

```
// Daca se doreste utilizarea bibliotecii GLUT trebuie
// inclus fisierul header GL/glut.h (acesta va include
// la GL/gl.h si GL/glu.h, fisierele header pentru
// utilizarea bibliotecii OpenGL). Functiile din biblioteca
// OpenGL sunt prefixate cu gl, cele din GLU cu glu si
// cele din GLUT cu glut.
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <glut.h>
```

```
#include <gl/GL.h>
```

```
#include <math.h>
```

```
unsigned char prevKey;
```

```
void Display1() {
    glColor3f(0.2,0.15,0.88); // albastru
    glBegin(GL_LINES); // trasarea unei linii
    glVertex2i(1,1); // coordonatele unui varf
    glVertex2i(-1,-1);
    glEnd();
```

```
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINES);
    glVertex2i(-1,1);
    glVertex2i(1,-1);
    glEnd();
```

```
    glBegin(GL_LINES);
    glVertex2d(-0.5,0);
    glVertex2d(0.5,0);
    glEnd();
}
```

```
void Display2() {
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINES);
    glVertex2f(1.0,1.0);
    glVertex2f(0.9,0.9);
    glVertex2f(0.8,0.8);
    glVertex2f(0.7,0.7);
    glVertex2f(0.6,0.6);
    glVertex2f(0.5,0.5);
    glVertex2f(-0.5,-0.5);
    glVertex2f(-1.0,-1.0);
    glEnd();
}
```

```
void Display3() {
    // trasare puncte GL_POINTS : deseneaza n puncte
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_POINTS);
    // de completat ...
}
```

```
    glVertex2f(0.5,0.5);
    glVertex2f(-0.5,-0.5);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.5,-0.5);

    glEnd();
}

void Display4() {
    glColor3f(1,0.1,0.1); // rosu
    glBegin(GL_LINES); // trasarea unei linii
    glVertex2f(1,1); // coordonatele unui varf
    glVertex2f(1,0.85);
    glVertex2f(0.5,0.6); // coordonatele unui varf
    glVertex2f(0.5,0.45);
    glEnd();
    // trasare linie poligonala GL_LINE_STRIP : (v0,v1), (v1,v2), (v_{n-2},v_{n-1})
    glBegin(GL_LINE_STRIP);
    // de completat ...
    glVertex2f(1,0.85); // coordonatele unui varf
    glVertex2f(0.5,0.6);
    glEnd();
}

void Display5() {
    glColor3f(1,0.1,0.1); // rosu
    // trasare linie poligonala inchisa GL_LINE_LOOP : (v0,v1), (v1,v2), (v_{n-1},v0)
    glBegin(GL_LINE_LOOP);
    glVertex2f(1,1); // coordonatele unui varf
    glVertex2f(1,0.85);
    glVertex2f(0.5,0.6); // coordonatele unui varf
    glVertex2f(0.5,0.45);
    // de completat ...
    glEnd();
}

void Display6() {
    glColor3f(1,0.1,0.1); // rosu
    // trasare triunghiuri GL_TRIANGLES : (v0,v1,v2), (v3,v4,v5), ...
    glBegin(GL_TRIANGLES);
    glVertex3f(1,1,0);
    glVertex3f(1,0.85,0);
    glVertex3f(0.85,0.85,0);
    glVertex3f(-1,-1,0);
    glVertex3f(-1,-0.85,0);
    glVertex3f(-0.85,-0.85,0);

    // de completat ...
    glEnd();
}

void Display7() {
    // trasare patrulatere GL_QUADS : (v0,v1,v2,v3), (v4,v5,v6,v7), ...
