz[0]) #Número de columnas de la matriz.
    trans = [[0 for i in range(m)] for i in range(n)]
    for i in range(n):
        for j in range(m):
            trans[i][j] = matriz[j][i]
        #for j
    #for i
    return trans
```

```
#función transpuesta
matriz_transpuesta = transpuesta(lista_comprension_ceros)
print("Matriz transpuesta: \n", matriz_transpuesta)
x_lista_transpuesta = matriz_transpuesta[0]
y_lista_transpuesta = matriz_transpuesta[1]
print(x_lista_transpuesta)
print(y_lista_transpuesta)



#MATRIZ TRANSPUESTA (NUMPY):Forma rápida de sacar los valores xy con la librería numpy.
#En el primer parámetro se indica la lista que se quiere convertir a un arreglo de numpy y en el segundo parámetro
#se indica a que tipo de dato primitivo se quiere convertir los datos.
data_numpy = np.array(data_str,dtype=float)
data_final = data_numpy.T #Transpuesta de la matriz data_numpy
x_numpy = data_final[0]
y_numpy = data_final[1]
print("Matriz original: \n", data_numpy)
print("Matriz transpuesta: \n", data_final)
print("Vector x de la matriz transpuesta: \n",x_numpy)
print("Vector y de la matriz transpuesta: \n",y_numpy)



#GRAFICAR LOS DATOS EN PYTHON:
import matplotlib.pyplot as plt #matplotlib: Librería de graficación matemática.
#matplotlib.figure(): Método usado para crear la ventana de graficación (objeto pyplot).
fig = plt.figure()#Creación del objeto pyplot.
#matplotlib.axes(): Método usado para crear la rejilla en la ventana de graficación.
ax = plt.axes()#Se crea un objeto ax perteneciente de la clase axes().
#axes.plot(): Método usado para crear la gráfica en la rejilla previamente creada en la ventana de graficación.
ax.plot(x_numpy, y_numpy, 'C1o--')#'C1o--' significa C1: color naranja, o: simbolo de círculos --: líneas punteadas.
#axes.set_xlabel(): Método para indicar el texto que aparece en el eje x.
ax.set_xlabel(r'$x$')#Texto que aparece en el eje horizontal
#axes.set_ylabel(): Método para indicar el texto que aparece en el eje y.
ax.set_ylabel(r'$y=x^(2)$')#Texto que aparece en el eje vertical
#matplotlib.show(): Método para mostrar la gráfica creada.
plt.show()
```

Código simplificado:

```
# -*- coding: utf-8 -*-
"""
@author: JFC
Programa que genera un archivo con formato csv
con los datos de una parábola, y posteriomente
los grafica.
"""
import matplotlib.pylab as plt

#Vectores x y y
x=[]
y=[]
for i in range(0,11):
    x.append(i)
    y.append(i*i)
#Nombre del archivo en donde se guardan los datos generados
fileName='datosXY.txt'
#Escritura de los vectores anteriores en un archivo en formato csv
myFile=open(fileName,'w') #w=write
for i in range(0,len(x)):
    myFile.write(str(x[i])+","+str(y[i])+"\n")
myFile.close()

