Práctica 3. Uso de unidades y TADs
Estructuras de Datos

Objetivo:
Vamos a continuar con el uso y disfrute de unidades de Pascal para importar funcionalidad implementada anteriormente.

Funcionalidad básica del sistema gráfico:
En esta ocasión vamos a utilizar la unidad `Graph` (en algunas plataformas, como linux, se denomina `Ptcgraph`) que implementa funcionalidad para manejar gráficos en pantalla. Más información sobre esta unidad está disponible en http://www.freepascal.org/docs-html-3.0.0/rtl/graph/index.html

En particular, vamos a pintar píxeles de colores en pantalla para terminar creando el fractal de Mandelbrot mediante nuestra unidad de números complejos de la práctica anterior. Como comienzo vamos a probar el uso de la unidad `Graph` mediante un pequeño programa que muestre una gama de los 64 primeros tonos disponibles en el adaptador gráfico que carguemos en nuestro programa.

```
program PruebaPantalla;
uses graph;
const
  MAX_ITER  = 64;
  MAX_RES_X = 640;
  MAX_RES_Y = 480; {No lo usamos ahora}
var
   grdriver, grmode: smallint; {en algunas plataformas podrá ser integer}
   i, j, color: integer;
begin
   grdriver := D8bit;
   grmode := m1024x768;
   initgraph(grdriver, grmode,''); {Inicializa el sistema gráfico}
   for i:=0 to MAX_RES_X-1 do begin {anchura en número de píxeles}
       color := i div (MAX_RES_X div MAX_ITER)+1;
       for j:=0 to 99 do {altura}
           putpixel(i, j, color); {Pinta en posición (i,j) de pantalla un
                                   pixel con un color dado como tercer argumento}
   end;
   readln;
   closegraph {cierra el sistema gráfico iniciado}
end.
```

Figura 1: Ejemplo de uso de la librería `Graph` (o `Ptcgraph`).

El resultado en pantalla debería ser la apertura de una nueva ventana del sistema gráfico en la que se mostrará una paleta de 64 colores. Ahora puedes cambiar la paleta de colores y jugar con los parámetros de estas instrucciones para familiarizarte con el entorno gráfico (de hecho no sería difícil que hicieras un gráfico simple, *sprite*, mediante un subprograma y pudieras pintarlo donde quisieras a modo de videojuego "*old-school*").

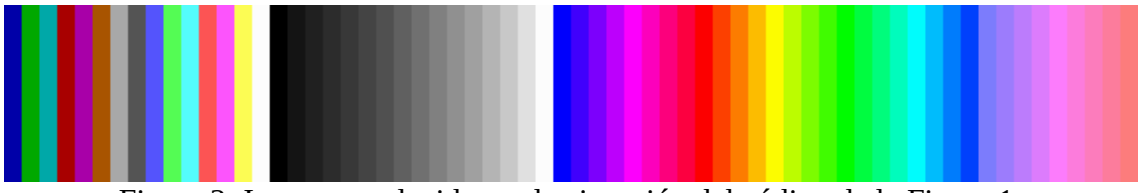

Figura 2: Imagen producida por la ejecución del código de la Figura 1.

Si algo no ha funcionado, seguramente sea que la unidad `Graph` no está disponible en tu sistema (en los equipos virtualizados no está accesible por defecto). Si eso es así, prueba a conseguir la unidad `Graph` (o `Ptcgraph`), guardarla junto con el código fuente de este pequeño programa en una carpeta a la que tengas acceso desde un explorador de archivos, y cambia el espacio de trabajo del IDE Eclipse mediante el menú *File/Switch Workspace/Other*… eligiendo la nueva carpeta donde guardaste la unidad `Graph` (o `Ptcgraph`) y el código del programa. Si aun así tienes problemas te recomendamos que trabajes en local en una máquina propia y no virtualizada.

Conjunto de Mandelbrot:
El conjunto de Mandelbrot está considerado como el objeto geométrico más complicado creado hasta el momento por el hombre. La frontera que delimita este objeto en el plano complejo es *fractal* (patrón que se repite en cada escala). Su nombre se debe al matemático *Benoît Mandelbrot* (1924-2010), que investigó sobre él en los años setenta.

