Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Об’єктно-орієнтоване програмування

Лабораторна робота №4

Тема: «ПОЛІМОРФІЗМ. ОБРОБКА ВИНЯТКІВ»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: асистент
Козірова Н. Л.

Кропивницький 2025

Мета: ознайомитись з поняттям поліморфізму у мові C++ та навчитись використовувати віртуальні функції для досягнення поліморфізму. Також вивчити принципи обробки винятків у мові C++.

Варіант – 14 - 10 = 4

Завдання 1: Напишіть гру «арештуй таракана» на формі хаотично рухається таракан, гравцю потрібно курсором миші обвести таракана, таким чином заперши його в намальованій фігурі. В реалізації програми має бути похідний клас, що унаслідується від базового класу «QMainWindow», в похідному класі, перевизначить функцію «event».

Лістинг **.h** файлу:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTimer>
#include <QMouseEvent>
#include <QPainter>

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

protected:
    bool event(QEvent*) override;
    void paintEvent(QPaintEvent*) override;

private:
    Ui::MainWindow *ui;
    int roachSize;
    int roachX, roachY;
    int speedX, speedY;
    int score = 0;

    QTimer *roachTimer;
```

```
    QPixmap roachUp, roachDown, roachLeft, roachRight, roachDead;
    QPixmap loadRoach(const QString&);

    bool drawing = false;
    bool roachCaught = false;
    QPolygon polygon;

private slots:
    void moveRoach();
};
#endif // MAINWINDOW_H
```

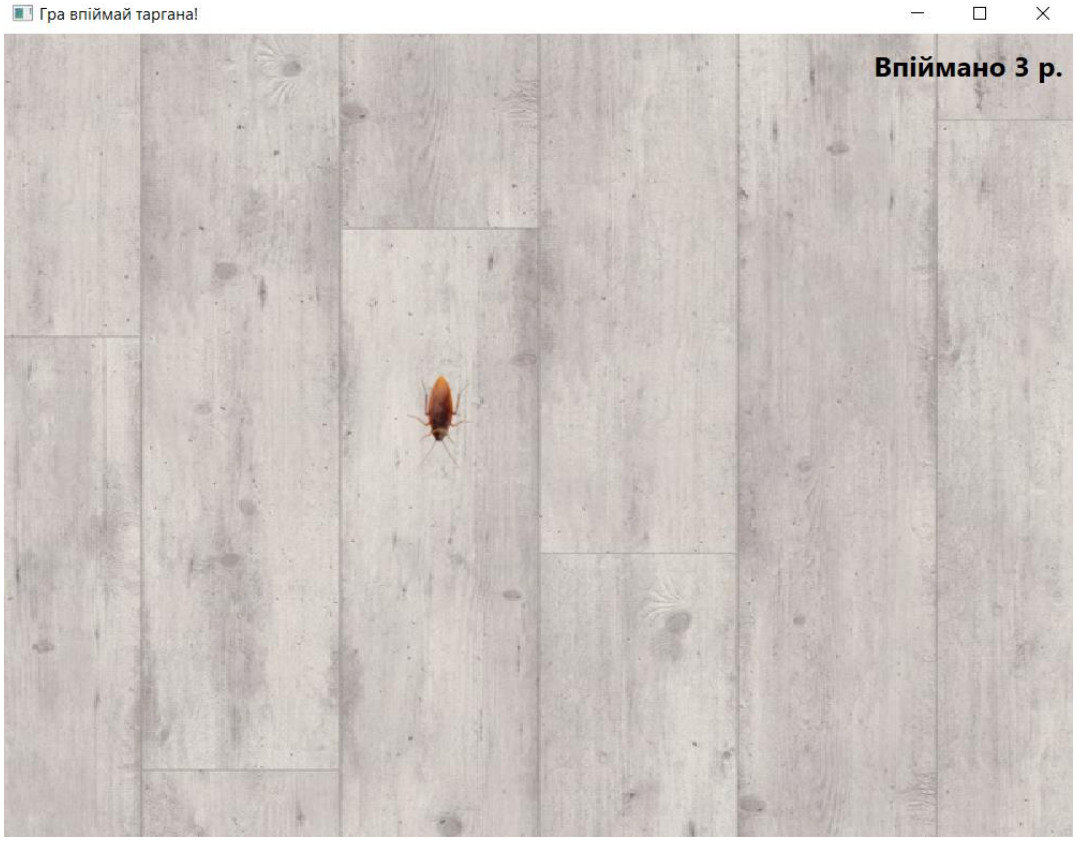
Лістинг **.cpp** файлу:

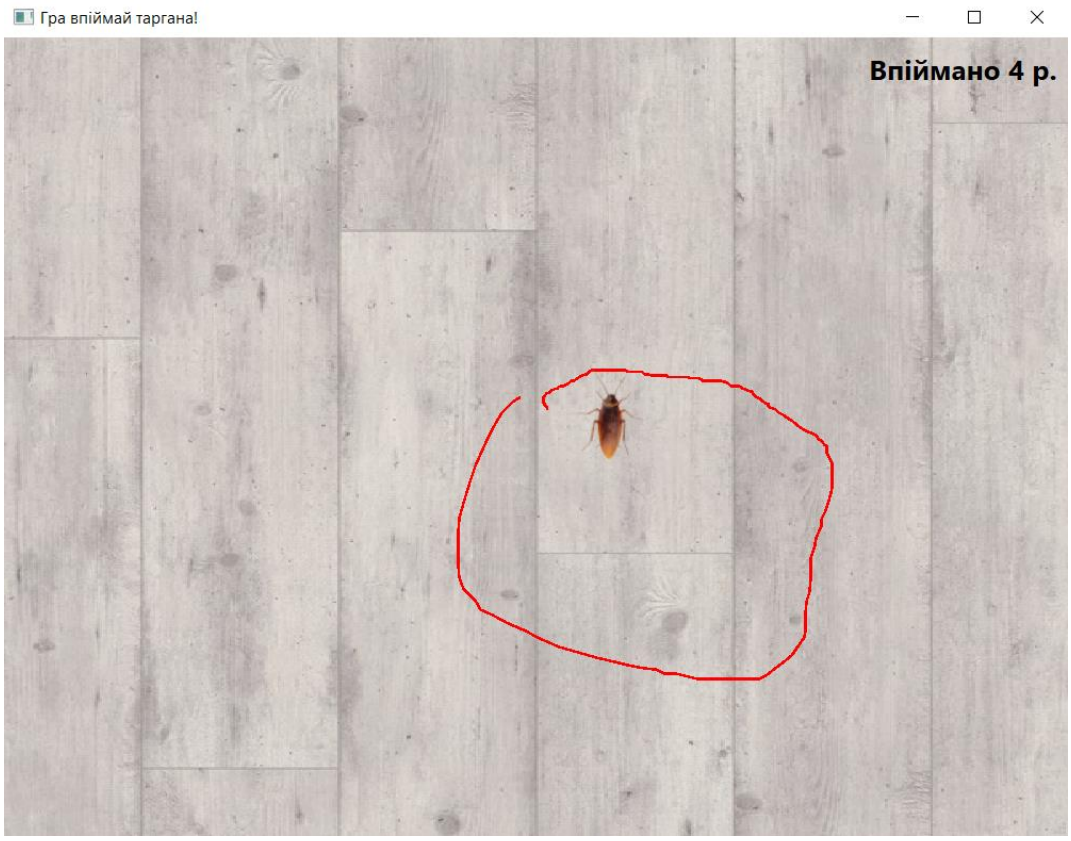
```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    setWindowTitle("Гра впіймай таргана!");

    roachSize = 80;
    roachX = roachY = 100;
    speedX = speedY = 2;

    std::srand(std::time(nullptr));
    ui->l_score->setStyleSheet("font-weight: bold; font-size: 16pt;");

    roachUp = loadRoach("up");
    roachDown = loadRoach("down");
    roachLeft = loadRoach("left");
    roachRight = loadRoach("right");
    roachDead = loadRoach("dead");

    ui->l_roach->setPixmap(roachUp);
    ui->l_roach->setGeometry(roachX, roachY, roachSize, roachSize);

    ui->l_message->setStyleSheet("font-weight: bold; font-size: 16pt;");
    ui->l_message->setGeometry(0, 0, width(), height());
    ui->l_message->hide();

    roachTimer = new QTimer(this);
    connect(roachTimer, &QTimer::timeout, this, &MainWindow::moveRoach);
    roachTimer->start(5);
}

MainWindow::~MainWindow()
{
    delete ui;
}

QPixmap MainWindow::loadRoach(const QString &direction) {
    return QPixmap(":/roach_" + direction + ".png")
        .scaled(80, 80, Qt::KeepAspectRatio, Qt::SmoothTransformation);
```