#Lectura de los datos
myFile=open(fileName,'r')
numRows=0
list1=[]
for line in myFile:
    line=line.replace("\n","")#Remueve los saltos de línea
    line=line.replace(" ","")#Remueve los espacios en blanco
    list1.append(line)
    numRows+=1
myFile.close()
#Lista con los índices en donde se ubican las comas que separan los datos
commaIndices=[]
for i in range(0,numRows):
    commaIndices.append(list1[i].find(','))
#Extracción de los valores de x y y
xRead=[]
yRead=[]
for i in range(0,numRows):
    xRead.append(float(list1[i][:commaIndices[i]]))
    yRead.append(float(list1[i][commaIndices[i]+1:]))
#Graficación de los valores
plt.figure(1)
plt.title(r'$y=x^2$',fontsize=20)
plt.xlabel(r'$x$',fontsize=18)
plt.ylabel(r'$y$', fontsize=18)
plt.xlim(-0.2,10.2)
plt.ylim(-2,102)
plt.plot(xRead,yRead,'or')
plt.grid(True)

```

Código C# (.Net Framework):

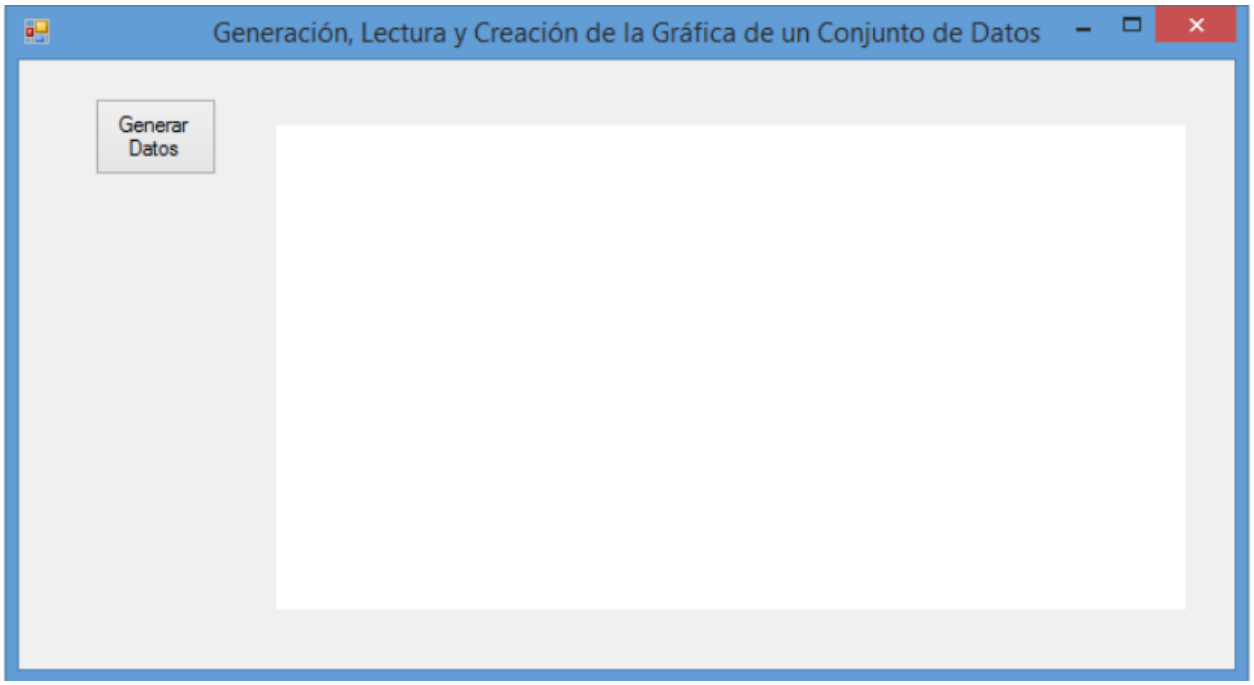
Código con resultado en GUI de Visual Studio .Net Framework:

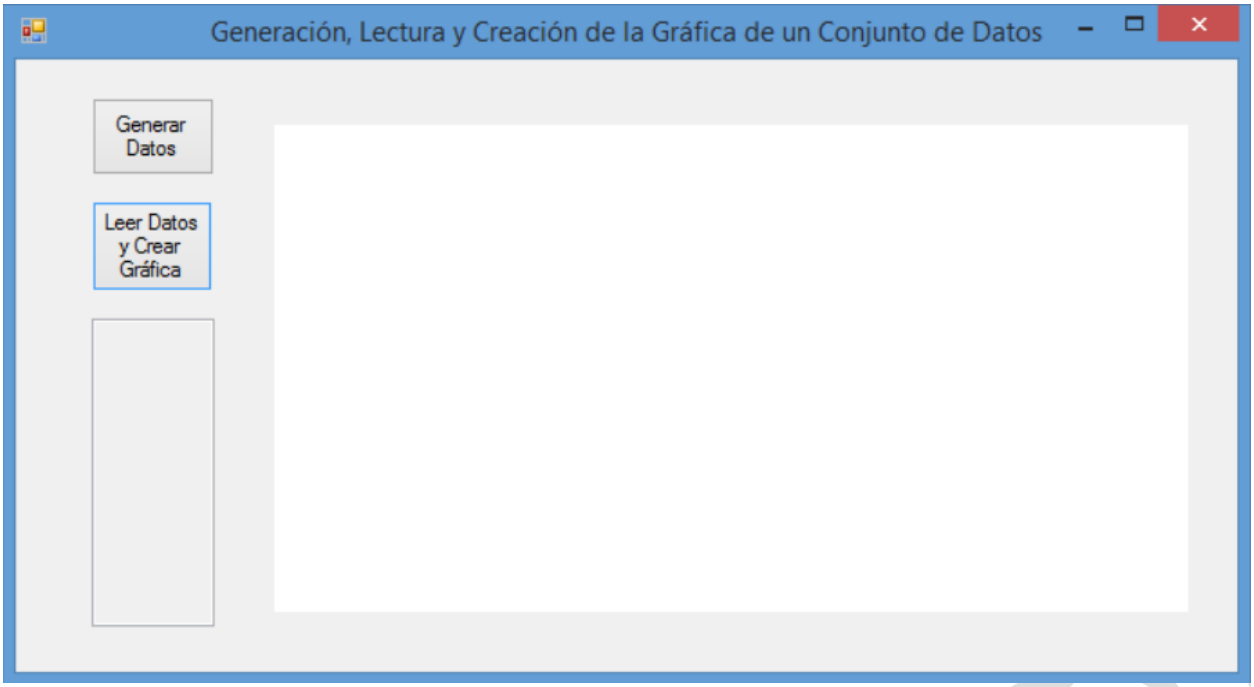
```
11  //Agregar referencia
12  using System.IO;
13
14  namespace LecturayGraficacionDatos
15  {
16      public partial class Form1 : Form
17      {
18          /*Programa que genera los datos de una parábola, los guarda
19           * en un archivo en la ruta que indica el usuario, los lee
20           * con la clase OpenFileDialog, y, finalmente, los grafica.*/
21
22          public Form1()
23          {
24              InitializeComponent();
25          }//Constructor Form1
26
27          /*Método que se usa para generar el vector de tipo
28           * string con los datos en formato CSV*/
29          private string[] GenerateData()
30          {
31              int [] x = new int[11];
32              int[] y = new int[x.Length];
33              string[] stDataCSV = new string[x.Length];
34              for (int i = 0; i < x.Length; i++)
35              {
36                  x[i] = i;
37                  y[i] = x[i] * x[i];
38                  stDataCSV[i] = Convert.ToString(x[i]) + "," + Convert.ToString(y
                      [i]);
39              }//For
40              return stDataCSV;
41          }//Fin del método GenerateData
42
43          //Método que se usa para graficar los datos.
44          private void PlotGraphic(int nSamples, double[] xPlot, double[] yPlot)
45          {
46              for (int i = 0; i < nSamples; i++)
47              {
48                  Plt1.Series["xy"].Points.AddXY(xPlot[i], yPlot[i]);
49                  Plt1.Series["xy"].Color = Color.Red;
50              }//For
51          }//Fin del método PlotGraphic
```