```

```

glBegin(GL_QUADS);
// de completat ...
glColor3f(1,0.1,0.1);//rosu
glVertex2f(1,1);
glVertex2f(1,0.6);
glVertex2f(0.6,0.6);
glVertex2f(0.4,0.75);
glEnd();
}

void Display8() {
// trasare poligon convex GL_QUADS : (v0,v1,v2, ..., v_{n-1})
glBegin(GL_POLYGON);
glColor3f(0.2,0.15,0.88);//albastru

glVertex3f(-0.4,0.6,0);
glVertex3f(0.4,0.6,0);
glVertex3f(0.7,0,0);
glVertex3f(0.4,-0.6,0);
glVertex3f(-0.4,-0.6,0);
glVertex3f(-0.7,0,0);
// de completat ...
glEnd();

glBegin(GL_POLYGON);
    glColor3f(1,0.1,0.1);//rosu

    glVertex3f(0.25,0.41,0);
    glVertex3f(0.48,0,0);
    glVertex3f(0.25,-0.41,0);
    glVertex3f(-0.25,-0.41,0);
    glVertex3f(-0.48,0,0);
    glVertex3f(-0.25,0.41,0);
glEnd();

//glBegin(GL_POLYGON);
//  glColor3f(1,1,1);//alb

//  glVertex3f(0.24,0.40,0);
//  glVertex3f(0.466,0,0);
//  glVertex3f(0.24,-0.40,0);
//  glVertex3f(-0.24,-0.40,0);
//  glVertex3f(-0.466,0,0);
//  glVertex3f(-0.24,0.40,0);
//glEnd();
glScalef(0.75,0.75,0.75);
glBegin(GL_POLYGON);
glColor3f(1,1,1);//albastru

glVertex3f(-0.4,0.6,0);
glVertex3f(0.4,0.6,0);
glVertex3f(0.7,0,0);
glVertex3f(0.4,-0.6,0);

```

```
    glVertex3f(-0.4,-0.6,0);
    glVertex3f(-0.7,0,0);
    // de completat ...
    glEnd();
}

void Init(void) {
    // specifica culoarea unui buffer dupa ce acesta
    // a fost sters utilizand functia glClear. Ultimul
    // argument reprezinta transparenta (1 - opacitate
    // completa, 0 - transparenta totala)
    glClearColor(1.0,1.0,1.0,1.0);

    // grosimea liniilor
    glLineWidth(3);

    // dimensiunea punctelor
    glPointSize(4);

    // functia void glPolygonMode (GLenum face, GLenum mode)
    // controleaza modul de desenare al unui poligon
    // mode : GL_POINT (numai vf. primitivei) GL_LINE (numai
    //          muchii) GL_FILL (poligonul plin)
    // face : tipul primitivei geometrice dpdv. al orientarii
    //          GL_FRONT - primitive orientate direct
    //          GL_BACK - primitive orientate invers
    //          GL_FRONT_AND_BACK - ambele tipuri
    glPolygonMode(GL_FRONT, GL_LINE);
}

void Display(void) {
    printf("Call Display\n");

    // sterge buffer-ul indicat
    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;
    default:
        break;
}

// forteaza redesenarea imaginii
glFlush();
}

/*
Parametrii w(latime) si h(inaltime) reprezinta noile
dimensiuni ale ferestrei
*/
void Reshape(int w, int h) {
    printf("Call Reshape : latime = %d, inaltime = %d\n", w, h);

    // functia void glViewport (GLint x, GLint y,
    //                               GLsizei width, GLsizei height)
    // defineste poarta de afisare : acea suprafata dreptunghiulara
    // din fereastra de afisare folosita pentru vizualizare.
    // x, y sunt coordonatele pct. din stg. jos iar
    // width si height sunt latimea si inaltimea in pixeli.
    // In cazul de mai jos poarta de afisare si fereastra coincid
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
}

/*
Parametrul key indica codul tastei iar x, y pozitia
cursorului de mouse
*/
void KeyboardFunc(unsigned char key, int x, int y) {
    printf("Ati tastat <%c>. Mouse-ul este in pozitia %d, %d.\n",
        key, x, y);
    // tasta apasata va fi utilizata in Display ptr.
    // afisarea unor imagini
    prevKey = key;
    if (key == 27) // escape
        exit(0);
    glutPostRedisplay();
}

/*
Codul butonului poate fi :
GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
Parametrul state indica starea: "apasat" GLUT_DOWN sau
"eliberat" GLUT_UP

```

```

Parametrii x, y : coordonatele cursorului de mouse
*/
void MouseFunc(int button, int state, int x, int y) {
    printf("Call MouseFunc : ati %s butonul %s in pozitia %d %d\n",
        (state == GLUT_DOWN) ? "apasat" : "eliberat",
        (button == GLUT_LEFT_BUTTON) ?