```
}

void MainWindow::moveRoach()
{
    if (roachCaught) return;

    if (rand() % 30 == 0) {
        speedX = (rand() % 7) - 3;
        speedY = (rand() % 7) - 3;
    }

    speedX = qBound(-10, speedX, 10);
    speedY = qBound(-10, speedY, 10);

    roachX += speedX;
    roachY += speedY;

    if (roachX <= 0 || roachX >= width() - ui->l_roach->width()) speedX = -
speedX;
    if (roachY <= 0 || roachY >= height() - ui->l_roach->height()) speedY = -
speedY;

    if (abs(speedX) > abs(speedY)) {
        if (speedX > 0) ui->l_roach->setPixmap(roachRight);
        else ui->l_roach->setPixmap(roachLeft);
    } else {
        if (speedY > 0) ui->l_roach->setPixmap(roachDown);
        else ui->l_roach->setPixmap(roachUp);
    }

    ui->l_roach->move(roachX, roachY);
}

bool MainWindow::event(QEvent *e)
{
    if (roachCaught && e->type() == QEvent::MouseButtonPress) {
        roachCaught = false;
        ui->l_message->hide();
        ui->l_roach->setPixmap(roachUp);
    }
    if (e->type() == QEvent::MouseButtonPress) {
        drawing = true;
        polygon.clear();
        polygon << static_cast<QMouseEvent*>(e)->pos();
        return true;
    }
    if (e->type() == QEvent::MouseMove && drawing) {
        polygon << static_cast<QMouseEvent*>(e)->pos();
        update();
        return true;
    }
    if (e->type() == QEvent::MouseButtonRelease && drawing) {
        drawing = false;

        QPoint center(roachX + roachSize/2, roachY + roachSize/2);
        if (polygon.containsPoint(center, Qt::OddEvenFill)){
            score++;
            ui->l_score->setText("Впіймано " + QString::number(score) + "
p.");
```

```
            qDebug() << "Тарган спійманий" << score << "раз!";

            roachCaught = true;
            ui->l_roach->setPixmap(roachDead);
            ui->l_message->show();
        }

        polygon.clear();
        update();
        return true;
    }

    return QMainWindow::event(e);
}

void MainWindow::paintEvent(QPaintEvent *event)
{
    QMainWindow::paintEvent(event);
    QPainter p(this);
    QPixmap bg(":/floor.jpeg");
    p.drawPixmap(0, 0, width(), height(), bg);
    ui->scoreWidget->setGeometry(0, 0, ui->centralwidget->width(), ui-
>centralwidget->height());

    if (!polygon.isEmpty()) {
        QPainter p(this);
        p.setPen(QPen(Qt::red, 2));
        p.drawPolyline(polygon);
    }
}
```

Результат виконання:




Гра впіймай таргана!
Впіймано 4 р.


Гра впіймай таргана!
Впіймано 5 р.
Натисніть, щоб продовжити!

Завдання 2:

Варіант 4

1. Розробіть систему керування банком, яка включатиме обробку різних типів банківських рахунків, таких як звичайний рахунок та рахунок з відсотковою ставкою. Кожен тип рахунку має свої характеристики, які потрібно реалізувати за допомогою наслідування, сетерів та гетерів.

Кожен банківський рахунок має наступні характеристики:

- Номер рахунку
- Власник рахунку
- Баланс

Кожен тип рахунку має власні додаткові характеристики:

Звичайний рахунок:

- Мінімальний дозволений баланс

Рахунок з відсотковою ставкою:

- Відсоткова ставка

2. Створіть базовий абстрактний клас BankAccount з віртуальними функціями та використати поліморфізм для реалізації додаткових методів та функцій. Також, додайте виняткові ситуації для обробки некоректних даних.

3. Створіть похідні класи RegularAccount та InterestAccount, які успадковуються від класу BankAccount. Реалізуйте в них відповідні віртуальні функції та додайте додаткові характеристики, які були зазначені вище.

4. У вашій програмі мають бути використані виняткові ситуації для обробки некоректних даних, наприклад, якщо некоректний номер рахунку або негативна відсоткова ставка.

Ви можете розширити його, додати додаткові методи та функціональні можливості, які вам здаються відповідними.

Лістинг проекту **bankAccount**(github)

Результат виконання