```
53          private void bt1_GenerateData_Click(object sender, EventArgs e)
54          {
55              //Revisa la clase SaveFileDialog en:
56              //https://msdn.microsoft.com/es-es/library/
                  system.windows.forms.savefiledialog(v=vs.110).aspx
57              Stream myStream;
58              SaveFileDialog saveFileDialog1 = new SaveFileDialog();
59
60              //saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|
                  *.*";
61              //saveFileDialog1.FilterIndex = 2;
62              saveFileDialog1.RestoreDirectory = true;
63
64              if (saveFileDialog1.ShowDialog() == DialogResult.OK)
65              {
66                  if ((myStream = saveFileDialog1.OpenFile()) != null)
67                  {
68                      using (StreamWriter outputFile = new StreamWriter(myStream))
69                      {
70                          foreach (string line in GenerateData())
71                              outputFile.WriteLine(line);
72                      }//Using
73                      myStream.Close();
74                  }//2ndo. if
75              }//1er. if
76              /*Muestra el botón que se usa en la lectura de los datos
77               * y el cuadro de texto, en donde se despliegan los datos leídos.*/
78              bt2_GetData.Visible = true;
79              txt1_Data.Visible = true;
80          }//Fin de bt1_GenerateData_Click
81
82          private void bt2_GetData_Click(object sender, EventArgs e)
83          {
84              //Revisar página:
85              //https://msdn.microsoft.com/es-es/library/cc221415(v=vs.95).aspx
86
87              OpenFileDialog openFileDialog1 = new OpenFileDialog();
88              // openFileDialog1.Filter = "Text Files (.txt)|*.txt|All Files (*.*)|
                    *.*";
89              // openFileDialog1.FilterIndex = 1;
90              openFileDialog1.Multiselect = false;
91              if (openFileDialog1.ShowDialog() ==
                    System.Windows.Forms.DialogResult.OK)
92              {
93                  string xstr;
94                  string ystr;
95                  double[] x = new double[0];
96                  double[] y = new double[0];
97                  int counter = 0;
98                  string line = "";
99                  int index = 0;
```

