Tema 2.

Utilizarea bibliotecii OpenGL pentru trasarea curbelor plane.

1. In exemplul urmator am utilizat primitiva grafica OpenGL de trasare a liniilor pentru a trasa
   1. graficul functiei : $|\sin x| \cdot e^{-\sin x}, x \in [0,8\pi]$ si
   2. graficul concoidei lui Nicomede (concoida dreptei) :
      $x = a \pm b \cdot \cos t, \quad y = a \cdot \operatorname{tg} t \pm b \cdot \sin t, \quad t \in (-\pi/2, \pi/2)$.
2. Integrati in exemplul precedent functii C care realizeaza :
   1. afisarea functiei :

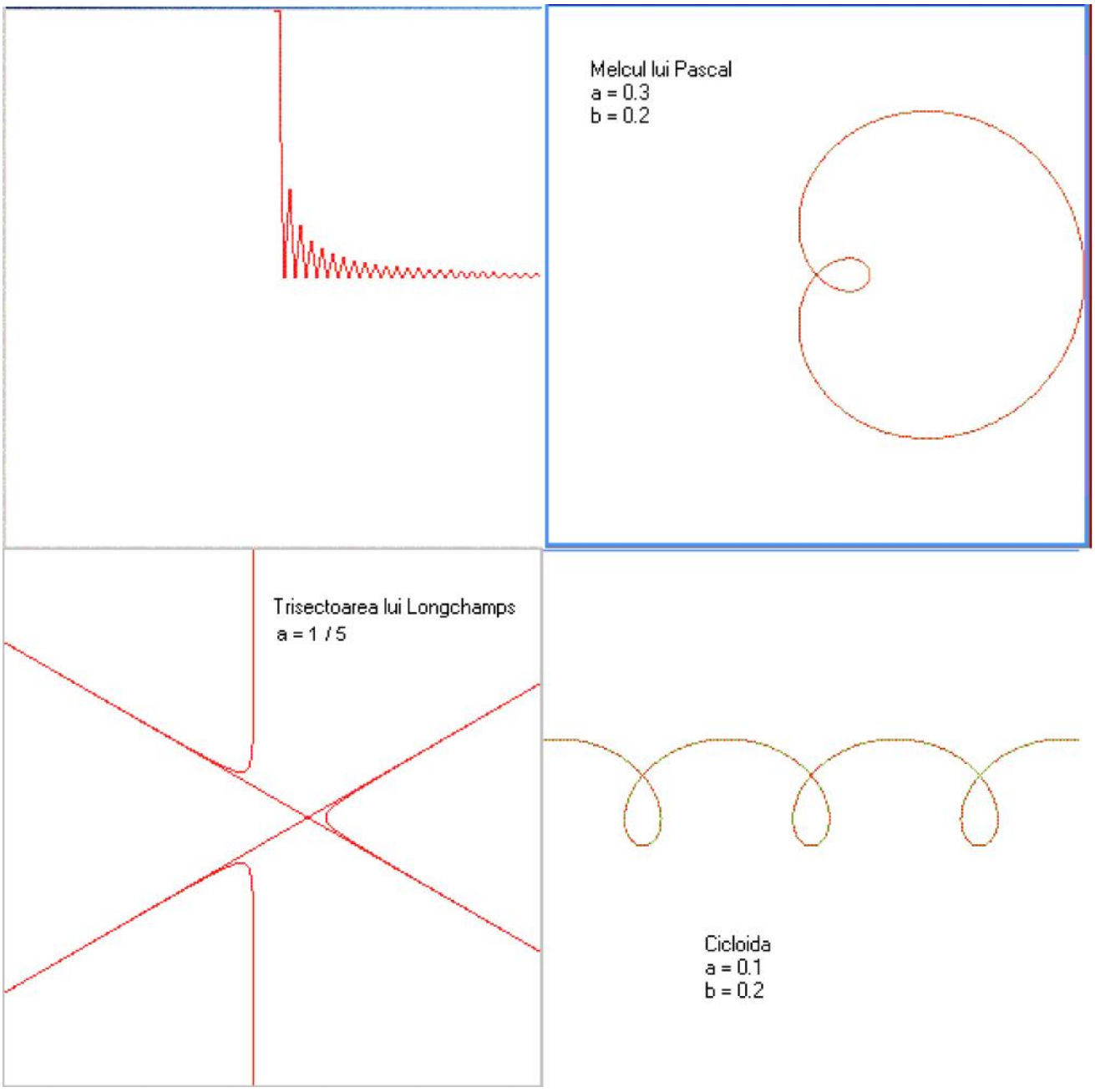
$$f(x) = \begin{cases} 1, & \text{pentru } x = 0 \\ \dfrac{d(x)}{x}, & \text{pentru } x > 0 \end{cases},$$

