Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №4**

**Тема:** «Поліморфізм. Обробка винятків»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Кобзар М. В. |
| Перевірив: Перевірив: Козірова Н. Л. |
|  |

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:**

Завдання 1

Примітка

Для виконання цього завдання вам потрібно середовище розробки Qt

Creator, якщо Ви раніше не працювали з ним то встановіть, та ознайомтесь.

Наступним кроком створіть новий проект Qt: Використовуйте Qt Creator

для створення нового проекту типу "Qt Widgets Application". При створені

проекту ви отримаєте всі файли, які потрібні і навіть більше, отже вітаю,

з цим «більше» вам і прийдеться працювати, а якщо точніше то до цього

завдання вам треба зробити інтерфейс. Тему для завдання можна обрати на

вибір

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»

2. 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».

3. Напишіть інтерфейс до лабораторної роботи №1. Реалізуйте можливість

введення інформації про об’єкт, а також виведення інформації різного

типу (наприклад тип1: кількість об’єктів, тип2: властивості об’єкту) в

одному label по запиту з клавіатури. В реалізації програми має бути

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

в похідному класі, перевизначить функцію «event.

Завдання 2

1. Розробіть систему керування одягом, яка включатиме різні типи одягу.

Кожен тип одягу має свої характеристики, які потрібно реалізувати за

допомогою наслідування, сетерів та гетерів.

Кожен одяг має наступні характеристики:

• Назва

• Розмір

• Колір

Кожен тип одягу має власні додаткові характеристики:

Футболка:

• Матеріал

• Тип коміра

Штани:

• Матеріал

• Тип застібки

2. Створіть базовий абстрактний клас Clothing з віртуальними функціями

та використати поліморфізм для реалізації додаткових методів та функцій.

Також, додайте виняткові ситуації для обробки некоректних даних.

3. Створіть похідні класи TShirt та Pants, які успадковуються від класу

Clothing. 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 розмір одягу або

порожнє ім'я.

Ви можете розширити його, додати додаткові методи та функціональні

можливості, які вам здаються відповідними.

**Варіант 8**

**Лістинг :**

Завдання 1

muha.h

#ifndef MUHA\_H

#define MUHA\_H

#include <QMainWindow>

#include <QEvent>

#include <QMouseEvent>

#include <QLabel>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class muha;

}

QT\_END\_NAMESPACE

class muha : public QMainWindow

{

Q\_OBJECT

public:

muha(QWidget \*parent = nullptr);

~muha();

protected:

bool event(QEvent \*e) override;

private:

Ui::muha \*ui;

QLabel \*fly;

QLabel \*trap;

void moveFlyAway(const QPoint &mousePos);

bool checkCollision();

};

#endif // MUHA\_H

muha.cpp

#include "muha.h"

#include "ui\_muha.h"

#include <QPixmap>

#include <QDebug>

#include <QRandomGenerator>

#include <QMessageBox>

muha::muha(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::muha)

{

ui->setupUi(this);

fly = new QLabel(this);

QPixmap pix;

pix.load("C://Users/kobza/OneDrive/Рабочий стол/PROJECT\_KNTU/OOP\_lab/lab\_4/fly/muha.png");

fly->setPixmap(pix);

fly->setScaledContents(true);

fly->setGeometry(50, 50, 50, 50);

trap = new QLabel(this);

QPixmap trapPix;

trapPix.load("C://Users/kobza/OneDrive/Рабочий стол/PROJECT\_KNTU/OOP\_lab/lab\_4/fly/trap.png"); // Шлях до зображення пастки

trap->setPixmap(trapPix);

trap->setScaledContents(true);

trap->setGeometry(300, 300, 100, 100);

}

muha::~muha()

{

delete ui;

}

bool muha::event(QEvent \*e)

{

if (e->type() == QEvent::MouseMove) {

QMouseEvent \*mouseEvent = static\_cast<QMouseEvent\*>(e);

moveFlyAway(mouseEvent->pos());

if (checkCollision()) {

QMessageBox::information(this, "Перемога", "Муха попалася в пастку!");

close();

}

}

return QMainWindow::event(e);

}

void muha::moveFlyAway(const QPoint &mousePos)

{

int flyX = fly->x();

int flyY = fly->y();