```
100                 txt1_Data.Text = "";
101                 Plt1.Series.Clear(); //Limpieza de la gráfica;
102                 //Genera una nueva serie llamada xy
103                 Plt1.Series.Add("xy");
104                 //Define el tipo de gráfica (puntos)
105                 Plt1.Series["xy"].ChartType =
106 System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
107                 //Abre el archivo que se seleccionó para su lectura.
108                 System.IO.Stream fileStreamIn = openFileDialog1.OpenFile();
109                 using (System.IO.StreamReader reader = new System.IO.StreamReader
                      (fileStreamIn))
110                 {
111                     /*Ciclo que lee todas las líneas del archivo
112                      * hasta que no encuentra caracteres*/
113                     while ((line = reader.ReadLine()) != null)
114                     {
115                         /* Método que encuentra la posición en que
116                          * se localiza la coma en cada línea*/
117                         index = line.IndexOf(",");
118                         //Genera un string con los caracteres antes de la coma
                        (Valor x[i])
119                         xstr = line.Substring(0, index);
120                         //Guarda los datos de x[i] en un arreglo tipo doble
121                         Array.Resize(ref x, x.Length + 1);
122                         x[x.Length - 1] = Convert.ToDouble(xstr);
123                         //Repite un proceso similar para obtener los valores y[i]
124                         ystr = line.Substring(index + 1);
125                         Array.Resize(ref y, y.Length + 1);
126                         y[y.Length - 1] = Convert.ToDouble(ystr);
127                         //Fije la propiedad multiline de la caja de texto como
                        true.
128                         txt1_Data.Text += Convert.ToString(x[x.Length - 1]) + " "
                         + Convert.ToString(y[y.Length - 1]) + "\r\n";
129                         counter++;
130                     }//While
131                 } //Using
132                 fileStreamIn.Close();//Se cierra el archivo
133                 //Método que grafica
134                 PlotGraphic(x.Length, x, y);
135             } //If
136         }//Fin de bt2_GetData_Click
137
138         private void Form1_Load(object sender, EventArgs e)
139         {
140             bt1_GenerateData.Visible = true;
141             bt2_GetData.Visible = false;
142             txt1_Data.Visible = false;
143         }//Fin de Form1_Load
144     } //Fin de la clase Form
145 } //Fin del espacio de nombres LecturayGraficacionDatos
```

En C# se usó un formulario de Windows (Windows Forms) en la solución de este ejercicio. En la primera imagen se muestra la pantalla inicial que ve un usuario cuando se inicia el programa. Cuando se oprime el botón generar datos, se agregan nuevos elementos al programa como se observa en la segunda ventana y en la tercera imagen se muestra la gráfica de los datos leídos, la cual se obtiene cuando se presiona el botón leer datos y crear gráfica.





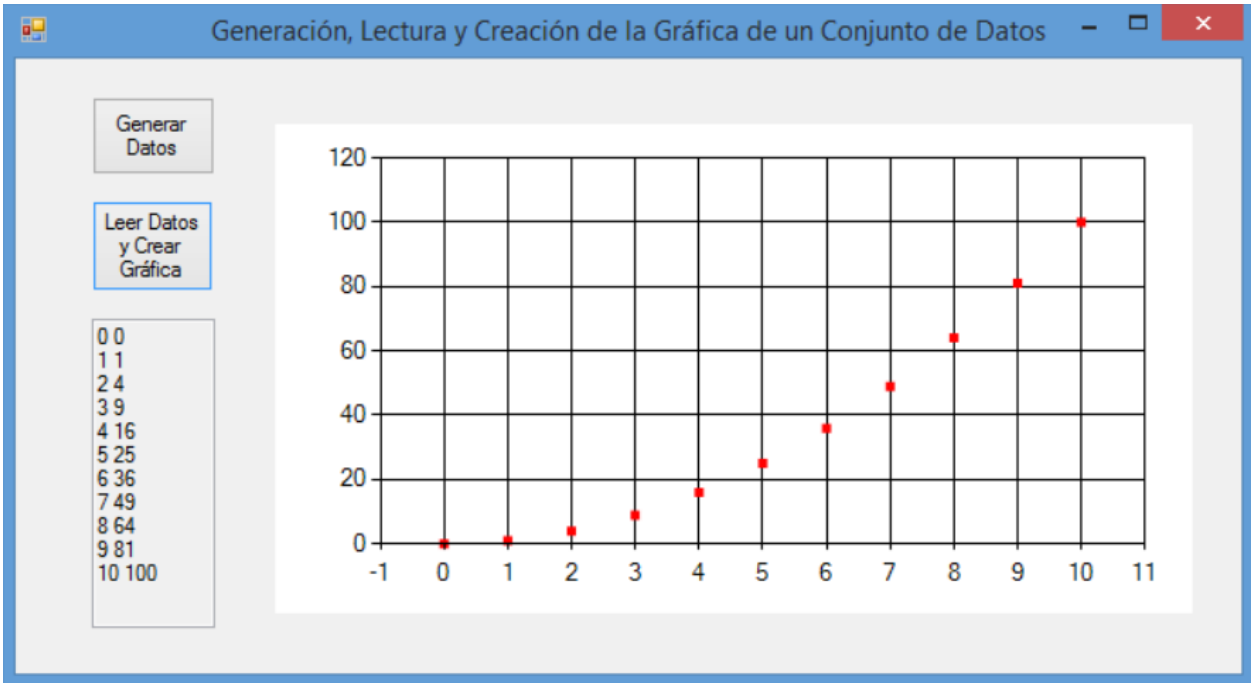


Diagrama LabVIEW: