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. КБ-23 |
| Захарченко А.О. |
| Перевірив: викладач  Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

**Тема: Поліморфізм. Обробка винятків**

**Мета: Ознайомитись з поняттям поліморфізму у мові C++ та навчитись**

**використовувати віртуальні функції для досягнення поліморфізму. Також**

**вивчити принципи обробки винятків у мові C++**

**Завдання 1:**

**Напишіть гру «Піймай муху», на формі знаходиться муха, яка має**

**тікати від курсора, також на формі має знаходитися пастка, коли муха**

**попадеться в пастку гра закінчується. В реалізації програми має бути**

**похідний клас, що унаслідується від базового класу «QMainWindow»,**

**в похідному класі, перевизначить функцію «event»**

**GameWindow.h:**

**#ifndef GAMEWINDOW\_H**

**#define GAMEWINDOW\_H**

**#include <QMainWindow>**

**#include <QMouseEvent>**

**#include <QLabel>**

**#include <QRandomGenerator>**

**#include <QDebug>**

**class GameWindow : public QMainWindow**

**{**

**Q\_OBJECT**

**public:**

**explicit GameWindow(QWidget \*parent = nullptr);**

**~GameWindow();**

**protected:**

**void mouseMoveEvent(QMouseEvent \*event) override; // Обробка руху миші**

**private:**

**QLabel \*fly; // Муха**

**QLabel \*trap; // Пастка**

**void moveFly(); // Функція для переміщення мухи**

**private slots:**

**void checkCollision(); // Перевірка на зіткнення**

**};**

**#endif // GAMEWINDOW\_H**

**GameWindow.cpp:**

**#include "GameWindow.h"**

**#include <QVBoxLayout>**

**#include <QTimer>**

**GameWindow::GameWindow(QWidget \*parent)**

**: QMainWindow(parent)**

**{**

**// Налаштування вікна**

**setWindowTitle("Піймай муху");**

**setFixedSize(800, 600);**

**// Створення мухи**

**fly = new QLabel(this);**

**fly->setStyleSheet("background-color: red; border-radius: 25px;");**

**fly->setFixedSize(50, 50);**

**fly->move(375, 275); // Початкова позиція**

**fly->show(); // Додано для відображення мухи**

**// Створення пастки**

**trap = new QLabel(this);**

**trap->setStyleSheet("background-color: blue;");**

**trap->setFixedSize(100, 100);**

**trap->move(350, 250); // Позиція пастки**

**trap->show(); // Додано для відображення пастки**

**}**

**GameWindow::~GameWindow()**

**{**

**}**

**void GameWindow::mouseMoveEvent(QMouseEvent \*event)**

**{**

**// При русі курсора над вікном переміщуємо муху**

**moveFly();**

**// Перевірка на зіткнення з пасткою**

**checkCollision();**

**}**

**void GameWindow::moveFly()**

**{**

**// Випадкове переміщення мухи**

**int x = QRandomGenerator::global()->bounded(0, width() - fly->width());**

**int y = QRandomGenerator::global()->bounded(0, height() - fly->height());**

**fly->move(x, y);**

**qDebug() << "Муха переміщена на:" << x << "," << y;**

**}**

**void GameWindow::checkCollision()**

**{**

**if (!fly || !trap) {**

**qDebug() << "Fly or trap is null!";**

**return; // Виходимо, якщо один з вказівників є нульовим**

**}**

**QRect flyRect = fly->geometry();**

**QRect trapRect = trap->geometry();**

