Curs 04

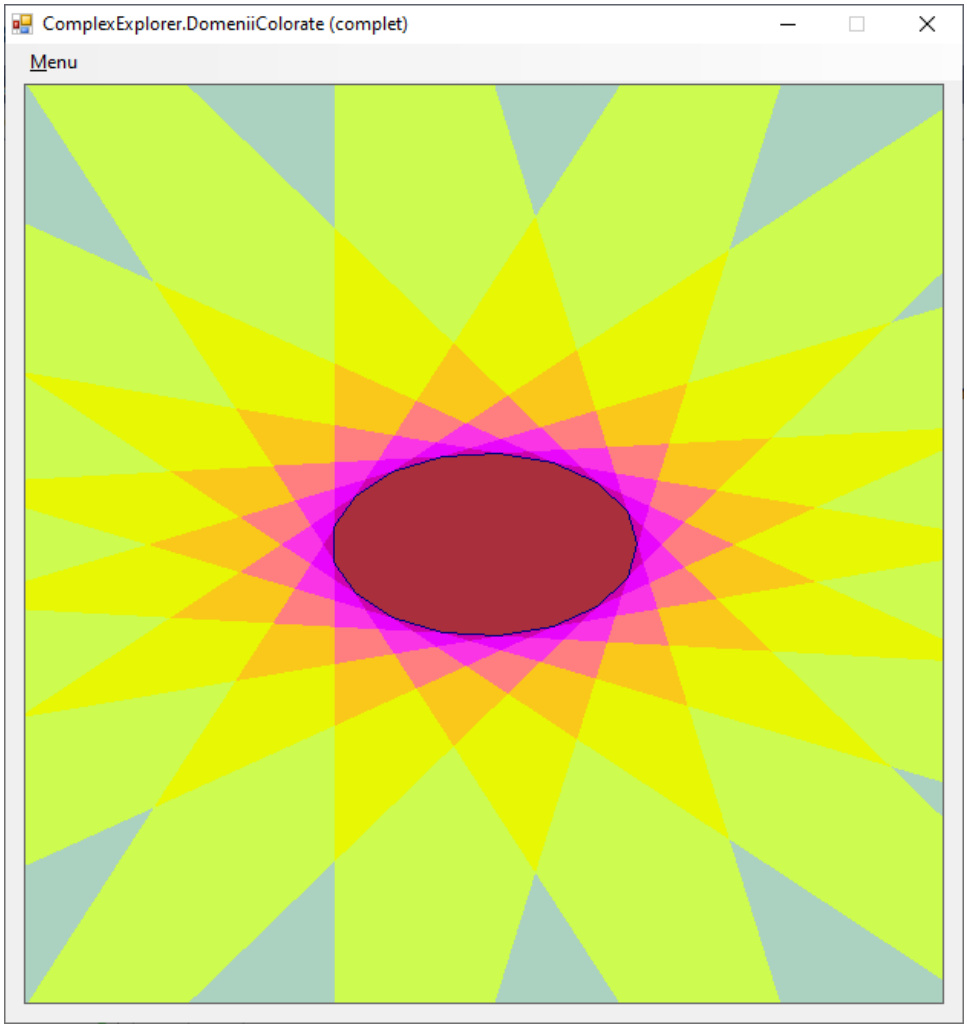
(*plan de curs*)

1. Argumentul unui număr complex, unghiul a două laturi

```
public double Theta {
      get {
          return Math.Atan2(y, x);
      }
 }
```

Semiplane, interiorul unui poligon convex, interiorul unei curbe Jordan.

2. Topologia planului complex. Şirul puterilor unui număr complex, seria geometrică, curbe plane.




```
using System;
using System.Drawing;
using static System.Math;
```

```
namespace ComplexExplorer
{
    public class DomeniiColorate : ComplexForm
    {
        const double doiPI = 2.0 * PI;

        bool EsteInStanga(Complex a, Complex b, Complex z)
        {
            return ((z - a) / (b - a)).Theta >= 0;
        }

        void TraseazaContur(Complex[] p, Color col)
        {
            for (int k = 1; k < p.Length; k++)
            {
                setLine(p[k - 1], p[k], col);
            }
            resetScreen();
        }

        void ColoreazaSemiplane(Complex[] p)
        {
            for (int ii = 0; ii <= imax; ii++)
            {
                for (int jj = 0; jj <= jmax; jj++)
                {
                    Complex z = getZ(ii, jj);
                    int niv = 0;
                    for (int k = 1; k < p.Length; k++)
                    {
                        if (EsteInStanga(p[k - 1], p[k], z)) niv++;
                    }
                    setPixel(ii, jj, getColor(700 + 50 * niv));
                }
            }

        }

        bool EsteInInteriorConvex(Complex[] p, Complex z)
        {
            int niv = 0;
            for (int k = 1; k < p.Length; k++)
            {
                if (EsteInStanga(p[k - 1], p[k], z)) niv++;
            }
            return niv == 0 || niv == p.Length - 1;
        }

        void UmpleInteriorConvex(Complex[] p, Color col)
        {
            for (int ii = 0; ii <= imax; ii++)
            {
                for (int jj = 0; jj <= jmax; jj++)
                {
                    if (EsteInInteriorConvex(p, getZ(ii, jj))) setPixel(ii, jj, col);
                }
            }

        }
```

```
void ColoreazaIndexJordan(Complex[] p)
{
    for (int ii = 0; ii <= imax; ii++)
    {
        for (int jj = 0; jj <= jmax; jj++)
        {
            Complex z = getZ(ii, jj);
            double s = 0;
            for (int k = 1; k < p.Length; k++)
            {
                s += ((p[k] - z) / (p[k - 1] - z)).Theta;
            }
            setPixel(ii, jj, getColor((int)(1000 + 20 * s)));
        }
    }
}

bool EsteInInteriorJordan(Complex[] p, Complex z)
{
    int N = p.Length;
    double s = 0;
    for (int k = 1; k < N; k++)
    {
        s += ((p[k] - z) / (p[k - 1] - z)).Theta;
    }
    return Abs(s - doiPI) < 0.001;
}

void UmpleInteriorJordan(Complex[] p, Color col)
{
    for (int ii = 0; ii <= imax; ii++)
    {
        for (int jj = 0; jj <= jmax; jj++)
        {
            if (EsteInInteriorJordan(p, getZ(ii, jj))) setPixel(ii, jj, col);
        }
    }
}

public override void makeImage()
{
    double R = 10;
    setXminXmaxYminYmax(-R, R, -R, R);
    ScreenColor = Color.WhiteSmoke;
    PenColor = Color.Navy;

    int N = 17; //numarul de varfuri
    double a = R / 3, b = R / 5, delta = 2 * PI / N;
    Complex[] v = new Complex[N + 1];
    for (int k = 0; k < N; k++)
    {
        v[k] = Complex.setReIm(a * Cos(k * delta), b * Sin(k * delta));
    }
    //este necesar ca p[p.Length-1]==p[0]
    v[N] = v[0];

    //v[N / 2] = new Complex(0, 0);
    ColoreazaSemiplane(v);
    //UmpleInteriorConvex(v, Color.DarkMagenta);
```

```
            //v[N / 2 + 1] = new Complex(0, b + 1);
            //ColoreazaIndexJordan(v);
            //UmpleInteriorJordan(v, Color.Crimson);
            TraseazaContur(v, PenColor);
            resetScreen();
        }
    }

}
```