

Lienzo Virtual

Una figura como la que se muestra a continuación sirve de "lienzo" para representar dibujos. Un conjunto de puntos vecinos en dicha matriz se considerará un dibujo. El problema consiste en implementar un "lienzo virtual" (es decir, sin límite de tamaño) para representar potencialmente cualquier cantidad de dibujos.

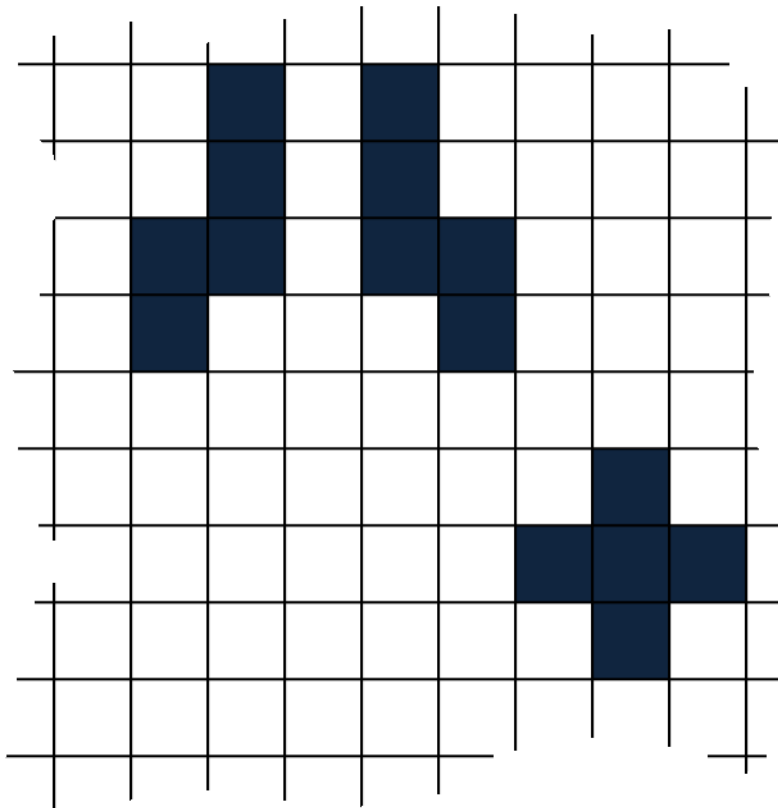


Figura 1- Ejemplo de lienzo con 3 dibujos

Los dibujos en el lienzo se forman añadiendo puntos. Dos puntos (x_1, y_1) y (x_2, y_2) se consideran vecinos si son adyacentes es decir si cumplen que $|x_1 - x_2| \leq 1$ y $|y_1 - y_2| \leq 1$.

Usted debe implementar la siguiente clase:

```

class Lienzo
{
    public void AdicionaPuntos(params Point[] puntos)
    {
        ...
    }

    public IEnumerable<Point> Dibujo(Point p)
    {
        ...
    }

    public IEnumerable<IEnumerable<Point>> Dibujos()
    {
        ...
    }
}

```

El método `AdicionaPuntos` añade puntos al Lienzo (note que pueden ser varios, uno o ninguno).

El método `IEnumerable<Point> Dibujo(Point p)` devuelve un `IEnumerable<Point>` con los puntos que conforman el dibujo al que pertenece el punto `p` (este enumerable puede ser vacío si el punto no está en el lienzo). Note que un punto solo puede formar parte de un dibujo.

El método `Dibujos` debe retornar un enumerable de todos los dibujos que hay en el lienzo, es decir un `IEnumerable<IEnumerable<Point>>`.

Ejemplo

```

Lienzo lienzo = new Lienzo();

//dibujo 1
lienzo.AdicionaPuntos(new Point(5, 7), new Point(5, 8), new Point(5, 9),
    new Point(4, 9), new Point(4, 10));

//dibujo 2
lienzo.AdicionaPuntos(new Point(7, 8), new Point(7, 9));
lienzo.AdicionaPuntos(new Point(8, 9), new Point(8, 10));

//dibujo 3
lienzo.AdicionaPuntos(new Point(10, 12), new Point(10, 13), new Point(10, 14),
    new Point(9, 13), new Point(11, 13));

```

Tras ejecutar el código anterior el lienzo quedaría así:

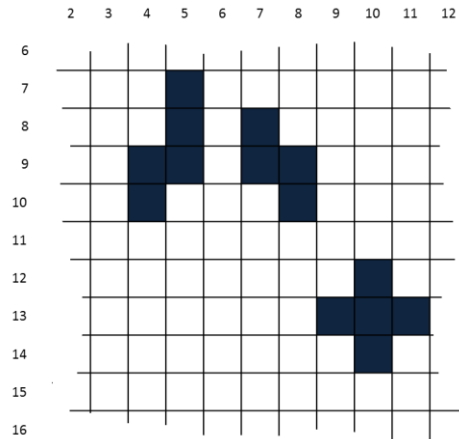


Figura 2 - Ejemplo de lienzo después de agregar una serie de puntos

Como puede apreciarse en el lienzo de la figura 2 existen 3 dibujos. El código siguiente itera por los tres dibujos existentes:

```
int i = 1;
foreach (IEnumerable<Point> dibujo in lienzo.Dibujos())
{
    Console.WriteLine("Figura {0}:", i++);
    foreach (Point p in dibujo)
    {
        Console.WriteLine("{0};{1}", p.X, p.Y);
    }
}
```

```
C:\Windows\system32\cmd.exe
Figura 1:
(5;7)
(5;8)
(5;9)
(4;9)
(4;10)
Figura 2:
(7;8)
(7;9)
(8;9)
(8;10)
Figura 3:
(10;12)
(10;13)
(10;14)
(9;13)
(11;13)
```

Figura 3 - Resultado tras iterar por los dibujos del lienzo de la figura 2

Otra opción sería a partir de un punto, iterar por el dibujo al que este pertenece. El código a continuación obtiene todos los puntos del dibujo al que pertenece el punto (5,8):

```

Console.WriteLine("Puntos del dibujo que contiene a (5,8):");
foreach (Point p in lienzo.Dibujo(new Point(5,8)))
{
    Console.WriteLine("{0};{1}", p.X, p.Y);
}

```

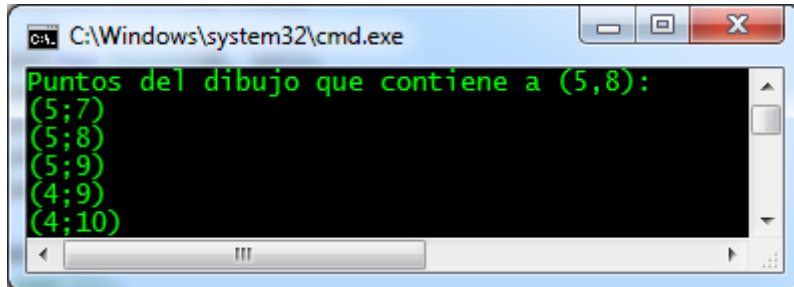


Figura 4 - Resultado tras iterar por los puntos del dibujo que contiene a (5,8)

Ahora bien, si se agregara el punto (6, 8) los tres dibujos anteriores se convertirían en sólo dos, y si se insertara además el punto (9,11) pues todos los puntos quedarían conectados entre sí y sólo habría un dibujo sobre el lienzo.

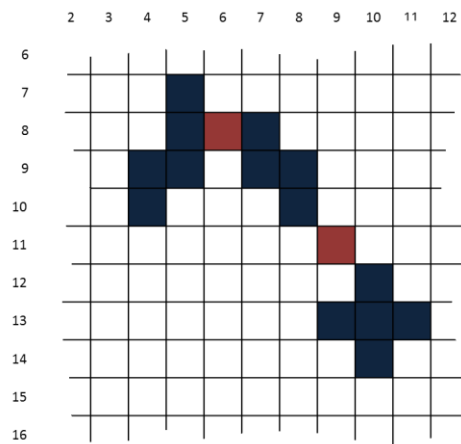
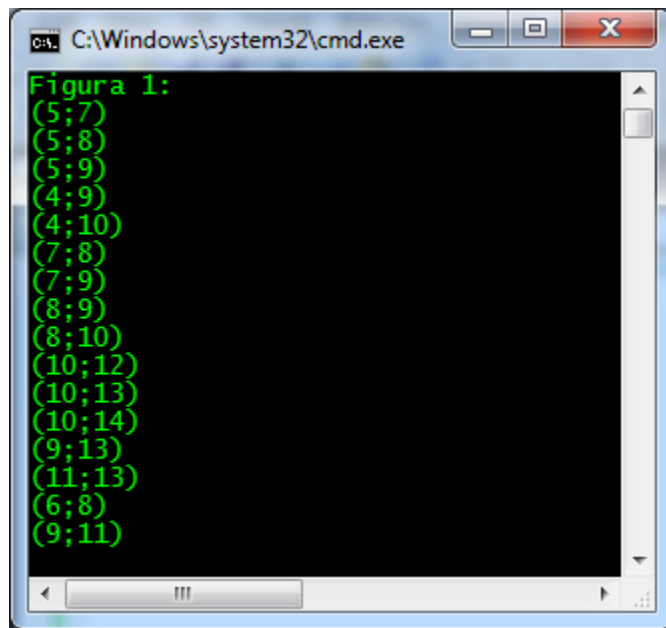


Figura 5 - Estado del lienzo después de agregar los puntos (6,8) y (9,11)

Si volviésemos a ejecutar el código que itera por los dibujos del gráfico el resultado sería el siguiente:



```
C:\Windows\system32\cmd.exe
Figura 1:
(5;7)
(5;8)
(5;9)
(4;9)
(4;10)
(7;8)
(7;9)
(8;9)
(8;10)
(10;12)
(10;13)
(10;14)
(9;13)
(11;13)
(6;8)
(9;11)
```