            "stang" :
        ((button == GLUT_RIGHT_BUTTON) ? "drept": "mijlociu"),
        x, y);
}

int main(int argc, char** argv) {
    // Initializarea bibliotecii GLUT. Argumentele argc
    // si argv sunt argumentele din linia de comanda si nu
    // trebuie modificate inainte de apelul functiei
    // void glutInit(int *argcp, char **argv)
    // Se recomanda ca apelul oricarei functii din biblioteca
    // GLUT sa se faca dupa apelul acestei functii.
    glutInit(&argc, argv);

    // Argumentele functiei
    // void glutInitWindowSize (int latime, int latime)
    // reprezinta latimea, respectiv inaltimea ferestrei
    // exprimate in pixeli. Valorile predefinite sunt 300, 300.
    glutInitWindowSize(300, 300);

    // Argumentele functiei
    // void glutInitWindowPosition (int x, int y)
    // reprezinta coordonatele varfului din stanga sus
    // al ferestrei, exprimate in pixeli.
    // Valorile predefinite sunt -1, -1.
    glutInitWindowPosition(100, 100);

    // Functia void glutInitDisplayMode (unsigned int mode)
    // seteaza modul initial de afisare. Acesta se obtine
    // printr-un SAU pe biti intre diverse masti de display
    // (constante ale bibliotecii GLUT) :
    // 1. GLUT_SINGLE : un singur buffer de imagine. Reprezinta
    // optiunea implicita ptr. nr. de buffere de
    // de imagine.
    // 2. GLUT_DOUBLE : 2 buffere de imagine.
    // 3. GLUT_RGB sau GLUT_RGBA : culorile vor fi afisate in
    // modul RGB.
    // 4. GLUT_INDEX : modul indexat de selectare al culorii.
    // etc. (vezi specificatia bibliotecii GLUT)
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);

    // Functia int glutCreateWindow (char *name)
    // creeaza o fereastră cu denumirea data de argumentul
    // name si intoarce un identificator de fereastră.
    glutCreateWindow (argv[0]);

    Init();
}

```

```
// Functii callback : functii definite in program si
// inregistrate in sistem prin intermediul unor functii
// GLUT. Ele sunt apelate de catre sistemul de operare
// in functie de evenimentul aparut

// Functia
// void glutReshapeFunc (void (*Reshape)(int width, int height))
// inregistreaza functia callback Reshape care este apelata
// oridecate ori fereastra de afisare isi modifica forma.
glutReshapeFunc(Reshape);

// Functia
// void glutKeyboardFunc (void (*KeyboardFunc)(unsigned char,int,int))
// inregistreaza functia callback KeyboardFunc care este apelata
// la actionarea unei taste.
glutKeyboardFunc(KeyboardFunc);

// Functia
// void glutMouseFunc (void (*MouseFunc)(int,int,int,int))
// inregistreaza functia callback MouseFunc care este apelata
// la apasarea sau la eliberarea unui buton al mouse-ului.
glutMouseFunc(MouseFunc);

// Functia
// void glutDisplayFunc (void (*Display)(void))
// inregistreaza functia callback Display care este apelata
// oridecate ori este necesara desenarea ferestrei: la
// initializare, la modificarea dimensiunilor ferestrei
// sau la apelul functiei
// void glutPostRedisplay (void).
glutDisplayFunc(Display);

// Functia void glutMainLoop() lanseaza bucla de procesare
// a evenimentelor GLUT. Din bucla se poate iesi doar prin
// inchiderea ferestrei aplicatiei. Aceasta functie trebuie
// apelata cel mult o singura data in program. Functiile
// callback trebuie inregistrate inainte de apelul acestei
// functii.
// Cand coada de evenimente este vida atunci este executata
// functia callback IdleFunc inregistrata prin apelul functiei
// void glutIdleFunc (void (*IdleFunc) (void))
glutMainLoop();

return 0;
}
```