//курскор - близько муха по тапкам

if (qAbs(mousePos.x() - flyX) < 100 && qAbs(mousePos.y() - flyY) < 100) {

int newX = QRandomGenerator::global()->bounded(this->width() - fly->width());

int newY = QRandomGenerator::global()->bounded(this->height() - fly->height());

fly->move(newX, newY);

}

}

bool muha::checkCollision()

{

QRect flyRect = fly->geometry();

QRect trapRect = trap->geometry();

return flyRect.intersects(trapRect);

}

main.cpp

#include "muha.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

muha w;

w.show();

return a.exec();

}

Завдання 2

Clothing.h

#ifndef CLOTHING\_H

#define CLOTHING\_H

#include <string>

#include <stdexcept>

using namespace std;

class Clothing {

protected:

string name;

string size;

string color;

public:

Clothing(const string& name, const string& size, const string& color) {

setName(name);

setSize(size);

this->color = color;

}

virtual ~Clothing() {}

virtual void setName(const string& name) {

if (name.empty()) {

throw invalid\_argument("no name");

}

this->name = name;

}

virtual void setSize(const string& size) {

if (size != "S" && size != "M" && size != "L" && size != "XL") {

throw invalid\_argument("no");

}

this->size = size;

}

virtual void setColor(const string& color) {

this->color = color;

}

virtual string getName() const { return name; }

virtual string getSize() const { return size; }

virtual string getColor() const { return color; }

virtual void displayInfo() const = 0;

};

#endif // CLOTHING\_H

Pants.h

#ifndef PANTS\_H

#define PANTS\_H

#include "Clothing.h"

#include <iostream>

using namespace std;

class Pants : public Clothing {

private:

string material;

string fasteningType;

public:

Pants(const string& name, const string& size, const string& color, const string& material, const string& fasteningType)

: Clothing(name, size, color), material(material), fasteningType(fasteningType) {}

void setMaterial(const string& material) {

this->material = material;

}

void setFasteningType(const string& fasteningType) {

this->fasteningType = fasteningType;

}

string getMaterial() const { return material; }

string getFasteningType() const { return fasteningType; }

void displayInfo() const override {

cout << "pants: " << getName() << ", Size: " << getSize()

<< ", coor: " << getColor() << ", Material: " << material

<< ", Fastener type: " << fasteningType << std::endl;

}

};

#endif // PANTS\_H

TShirt.h

#ifndef TSHIRT\_H

#define TSHIRT\_H

#include "Clothing.h"

#include <iostream>

using namespace std;

class TShirt : public Clothing {

private:

string material;

string collarType;

public:

TShirt(const string& name, const string& size, const string& color, const string& material, const string& collarType)

: Clothing(name, size, color), material(material), collarType(collarType) {}

void setMaterial(const string& material) {

this->material = material;

}

void setCollarType(const string& collarType) {

this->collarType = collarType;

}

string getMaterial() const { return material; }

string getCollarType() const { return collarType; }

void displayInfo() const override {

cout << "TShirt: " << getName() << ", Size: " << getSize()

<< ", Color: " << getColor() << ", Material: " << material

<< ", Collar type: " << collarType << endl;

}

};

#endif // TSHIRT\_H

main.cpp

#include <iostream>

#include "TShirt.h"

#include "Pants.h"

//#include <windows.h>

using namespace std;

int main() {

try {

TShirt tshirt("Nike", "M", "black", "bavovna", "circle");

Pants pants("Levi's", "L", "bluetooth", "jeans", "lightning");

tshirt.displayInfo();

pants.displayInfo();