**qDebug() << "Геометрія мухи:" << flyRect;**

**qDebug() << "Геометрія пастки:" << trapRect;**

**if (flyRect.intersects(trapRect)) {**

**// Гра закінчується**

**fly->setText("Ви зловили муху! Гра закінчена!");**

**fly->setStyleSheet("color: white; background-color: green;");**

**fly->setAlignment(Qt::AlignCenter);**

**fly->setFixedSize(400, 100);**

**fly->move((width() - fly->width()) / 2, (height() - fly->height()) / 2);**

**}**

**}**

**Main.cpp:**

**#include <QApplication>**

**#include "GameWindow.h"**

**int main(int argc, char \*argv[])**

**{**

**QApplication a(argc, argv);**

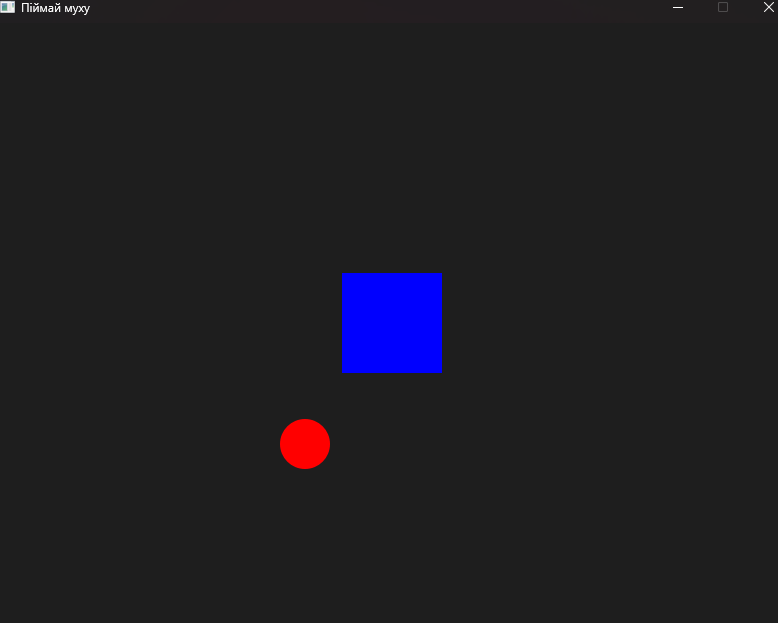
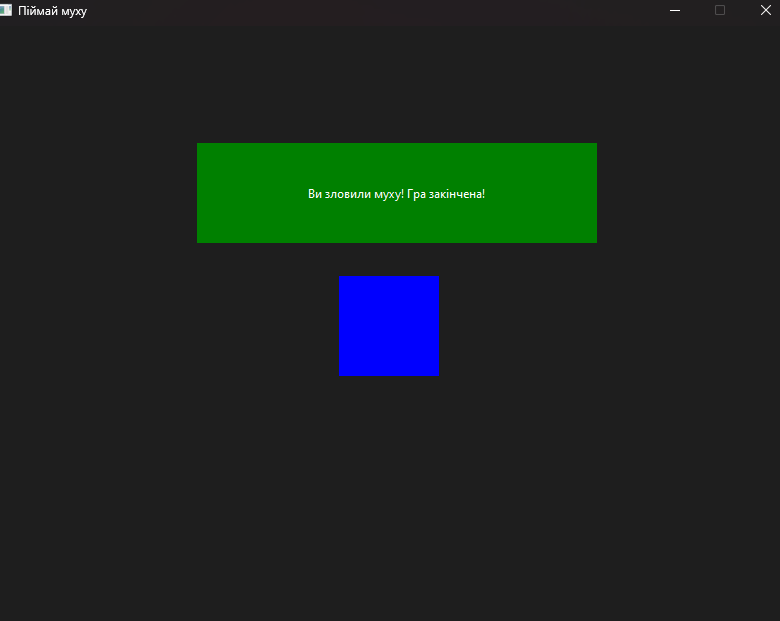
**GameWindow w;**

**w.show();**

**return a.exec();**

**}**

**Результат:**

****

**Завдання 2:**

**1. Розробіть систему керування продуктовим магазином, яка включатиме**

**обробку різних типів продуктів, таких як фрукти, овочі та молочні продукти.**

**Кожен тип продукту має свої характеристики, які потрібно реалізувати за**

**допомогою наслідування, сетерів та гетерів.**

**Кожен продукт має наступні характеристики:**

* **Назва продукту**
* **Ціна (за одиницю товару)**

**• Кількість на складі**

**Кожен тип продукту має власні додаткові характеристики:**

**Фрукти:**

* **Тип фрукту (яблуко, банан, апельсин і т.д.)**
  + **Вага (у кілограмах)**

**Овочі:**

* **Тип овоча (морква, капуста, буряк і т.д.)**
* **Країна походження**

**Молочні продукти:**

* **Тип продукту (молоко, йогурт, сир і т.д.)**

**• Виробник**

**2. Створіть базовий абстрактний клас Product з віртуальними функціями**

**та використати поліморфізм для реалізації додаткових методів та функцій.**

**Також, додайте виняткові ситуації для обробки некоректних даних.**

**3. Створіть похідні класи Fruit, Vegetable та DairyProduct, які**

**успадковуються від класу Product. 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 введена некоректна кількість на**

**складі або негативна ціна.**

**Ви можете розширити його, додати додаткові методи та функціональні**

**можливості, які вам здаються відповідними.**

**Product.cpp:**

#include "Product.h"

Product::Product(const std::string& name, double price, int quantity)

: name(name), price(price), quantity(quantity) {

if (price < 0) throw std::invalid\_argument("Price cannot be negative.");

if (quantity < 0) throw std::invalid\_argument("Quantity cannot be negative.");

}

void Product::setName(const std::string& name) { this->name = name; }

std::string Product::getName() const { return name; }

void Product::setPrice(double price) {

if (price < 0) throw std::invalid\_argument("Price cannot be negative.");

this->price = price;

}

double Product::getPrice() const { return price; }

void Product::setQuantity(int quantity) {

if (quantity < 0) throw std::invalid\_argument("Quantity cannot be negative.");

this->quantity = quantity;

}

int Product::getQuantity() const { return quantity; }

Fruit::Fruit(const std::string& name, double price, int quantity, const std::string& type, double weight)

