Об’єктно орієнтоване програмування

Лабораторна робота №10

Наслідування та поліморфізм

Теоретичні відомості

Приклад. Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє зображати на холсті одного з компонентів прямокутник та сектор круга. Користувач має можливість переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру.

Лістинг 10.1 – Файл baseshape.h

```
#ifndef BASESHAPE_H
#define BASESHAPE_H
#include <QPainter>
class BaseShape
{
protected:
    double alpha;
    int centerX, centerY;
public:
    BaseShape(int cx, int cy);
    virtual ~BaseShape();
    void Rotate(double dAlpha);
    virtual void Draw(QPainter *p) = 0;
    void setScale(double scale);
    void MoveLeft();
    void MoveRight();
    void MoveUp();
    void MoveDown();
};
#endif // BASESHAPE_H
```

Лістинг 10.2 – Файл baseshape.cpp

```
#include "baseshape.h"
BaseShape::BaseShape(int cx, int cy)
{
    alpha = 0;
    centerX = cx;
    centerY = cy;
}
BaseShape::~BaseShape()
{

}
void BaseShape::Rotate(double dAlpha)
{
    alpha += dAlpha;
}
```

```
void BaseShape::MoveLeft()
{
    centerX -= 10;
}
void BaseShape::MoveRight()
{
    centerX += 10;
}
void BaseShape::MoveUp()
{
    centerY -= 10;
}
void BaseShape::MoveDown()
{
    centerY += 10;
}
```

Лістинг 10.3 – Файл rotatedrectangle.h

```
#ifndef ROTATEDRECTANGLE_H
#define ROTATEDRECTANGLE_H
#include "baseshape.h"
class RotatedRectangle : public BaseShape
{
private:
    const static int N = 4;
    int width, height;
    QPoint vertices[N];
    void CalculateVertices();
public:
    void Draw(QPainter *p);
    RotatedRectangle(int cx, int cy, int w, int h);
};
#endif // ROTATEDRECTANGLE_H
```

Лістинг 10.4 – Файл rotatedrectangle.cpp

```
#include "rotatedrectangle.h"
#include <math.h>
RotatedRectangle::RotatedRectangle(int cx, int cy, int w, int
h): BaseShape(cx, cy)
{
    width = w;
    height = h;
}

void RotatedRectangle::Draw(QPainter *p)
{
    CalculateVertices();
    p->drawPolygon(vertices, N);
}

void RotatedRectangle::CalculateVertices()
{
```

```
    vertices[0] = QPoint( width / 2, -height / 2);
    vertices[1] = QPoint(-width / 2, -height / 2);
    vertices[2] = QPoint(-width / 2,  height / 2);
    vertices[3] = QPoint( width / 2,  height / 2);

    for(int i = 0; i < N ; i++)
    {
        int x = vertices[i].x();
        int y = vertices[i].y();
        vertices[i].setX(int(x * cos(alpha) - y * sin(alpha) +
centerX));
        vertices[i].setY(int(x * sin(alpha) + y * cos(alpha) +
centerY));
    }
}
```

Лістинг 10.5 – Файл rotatedpie.h

```
#ifndef ROTATEDPIE_H
#define ROTATEDPIE_H

#include <QPainter>
#include <baseshape.h>

class RotatedPie : public BaseShape
{
    int r;
    void CalculateVertices();
public:
    void Draw(QPainter *p);
    RotatedPie(int cx, int cy, int r);
};
#endif // ROTATEDPIE_H
```

Лістинг 10.6 – Файл rotatedpie.cpp

```
#include "rotatedpie.h"
#include <math.h>
#include <qdebug.h>
RotatedPie::RotatedPie(int cx, int cy, int r):BaseShape(cx, cy)
{
    this->r = r;
}
void RotatedPie::Draw(QPainter *p)
{
/* startAngle та spanAngle повинні бути вказані як 1/16 градуса,
 * тобто повне коло дорівнює 5760 одиниць (16 * 360). Додатні
 * значення кутів означають напрям проти годинникової стрілки,
 * а від'ємні - за годинниковою стрілкою. Нульовий градус
 * знаходиться в положенні годинної стрілки на 3 години. */
     int startAngle = int( 30 + alpha*180./M_PI)*16 ;
     int spanAngle =  300*16;
```

```
    p->drawPie(centerX, centerY, 2*r, 2*r, startAngle,
spanAngle);
}
```

Лістинг 10.7 – Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <baseshape.h>
#include <QVector>

namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private slots:
    void on_pushButton_clicked();
    void on_pushButton_6_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_2_clicked();
    void on_pushButton_5_clicked();
private:
    Ui::MainWindow *ui;
    bool eventFilter(QObject* watched, QEvent* event);
    QVector<BaseShape*> shapes;
};