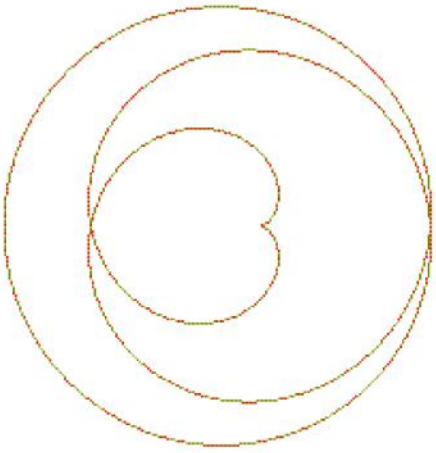
      unde *d(x)* este distanta de la x la cel mai apropiat intreg, pe intervalul [0,100].
   2. afisarea urmatoarelor curbe date prin ecuatii parametrice :
      1. melcul lui Pascal (concoida cercului) :
         $x = 2 \cdot (a \cdot \cos t + b) \cdot \cos t, \quad y = 2 \cdot (a \cdot \cos t + b) \cdot \sin t, \quad t \in (-\pi, \pi)$
      2. trisectoarea lui Longchamps :
         $$x = \frac{a}{4 \cdot \cos^2 t - 3}, \quad y = \frac{a \cdot \operatorname{tg} t}{4 \cdot \cos^2 t - 3}, \quad t \in (-\pi/2, \pi/2) \setminus \{\pm \pi/6\}$$
      3. cicloida :
         $x = a \cdot t - b \cdot \sin t, \quad y = a - b \cdot \cos t, \quad t \in \Re$
      4. epicicloida :
         $$x = (R + r) \cdot \cos\left(\frac{r}{R} \cdot t\right) - r \cdot \cos\left(t + \frac{r}{R} \cdot t\right),$$
         $$y = (R + r) \cdot \sin\left(\frac{r}{R} \cdot t\right) - r \cdot \sin\left(t + \frac{r}{R} \cdot t\right), \quad t \in [0, 2\pi]$$
      5. hipocicloida :
         $$x = (R - r) \cdot \cos\left(\frac{r}{R} \cdot t\right) - r \cdot \cos\left(t - \frac{r}{R} \cdot t\right),$$
         $$y = (R - r) \cdot \sin\left(\frac{r}{R} \cdot t\right) - r \cdot \sin\left(t - \frac{r}{R} \cdot t\right), \quad t \in [0, 2\pi]$$
3. Curbe date de ecuatii polare : coordonatele polare sunt *(r,t)*, unde $t \in [a,b]$ iar *r=f(t)*.
   Transformarea in coordonate carteziene a coordonatelor polare *(r,t)* este
   $x = r \cdot \cos t$
   $y = r \cdot \sin t$
   Sa se reprezinte urmatoarele curbe date prin ecuatii polare:
   1. lemniscata lui Bernoulli :

$$r = \pm a \cdot \sqrt{2 \cdot \cos(2 \cdot t)}, \quad t \in (-\pi/4, \pi/4)$$

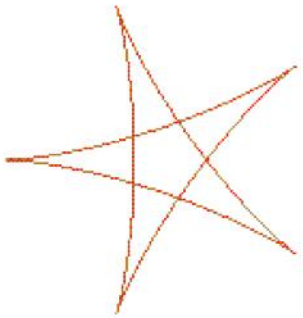
2. spirala logaritmica :

$$r = a \cdot e^{1+t}, \quad t \in (0, \infty)$$

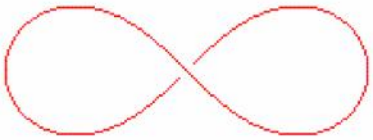




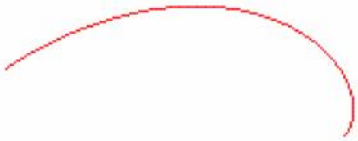
Epicicloida
R = 0.1 r = 0.3



Hipocicloida
R = 0.1
r = 0.3



Lemniscata lui Bernoulli
a = 0.4



Spirala logaritmica
a = 0.02

rebari, etc. : ghirvu@infoiasi.ro

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <limits>

#include <glut.h>

// dimensiunea ferestrei in pixeli
#define dim 300

unsigned char prevKey;

