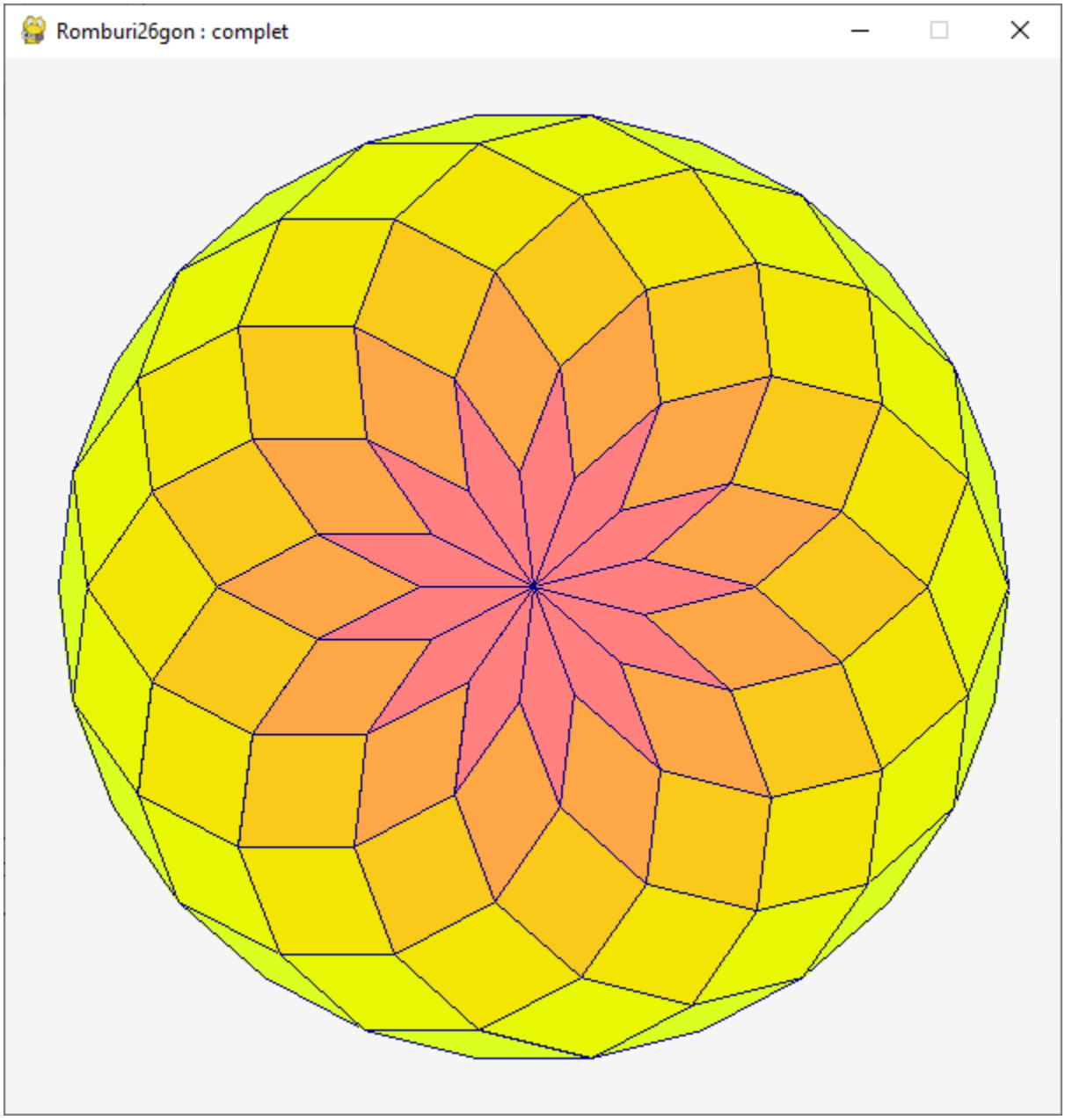
Curs 09

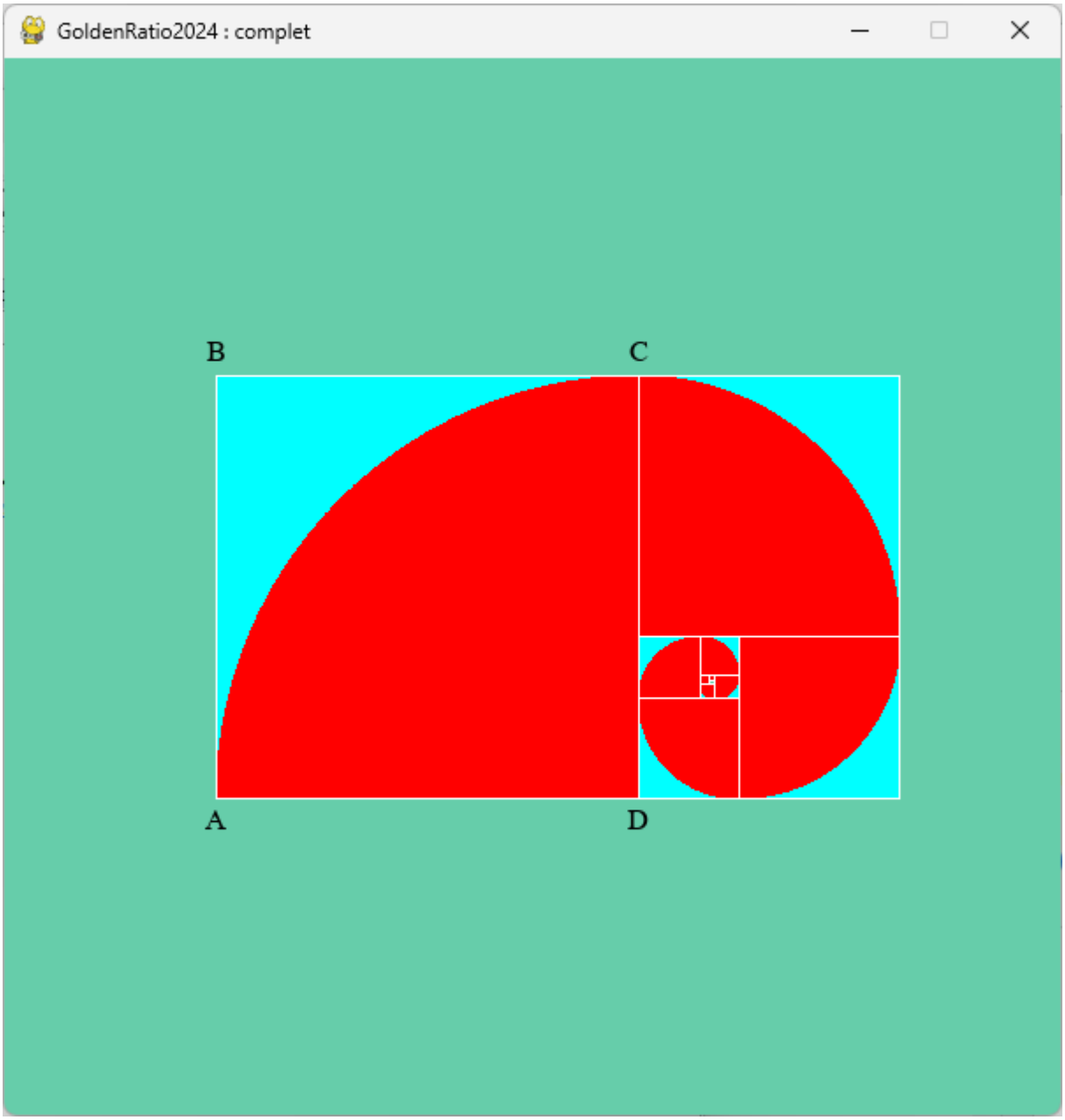
(*plan de curs*)

1. Tema 8, Exercițiul 9a



```
import ComplexPygame as C
import Color
import math

def Romburi26gon():
    def deseneazaRomb(p, kol):
        C.fillNgon(p, Color.Index(kol))
        C.drawNgon(p, Color.Navy)

    C.setXminXmaxYminYmax(-10, 10, -10, 10)
    C.fillScreen(Color.Mediumaquamarine)
    R = 9
    N = 13
    # plecam de la 2N-gon-ul exterior
    omega = math.pi / N
    A = [C.fromRhoTheta(R, k * omega) for k in range(2 * N + 1)]

    for nr in range(6):
        for nn in range(0, 2 * N, 2):
            aux = A[nn] - A[nn + 1] + A[nn + 2]
            deseneazaRomb([A[nn], A[nn + 1], A[nn + 2], aux], 200 + 25 * nr)
            A[nn + 1] = aux
        # rotim lista
        del A[0]
        A.append(A[0])
        if C.mustClose():
            return
        C.wait(100)

if __name__ == '__main__':
    C.initPygame()
    C.run(Romburi26gon)
```

2. Raportul de aur




```
import ComplexPygame as C
import Color
import math
```

```
def GoldenRatio2024():
    fi = (1 + math.sqrt(5.0)) / 2
    omegaCDprimPeAD = -1j / fi

    def traseazaSiTransforma(sector):
        a = sector[0]
        d = sector[- 1]
