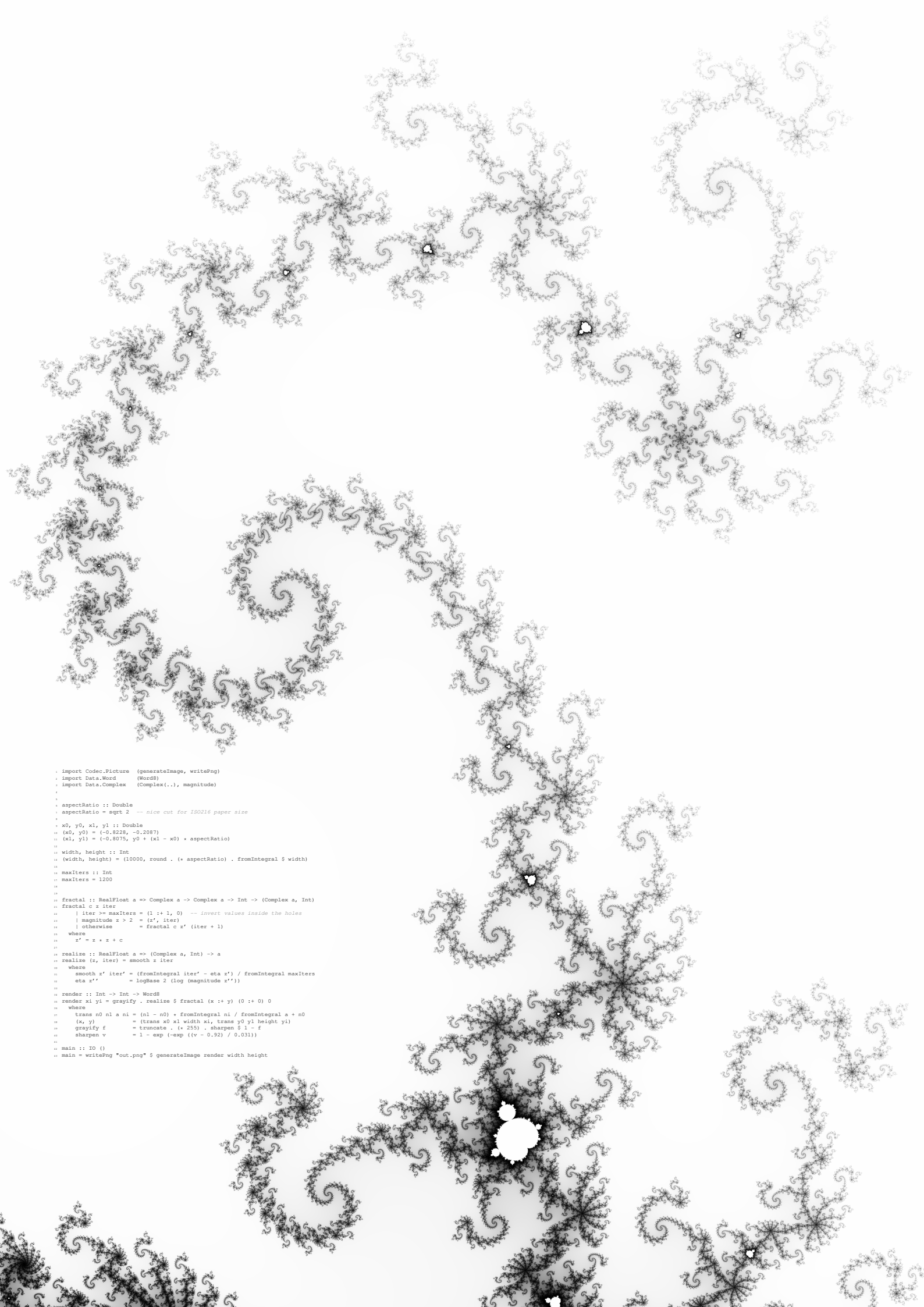

```haskell
import Codec.Picture  (generateImage, writePng)
import Data.Word      (Word8)
import Data.Complex   (Complex(..), magnitude)


aspectRatio :: Double
aspectRatio = sqrt 2  -- nice cut for ISO216 paper size

x0, y0, x1, y1 :: Double
(x0, y0) = (-0.8228, -0.2087)
(x1, y1) = (-0.8075, y0 + (x1 - x0) * aspectRatio)

width, height :: Int
(width, height) = (10000, round . (* aspectRatio) . fromIntegral $ width)

maxIters :: Int
maxIters = 1200


fractal :: RealFloat a => Complex a -> Complex a -> Int -> (Complex a, Int)
fractal c z iter
    | iter >= maxIters = (1 :+ 1, 0)  -- invert values inside the holes
    | magnitude z > 2  = (z', iter)
    | otherwise        = fractal c z' (iter + 1)
  where
    z' = z * z + c

realize :: RealFloat a => (Complex a, Int) -> a
realize (z, iter) = smooth z iter
  where
    smooth z' iter' = (fromIntegral iter' - eta z') / fromIntegral maxIters
    eta z''         = logBase 2 (log (magnitude z''))

render :: Int -> Int -> Word8
render xi yi = grayify . realize $ fractal (x :+ y) (0 :+ 0) 0
  where
    trans n0 n1 a ni = (n1 - n0) * fromIntegral ni / fromIntegral a + n0
    (x, y)           = (trans x0 x1 width xi, trans y0 y1 height yi)
    grayify f        = truncate . (* 255) . sharpen $ 1 - f
    sharpen v        = 1 - exp (-exp ((v - 0.92) / 0.031))

main :: IO ()
main = writePng "out.png" $ generateImage render width height
```