// concoida lui Nicomede (concoida dreptei)
// $x = a + b \cdot cos(t), y = a \cdot tg(t) + b \cdot sin(t)$. sau
// $x = a - b \cdot cos(t), y = a \cdot tg(t) - b \cdot sin(t)$. unde
// $t \in (-\pi / 2, \pi / 2)$
void Display1() {
    double xmax, ymax, xmin, ymin;
    double a = 1, b = 2;
    double pi = 4 * atan(1.0);
    double ratia = 0.05;
    double t;

    // calculul valorilor maxime/minime ptr. x si y
    // aceste valori vor fi folosite ulterior la scalare
    xmax = a - b - 1;
    xmin = a + b + 1;
    ymax = ymin = 0;
    for (t = - pi/2 + ratia; t < pi / 2; t += ratia) {
        double x1, y1, x2, y2;
        x1 = a + b * cos(t);
        xmax = (xmax < x1) ? x1 : xmax;
        xmin = (xmin > x1) ? x1 : xmin;

        x2 = a - b * cos(t);
        xmax = (xmax < x2) ? x2 : xmax;
        xmin = (xmin > x2) ? x2 : xmin;

        y1 = a * tan(t) + b * sin(t);
        ymax = (ymax < y1) ? y1 : ymax;
        ymin = (ymin > y1) ? y1 : ymin;

        y2 = a * tan(t) - b * sin(t);
        ymax = (ymax < y2) ? y2 : ymax;
        ymin = (ymin > y2) ? y2 : ymin;
    }

    xmax = (fabs(xmax) > fabs(xmin)) ? fabs(xmax) : fabs(xmin);
    ymax = (fabs(ymax) > fabs(ymin)) ? fabs(ymax) : fabs(ymin);

    // afisarea punctelor propriu-zise precedata de scalare
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
```

```
    for (t = - pi/2 + ratia; t < pi / 2; t += ratia) {
        double x1, y1, x2, y2;
        x1 = (a + b * cos(t)) / xmax;
        x2 = (a - b * cos(t)) / xmax;
        y1 = (a * tan(t) + b * sin(t)) / ymax;
        y2 = (a * tan(t) - b * sin(t)) / ymax;

        glVertex2f(x1,y1);
    }
    glEnd();

    glBegin(GL_LINE_STRIP);
    for (t = - pi/2 + ratia; t < pi / 2; t += ratia) {
        double x1, y1, x2, y2;
        x1 = (a + b * cos(t)) / xmax;
        x2 = (a - b * cos(t)) / xmax;
        y1 = (a * tan(t) + b * sin(t)) / ymax;
        y2 = (a * tan(t) - b * sin(t)) / ymax;

        glVertex2f(x2,y2);
    }
    glEnd();
}

// graficul functiei
// $f(x) = \bar sin(x) \bar \cdot e^{-sin(x)}, x \in \langle 0, 8 \cdot \pi \rangle$,
void Display2() {
    double pi = 4 * atan(1.0);
    double xmax = 8 * pi;
    double ymax = exp(1.1);
    double ratia = 0.05;

    // afisarea punctelor propriu-zise precedata de scalare
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for (double x = 0; x < xmax; x += ratia) {
        double x1, y1;
        x1 = x / xmax;
        y1 = (fabs(sin(x)) * exp(-sin(x))) / ymax;

        glVertex2f(x1,y1);
    }
    glEnd();
}
void Display3() {
    double ratia = 0.05;
    double xmax = 100;
    double ceilValue, floorValue;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for (double x = 0; x <=xmax; x += ratia) {
        double x1=x/100, y1;
        if(x==0)
```

```
            y1 = 1;
        else {
            ceilValue = ceil(x)-x;
            floorValue = x-floor(x);
            if(floorValue<ceilValue){
                y1 = floorValue/x;
            } else {
                y1 = ceilValue/x;
            }
        }

        glVertex2f(x1,y1);
    }
    glEnd();
}
void Display4() {
    double xmax = 100;
    double ratia = 0.05;
    double pi = 4 * atan(1.0);
    double t,x1,y1, a= 0.3, b= 0.2;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
        for(t = -pi+ratia; t < pi; t+=ratia){
            x1 = 2*(a*cos(t)+b)*cos(t);
            y1 = 2*(a*cos(t)+b)*sin(t);
            glVertex2f(x1,y1);
        }