#endif // MAINWINDOW_H
```

Лістинг 10.8 – Файл mainwindow.h

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <math.h>
#include "rotatedrectangle.h"
#include "rotatedpie.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->widget->installEventFilter(this);
    shapes.push_back(new RotatedRectangle(100, 100, 60, 40));
    shapes.push_back(new RotatedRectangle(50, 30, 60, 40));
    shapes.push_back(new RotatedPie(70, 130, 40));
```

```
    for (int i = 0; i < shapes.size(); i++)
        ui->comboBox->addItem(QString::number(i));
}
MainWindow::~MainWindow()
{
    delete ui;
    for(auto i = shapes.begin(); i < shapes.end(); i++)
    {
        delete *i;
    }
    shapes.clear();
}
bool MainWindow::eventFilter(QObject* watched, QEvent* event)
{
    if(watched == ui->widget && event->type() == QEvent::Paint)
    {
        QPainter p(ui->widget);
        for(auto i = shapes.begin(); i < shapes.end(); i++)
        {
            (*i)->Draw(&p);
        }
        return true;
    }
    return false;
}
void MainWindow::on_pushButton_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->Rotate(M_PI/10.0);
    ui->widget->update();
}
void MainWindow::on_pushButton_6_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->Rotate(-M_PI/10.0);
    ui->widget->update();
}
void MainWindow::on_pushButton_3_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->MoveUp();
    ui->widget->update();
}
void MainWindow::on_pushButton_4_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->MoveDown();
    ui->widget->update();
}
void MainWindow::on_pushButton_2_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->MoveLeft();
```

```
    ui->widget->update();
}
void MainWindow::on_pushButton_5_clicked()
{
    int i = ui->comboBox->currentIndex();
    shapes[i]->MoveRight();
    ui->widget->update();
}
```

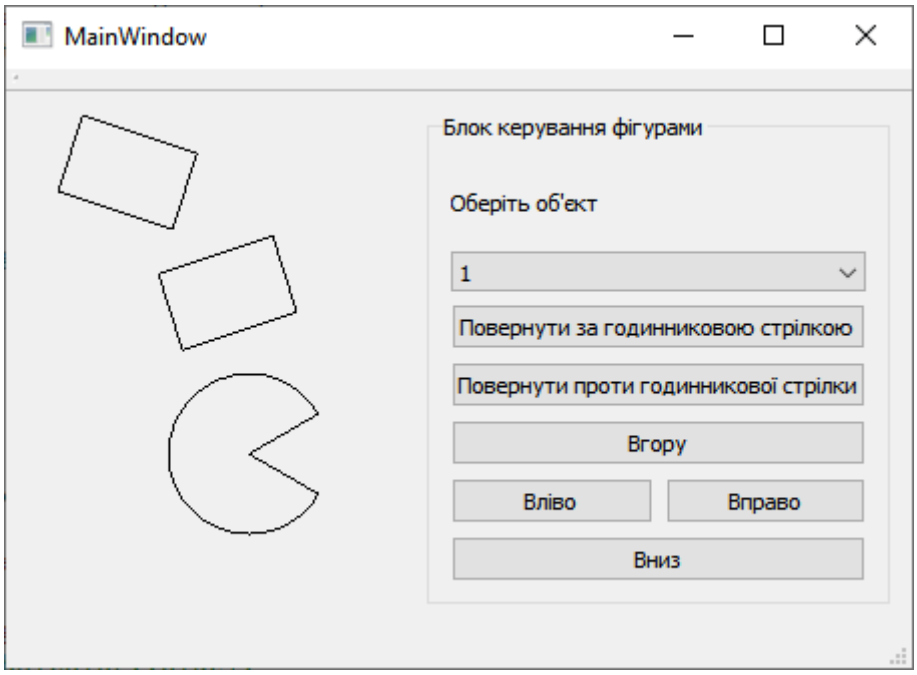

Рисунок 10.1 – Результат роботи програми

Варіанти завдань

Варіант 1.

Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:

1. зображати на холсті одного з компонентів рівнобедрений трикутник та сектор еліпсу 120°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 2.

Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:

1. зображати на холсті одного з компонентів ромб та сектор еліпсу 130°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 3.

Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:

1. зображати на холсті одного з компонентів трапецію та сектор еліпсу 140°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 4.

Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:

1. зображати на холсті одного з компонентів паралелограм та сектор еліпсу 150°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 5.

Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:

1. зображати на холсті одного з компонентів правильний п’ятикутник та сектор еліпсу 160°;
2. додавати необмежену кількість фігур відповідно до п.1;

3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 6.
Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:
1. зображати на холсті одного з компонентів правильний шестикутник та сектор еліпсу 170°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 7.
Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:
1. зображати на холсті одного з компонентів прямокутний трикутник та сектор еліпсу 180°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 8.
Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:
1. зображати на холсті одного з компонентів правильний восьмикутник та сектор еліпсу 190°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;

Варіант 9.
Написати програму, яка використовує механізм наслідування та поліморфізму. Програма дозволяє:
1. зображати на холсті одного з компонентів зірку та сектор еліпсу 200°;
2. додавати необмежену кількість фігур відповідно до п.1;
3. переміщувати фігури (вліво, вправо, вверх, вниз) та повертати їх навколо центру;
4. задавати колір контуру та колір зафарбування для кожної фігури;