Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

**ЛАБОРАТОРНА РОБОТА №4**

**Тема:** Поліморфізм. Обробка винятків

з дисципліни: Об’єктно-орієнтоване програмування

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Царенко С. О. |
| Перевірив: асистент  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++.

**Варіант 5**

**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»

**Завдання 2**

1. Розробіть систему керування університетом, яка включатиме обробку різних типів студентів та викладачів. Кожен тип особи має свої характеристики, які потрібно реалізувати за допомогою наслідування, сетерів та гетерів. Кожна особа (студент або викладач) має наступні характеристики:

• Ім'я

• Вік

• Стать

Кожен тип особи має власні додаткові характеристики:

Студент:

• Курс

• Спеціальність

Викладач:

• Посада

• Предмет, який він викладає

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

3. Створіть похідні класи Student та Professor, які успадковуються від класу Person. 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 вік або порожнє ім'я.

**Завдання 1.**

***Mainwindow.h:***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QLabel>

#include <QTimer>

#include <QMouseEvent>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

protected:

void mouseMoveEvent(QMouseEvent \*event) override;

private:

Ui::MainWindow \*ui;

QLabel \*fly;

QLabel \*trap;

QTimer \*moveTimer;

void moveFly();

bool isCaught();

};

#endif // MAINWINDOW\_H

***Mainwindow.cpp:***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <QPixmap>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow) {

ui->setupUi(this);

setFixedSize(800, 600);

fly = new QLabel(this);

fly->setGeometry(100, 100, 80, 80);

fly->setAlignment(Qt::AlignCenter);

trap = new QLabel(this);

trap->setGeometry(650, 250, 120, 80);

trap->setAlignment(Qt::AlignCenter);

moveTimer = new QTimer(this);

connect(moveTimer, &QTimer::timeout, this, &MainWindow::moveFly);

moveTimer->start(90);

setMouseTracking(true);

QPixmap flyPixmap("D:/Стас/Универ/OOP/OOP\_Lab4/Task1/fly.png");

QPixmap trapPixmap("D:/Стас/Универ/OOP/OOP\_Lab4/Task1/trap.jfif");

flyPixmap = flyPixmap.scaled(120, 120, Qt::KeepAspectRatio);

fly->setPixmap(flyPixmap);

trapPixmap = trapPixmap.scaled(120, 120, Qt::KeepAspectRatio);

trap->setPixmap(trapPixmap);

}

MainWindow::~MainWindow() {

delete ui;

}

void MainWindow::mouseMoveEvent(QMouseEvent \*event) {

if (fly->geometry().contains(event->pos())) {

moveFly();

}

}

void MainWindow::moveFly() {

static QPoint prevPos;

QPoint cursorPos = mapFromGlobal(QCursor::pos());

if (prevPos == cursorPos) {

return;

}

prevPos = cursorPos;

int x = fly->x();

int y = fly->y();

int dx = (x > cursorPos.x()) ? 10 : -10;

int dy = (y > cursorPos.y()) ? 10 : -10;

x += dx;

y += dy;

x = std::clamp(x, 0, width() - fly->width());

y = std::clamp(y, 0, height() - fly->height());

fly->move(x, y);

if (isCaught()) {

moveTimer->stop();

QMessageBox::information(this, "Гра завершена", "Ви піймали муху!");

}

}

bool MainWindow::isCaught() {

return fly->geometry().intersects(trap->geometry());

}

***Main.cpp:***

#include "mainwindow.h"

#include <QApplication>

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

QApplication a(argc, argv);