    glEnd();
}
void Display5() {
    double xmax = 100;
    double ratia = 0.05;
    double pi = 4 * atan(1.0);
    double piPe2 = pi/2;
    double piPe6 = pi/6;
    double t,x1,y1, a= 0.2;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for(t = -piPe2+ratia; t < piPe2; t+=ratia){
        if(t!=piPe6 || t!=(-piPe6)){
            x1 = a/(4*pow(cos(t),2)-3);
            y1 = (a*tan(t))/(4*pow(cos(t),2)-3);
        }

        glVertex2f(x1,y1);
    }

    glEnd();
}
void Display6() {
    double xmax = 100;
    double ratia = 0.05;
```

```
    double pi = 4 * atan(1.0);
    double t,x1,y1, a=0.1, b=0.2;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for(t = -(4*pi); t <= (4*pi); t+=ratia){
        x1 = a*t-b*sin(t);
        y1 = a-b*cos(t);
        glVertex2f(x1,y1);
    }

    glEnd();
}
void Display7() {
    double xmax = 100;
    double ratia = 0.05;
    double pi = 4 * atan(1.0);
    double t,x1,y1, R=0.1, r=0.3;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for(t = 0; t <= (2*pi); t+=ratia){
        x1 = (R+r)*cos((r/R)*t)-r*cos(t+(r/R)*t);
        y1 = (R+r)*sin((r/R)*t)-r*sin(t+(r/R)*t);
        glVertex2f(x1,y1);
    }

    glEnd();
}
void Display8() {
    double xmax = 100;
    double ratia = 0.05;
    double pi = 4 * atan(1.0);
    double t,x1,y1, R=0.1, r=0.3;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for(t = 0; t <= (2*pi); t+=ratia){
        x1 = (R-r)*cos((r/R)*t)-r*cos(t-(r/R)*t);
        y1 = (R-r)*sin((r/R)*t)-r*sin(t-(r/R)*t);
        glVertex2f(x1,y1);
    }

    glEnd();
}
void Display9() {
    double xmax = 100;
    double ratia = 0.005;
    double pi = 4 * atan(1.0);
    double piPe4 = pi/4;
    double t,x1,y1, a=0.4,r;
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINE_STRIP);
    for(t = piPe4-ratia; t > -piPe4; t-=ratia){
        r=a*sqrt(2*cos(2*t));
        x1 =r*cos(t);
```

```
        y1 =r*sin(t);
        glVertex2f(x1,y1);
    }
    for(t = -piPe4+ratia; t < piPe4; t+=ratia){
        r=-a*sqrt(2*cos(2*t));
        x1 =r*cos(t);
        y1 =r*sin(t);
        glVertex2f(x1,y1);
    }

    glEnd();
}
void Display10() {
    double xmax = 100;
    double ratia = 0.05;
    double pi = 4 * atan(1.0);
    double piPe4 = pi/4;
    double t,x1,y1, a=0.02,r;
    glColor3f(1,0.1,0.1);
    glBegin(GL_LINE_STRIP);
    for(t = 0+ratia; t < (9999*pi); t+=ratia){
        r=a*exp(1+t);
        x1 =r*cos(t);
        y1 =r*sin(t);
        glVertex2f(x1,y1);
    }
    glEnd();
}
void Init(void) {

    glClearColor(1.0,1.0,1.0,1.0);

    glLineWidth(1);

    //   glPointSize(4);

    glPolygonMode(GL_FRONT, GL_LINE);
}

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);

    switch(prevKey) {
    case '1':
        Display1();
        break;
    case '2':
        Display2();
        break;
    case '3':
        Display3();
        break;
    case '4':
```

```
        Display4();
        break;
    case '5':
        Display5();
        break;
    case '6':
        Display6();
        break;
    case '7':
        Display7();
        break;
    case '8':
        Display8();
        break;
    case '9':
        Display9();
        break;
    case '0':
        Display10();
        break;
    default:
        break;
    }

    glFlush();
}

void Reshape(int w, int h) {
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
}

void KeyboardFunc(unsigned char key, int x, int y) {
    prevKey = key;
    if (key == 27) // escape
        exit(0);
    glutPostRedisplay();
}

void MouseFunc(int button, int state, int x, int y) {
}

int main(int argc, char** argv) {

    glutInit(&argc, argv);

    glutInitWindowSize(dim, dim);

    glutInitWindowPosition(100, 100);

    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);

    glutCreateWindow (argv[0]);
```

```
    Init();

    glutReshapeFunc(Reshape);

    glutKeyboardFunc(KeyboardFunc);

    glutMouseFunc(MouseFunc);

    glutDisplayFunc(Display);

    glutMainLoop();

    return 0;
}
```