        c = sector[-2]
        b = a + c - d
        dprim = c + (d - c) / fi
        C.fillNgon([a, b, c, d], Color.Aqua)
        C.fillNgon(sector, Color.Red)
        C.drawNgon([a, b, c, d], Color.White)
        return [dprim + omegaCDprimPeAD * (z - d) for z in sector]

    C.setXminXmaxYminYmax(-0.5, 2, -0.75, 1.75)
    C.fillScreen(Color.Mediumaquamarine)
    a = 0
    b = 1j
    c = 1 + 1j
    d = 1
    C.setText("A", a - 0.1j)
    C.setText("B", b + 0.01j)
    C.setText("C", c + 0.01j)
    C.setText("D", d - 0.1j)
    nrPuncte = 1000
    alfa = -math.pi / (2 * nrPuncte)
    sector = [d + C.fromRhoTheta(1, n * alfa) * (a - d) for n in range(nrPuncte)]
    sector.append(d)
    for k in range(10):
        sector = traseazaSiTransforma(sector)
        C.refreshScreen()
        C.wait(100)



if __name__ == '__main__':
    C.initPygame()
    # C.run(Romburi26gon)
    C.run(GoldenRatio2024)
```

3. Teorema reziduurilor

```
def Reziduuri():

    a, b = 0, 2 * math.pi
    p1 = 3 + 1j
    p2 = -1 + 3j

    def g(z):
        return z * z

    def f(z):
        if (z - p1) * (z - p2) == 0:
            return 0
        else:
            return g(z) / ((z - p1) * (z - p2))

    # def cercQr(q, r):