El conjunto de Mandelbrot se puede definir en el plano complejo de la siguiente manera. Llamemos c a un número complejo cualquiera. Con ese valor de c, se construye la siguiente sucesión:

$$\begin{Bmatrix} z_0 = 0 & n = 0 \\ z_{n+1} = z_n^2 + c & n > 0 \end{Bmatrix}$$

donde n hace referencia al **número de iteración**. Veamos algunos ejemplos:
- Supongamos que $c = 1 + 0i = 1$. Para ese caso, los primeros términos de la sucesión de Mandelbrot son:

$$z_0 = 0$$
$$z_1 = z_0^2 + c = 0 + 1 = 1$$
$$z_2 = z_1^2 + c = 1^2 + 1 = 2$$
$$z_3 = z_2^2 + c = 2^2 + 1 = 5$$
$$z_4 = z_3^2 + c = 5^2 + 1 = 26$$
$$\dots$$

- Para $c = -1 + 0i = -1$, la sucesión de Mandelbrot es:

$$z_0 = 0$$
$$z_1 = z_0^2 + c = 0 - 1 = -1$$
$$z_2 = z_1^2 + c = (-1)^2 - 1 = 0$$
$$z_3 = z_2^2 + c = 0^2 - 1 = -1$$
$$z_4 = z_3^2 + c = (-1)^2 - 1 = 0$$
$$\dots$$

- La sucesión de Mandelbrot para $c = -0.1 + 0.1i$ es:

$$z_0 = 0$$
$$z_1 = z_0^2 + c = 0^2 - 0.1 + 0.1i = -0.1 + 0.1i$$
$$z_2 = z_1^2 + c = (-0.1 + 0.1i)^2 - 0.1 + 0.1i = -0.1 + 0.08i$$
$$z_3 = z_2^2 + c = (-0.1 + 0.08i)^2 - 0.1 + 0.1i = -0.0964 + 0.084i$$
$$z_4 = z_3^2 + c = (-0.0964 + 0.084i)^2 - 0.1 + 0.1i = -0.097763 + 0.083805i$$
$$\dots$$

Si esta sucesión queda acotada, entonces se dice que c pertenece al conjunto de Mandelbrot, y si no, c no pertenece. Como se puede intuir en los ejemplos anteriores, $c = 1 + 0i$ no es un elemento del conjunto de Mandelbrot, puesto que su sucesión no parece estar acotada, mientras que $c = -1 + 0i$ y $c = -0.1 + 0.1i$ sí pertenecen al conjunto de Mandelbrot, ya que sus sucesiones sí parecen estar acotadas.

A menudo se visualiza el conjunto mediante el algoritmo de tiempo de escape, que asigna un color a cada punto complejo en el espacio 2D en función de la iteración en la que se detecta su divergencia.