: Product(name, price, quantity), type(type), weight(weight) {}

void Fruit::setType(const std::string& type) { this->type = type; }

std::string Fruit::getType() const { return type; }

void Fruit::setWeight(double weight) { this->weight = weight; }

double Fruit::getWeight() const { return weight; }

void Fruit::displayInfo() const {

std::cout << "Fruit: " << name << ", Price: " << price

<< ", Quantity: " << quantity << ", Type: " << type

<< ", Weight: " << weight << " kg" << std::endl;

}

Vegetable::Vegetable(const std::string& name, double price, int quantity, const std::string& type, const std::string& originCountry)

: Product(name, price, quantity), type(type), originCountry(originCountry) {}

void Vegetable::setType(const std::string& type) { this->type = type; }

std::string Vegetable::getType() const { return type; }

void Vegetable::setOriginCountry(const std::string& originCountry) { this->originCountry = originCountry; }

std::string Vegetable::getOriginCountry() const { return originCountry; }

void Vegetable::displayInfo() const {

std::cout << "Vegetable: " << name << ", Price: " << price

<< ", Quantity: " << quantity << ", Type: " << type

<< ", Origin Country: " << originCountry << std::endl;

}

DairyProduct::DairyProduct(const std::string& name, double price, int quantity, const std::string& type, const std::string& manufacturer)

: Product(name, price, quantity), type(type), manufacturer(manufacturer) {}

void DairyProduct::setType(const std::string& type) { this->type = type; }

std::string DairyProduct::getType() const { return type; }

void DairyProduct::setManufacturer(const std::string& manufacturer) { this->manufacturer = manufacturer; }

std::string DairyProduct::getManufacturer() const { return manufacturer; }

void DairyProduct::displayInfo() const {

std::cout << "Dairy Product: " << name << ", Price: " << price

<< ", Quantity: " << quantity << ", Type: " << type

<< ", Manufacturer: " << manufacturer << std::endl;

}

**Product.h:**

#pragma once

#include <string>

#include <iostream>

#include <stdexcept>

// Абстрактний клас Product

class Product {

protected:

std::string name;

double price;

int quantity;

public:

Product(const std::string& name, double price, int quantity);

virtual ~Product() = default;

void setName(const std::string& name);

std::string getName() const;

void setPrice(double price);

double getPrice() const;

void setQuantity(int quantity);

int getQuantity() const;

// Віртуальна функція для виведення інформації про продукт

virtual void displayInfo() const = 0;

};

class Fruit : public Product {

private:

std::string type;

double weight;

public:

Fruit(const std::string& name, double price, int quantity, const std::string& type, double weight);

void setType(const std::string& type);

std::string getType() const;

void setWeight(double weight);

double getWeight() const;

void displayInfo() const override;

};

class Vegetable : public Product {

private:

std::string type;

std::string originCountry;

public:

Vegetable(const std::string& name, double price, int quantity, const std::string& type, const std::string& originCountry);

void setType(const std::string& type);

std::string getType() const;

void setOriginCountry(const std::string& originCountry);

std::string getOriginCountry() const;

void displayInfo() const override;

};

class DairyProduct : public Product {

private:

std::string type;

std::string manufacturer;

public:

DairyProduct(const std::string& name, double price, int quantity, const std::string& type, const std::string& manufacturer);

void setType(const std::string& type);

std::string getType() const;

void setManufacturer(const std::string& manufacturer);

std::string getManufacturer() const;

void displayInfo() const override;

};

**Task-2.cpp:**

#include <iostream>

#include "Product.h"

int main() {

try {

Fruit apple("Apple", 2.5, 100, "Apple", 0.2);

Vegetable carrot("Carrot", 1.5, 200, "Carrot", "USA");

DairyProduct milk("Milk", 1.2, 50, "Milk", "DairyFarm");

apple.displayInfo();

carrot.displayInfo();

milk.displayInfo();

}

catch (const std::invalid\_argument& e) {

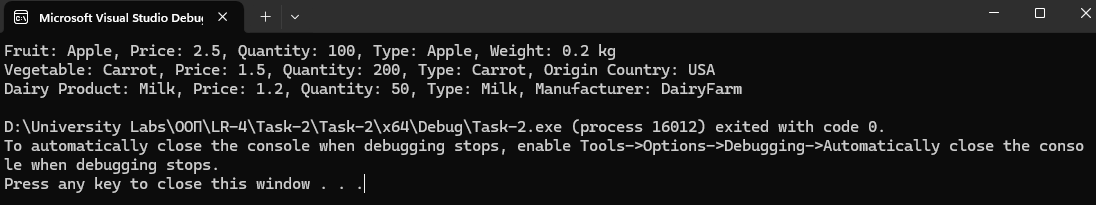
std::cerr << "Error: " << e.what() << std::endl;

}

return 0;

}

**Результат:**

****

**Висновок:**

Виконавши ЛР4 було покращено навичики написання коду на мові програмування С++, ознайомлено з поняттями Поліморфізму та обробкою винятків у ООП. Також при виконанні завдання 1 (написати власну гру) було ознайомлено з Qt Creator.