    #     return lambda t: q + C.fromRhoTheta(r, t)

    def cercQR(q, r):
        def gamma(t):
            return q + C.fromRhoTheta(r, t)

        return gamma

    def arataModulul(F):
        for z in C.screenAffixes():
            k = int(10 * C.rho(F(z)))
            C.setPixel(z, Color.Index(k))

    def arataIntegrala(F, Gamma):
        nrPasi = 5000
        dt = (b - a) / nrPasi
        # calculam suma Riemann-Stieltjes
        suma = 0
        z0 = Gamma(a)
        for k in range(1, nrPasi):
            z1 = Gamma(a + k * dt)
            suma += F(z1) * (z1 - z0)
            col = Color.Index(300 + k // 10)
            C.drawLine(0, suma, col)
            C.setPixel(z1, col)
            z0 = z1
            C.refreshScreen()
        return suma

    lat = 30
    C.setXminXmaxYminYmax(-lat, lat, -lat, lat)
    arataModulul(f)
    C.setAxis()
    q0 = 1 + 1j
    r0 = 5
    intRS = arataIntegrala(f, cercQR(q0, r0))
    sumaReziduuri = 0
```

```
if C.rho(p1 - q0) < r0:
    sumaReziduuri += g(p1) / (p1 - p2)
if C.rho(p2 - q0) < r0:
    sumaReziduuri += g(p2) / (p2 - p1)

intTR = 2j * math.pi * sumaReziduuri
C.setTextIJ(f"intRS = {intRS}", 10, 20, Color.White)
C.setTextIJ(f"intTR = {intTR}", 10, 40, Color.White)
```