}

catch (const exception& e) {

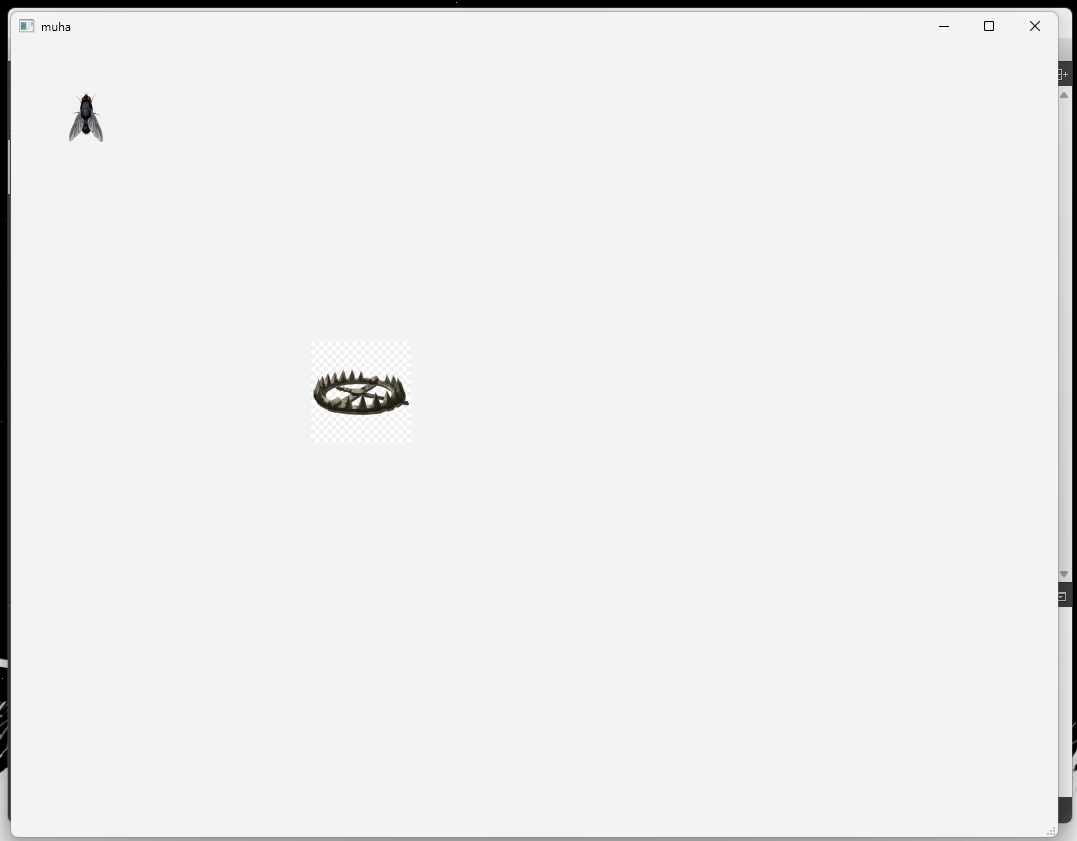
cerr << "error: " << e.what() << endl;

}

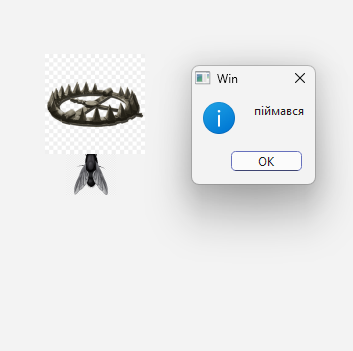
return 0;

}

**Результат:**

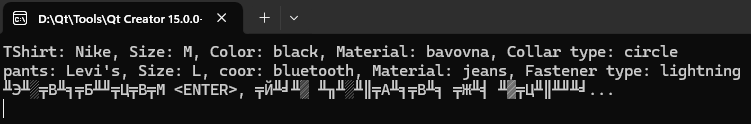
Після натиску Ctr + R викликається вікно з грою

Після доторкання мухи до границь пастки – виводеться вікно «гру завершенно» або як у мене «піймався»



Завдання 2

Варіант 8



**Висновок :**

**Завдання 1**

У цій лабораторній роботі було створено гру "Піймай муху" з використанням бібліотеки Qt. Основна мета полягала у тому, щоб продемонструвати роботу з графічним інтерфейсом користувача (GUI) в середовищі C++ та навчитися працювати з подіями миші.

**Завдання 2**

У даній лабораторній роботі було реалізовано систему керування одягом з використанням принципів об’єктно-орієнтованого програмування, зокрема, наслідування, поліморфізм та виняткові ситуації. Основним досягненням було створення базового абстрактного класу Clothing, який включає загальні характеристики для всіх типів одягу, та похідних класів TShirt і Pants, які розширюють функціонал для конкретних типів одягу.