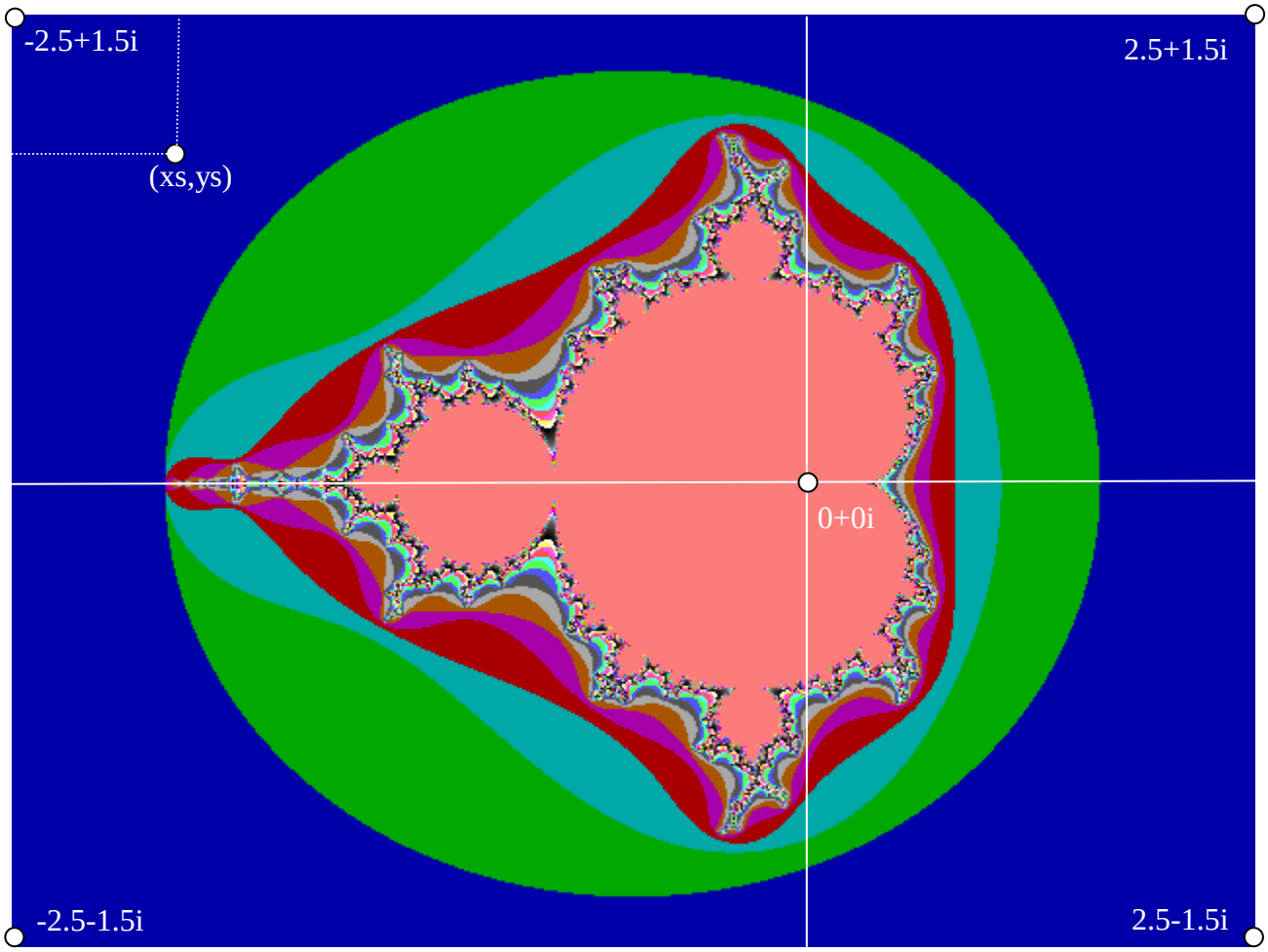

Figura 3: Conjunto de Mandelbrot para los números complejos en el rango [-2.5-1.5i, 1.5+1.5i].

En la imagen mostrada en la Figura 3 se representa dicho tiempo de escape con un color diferente en cada punto del plano complejo 2D. Como se puede observar, aparecen regiones de colores diferentes, que indican que los tiempos de escape son diferentes en cada caso. Por ejemplo, la región circular de números coloreada de verde diverge al cabo de pocas iteraciones, mientras que los puntos de la región central coloreados de salmón no presentan una sucesión divergente tras calcular 64 términos de la sucesión.

¿Cómo saber entonces si una sucesión es divergente? Se sabe que los puntos del plano complejo cuyo módulo es superior a 2 no pertenecen al conjunto de Mandelbrot. Por lo tanto, basta encontrar un solo término de la sucesión que verifique que $|z_n| > 2$ para estar seguro de que c no está en el conjunto. Por tanto, si el valor del módulo pasada alguna iteración de cálculo es mayor que 2 se le asigna el número de iteración al color del punto.

Y ese va a ser el objetivo de esta práctica. Utilizando la unidad de los números complejos ya realizada en clase y las indicaciones dadas en este enunciado, construir un programa que represente en una imagen de 640x480 píxeles la velocidad de divergencia para cada punto del plano complejo entre los valores $[-2.5 - 1.5i, 1.5 + 1.5i]$. Como queremos dibujar en el rango de valores $[-2.5,1.5]$ en el eje real (4 unidades de anchura), y $[-1.5i, 1.5i]$ en el eje imaginario (3 unidades de altura), tenemos que asociar a cada píxel el número complejo que va a representar. Sabiendo que el

origen de coordenadas de la imagen (0,0) es la esquina superior izquierda ($-2.5 + 1.5i$), tenemos que a cada píxel de la imagen en posición $[xs, ys]$ le corresponde el número complejo $\left(\frac{xs}{160} - 2.5\right) + \left(\frac{ys}{160} - 1.5\right) i$ (considerando la proporción 640/4=160 y 480/3=160). A continuación, hay que calcular la velocidad de escape para cada uno de esos números complejos. Esto se hace calculando su sucesión hasta llegar al número máximo de iteraciones predefinido (nosotros calcularemos un máximo de 64 iteraciones) o hasta comprobar que diverge (si alguno de los términos calculados tiene un módulo mayor que 2, esa sucesión es divergente). Finalmente, se colorea el píxel correspondiente al valor complejo con el valor de la máxima iteración alcanzada.