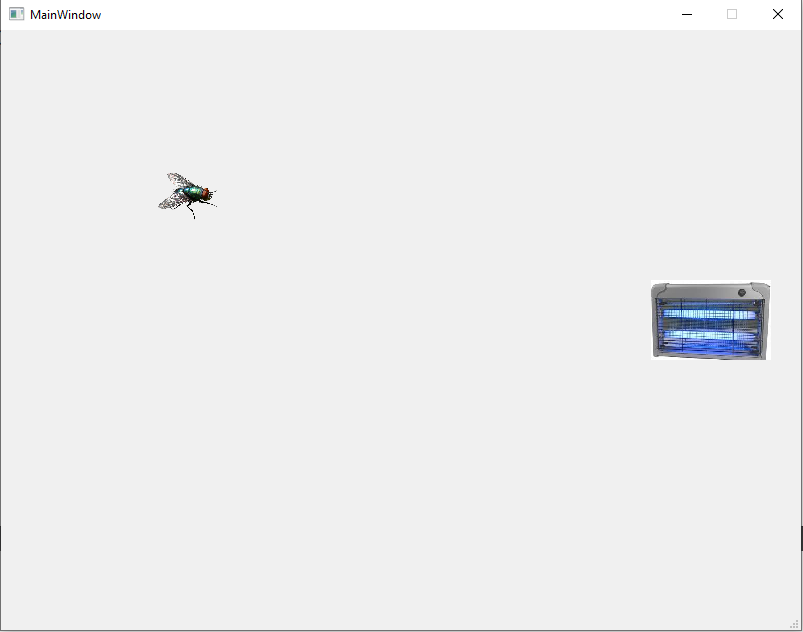
MainWindow w;

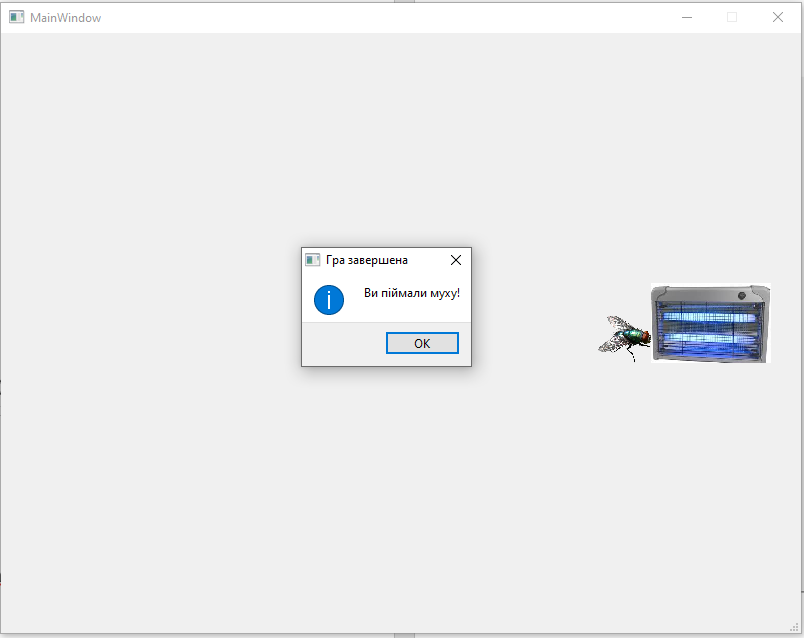
w.show();

return a.exec();

}

***Скріншот:***

******

****

**Завдання 2.**

***Person.h:***

#ifndef PERSON\_H

#define PERSON\_H

#include <string>

#include <iostream>

using namespace std;

class Person {

protected:

string name;

int age;

char gender;

public:

virtual ~Person() {}

virtual void displayInfo() = 0;

void setName(const string& name);

void setAge(int age);

void setGender(char gender);

string getName();

int getAge();

char getGender();

};

#endif // PERSON\_H

***Person.cpp:***

#include "Person.h"

#include <iostream>

void Person::setName(const string& name) {

if (name.empty()) {

throw invalid\_argument("Name cannot be empty.");

}

this->name = name;

}

void Person::setAge(int age) {

if (age <= 0 || age >= 100) {

throw invalid\_argument("Invalid age. Age must be between 1 and 99.");

}

this->age = age;

}

void Person::setGender(char gender) {

if (gender != 'M' && gender != 'F') {

throw invalid\_argument("Invalid gender. Use 'M' for Male or 'F' for Female.");

}

this->gender = gender;

}

string Person::getName() {

return name;

}

int Person::getAge() {

return age;

}

char Person::getGender() {

return gender;

}

***Student.h:***

#ifndef STUDENT\_H

#define STUDENT\_H

#include "Person.h"

class Student : public Person {

private:

int course;

string specialty;

public:

Student(const string& name, int age, char gender, int course, const string& specialty);

void displayInfo() override;

void setCourse(int course);

void setSpecialty(const string& specialty);

int getCourse();

string getSpecialty();

};

#endif // STUDENT\_H

***Student.cpp:***

#include "Student.h"

#include <iostream>

Student::Student(const string& name, int age, char gender, int course, const string& specialty) {

setName(name);

setAge(age);

setGender(gender);

setCourse(course);

setSpecialty(specialty);

}

void Student::displayInfo() {

cout << "Student Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "Gender: " << gender << endl;

cout << "Course: " << course << endl;

cout << "Specialty: " << specialty << endl;

}

void Student::setCourse(int course) {

if (course < 1 || course > 6) {

throw invalid\_argument("Invalid course number. It must be between 1 and 6.");

}

this->course = course;

}

void Student::setSpecialty(const string& specialty) {

if (specialty.empty()) {

throw invalid\_argument("Specialty cannot be empty.");

}

this->specialty = specialty;

}

int Student::getCourse() {

return course;

}

string Student::getSpecialty() {

return specialty;

}

***Professor.h:***

#ifndef PROFESSOR\_H

#define PROFESSOR\_H

#include "Person.h"

class Professor : public Person {

private:

string position;

string subject;

public:

Professor(const string& name, int age, char gender, const string& position, const string& subject);

void displayInfo() override;

void setPosition(const string& position);

void setSubject(const string& subject);

string getPosition();

string getSubject();

};

#endif // PROFESSOR\_H

***Professor.cpp:***

#include "Professor.h"

#include <iostream>

Professor::Professor(const string& name, int age, char gender, const string& position, const string& subject) {

setName(name);

setAge(age);

setGender(gender);

setPosition(position);

setSubject(subject);

}

void Professor::displayInfo() {

cout << "Professor Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "Gender: " << gender << endl;

cout << "Position: " << position << endl;

cout << "Subject: " << subject << endl;

}

void Professor::setPosition(const string& position) {

if (position.empty()) {

throw invalid\_argument("Position cannot be empty.");

}

this->position = position;

}

void Professor::setSubject(const string& subject) {

if (subject.empty()) {

throw invalid\_argument("Subject cannot be empty.");

}

this->subject = subject;

}

string Professor::getPosition() {

return position;

}

string Professor::getSubject() {

return subject;

}

***Task2.cpp:***

#include <iostream>

#include "Student.h"

#include "Professor.h"

using namespace std;

int main() {

try {

try {

Student student("Ben", 20, 'M', 2, "Computer Science");

student.displayInfo();

}

catch (const invalid\_argument& e) {

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

}

cout << endl;

try {

Professor professor("Maria", 40, 'F', "Professor", "Mathematics");

professor.displayInfo();

}

catch (const invalid\_argument& e) {

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

}

} catch (const invalid\_argument& e) {

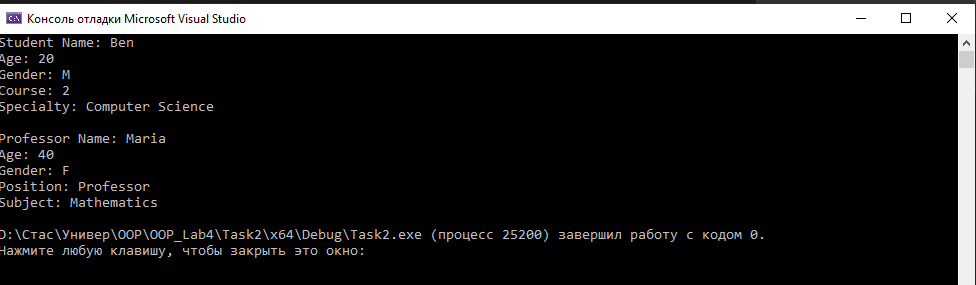
cerr << "Error: " << e.what() << endl;

}

return 0;

}

***Скріншот:***

****

**Висновок:**

У ході виконання лабораторної роботи №4 “ Поліморфізм. Обробка винятків”. Я ознайомився з поняттям поліморфізму у мові C++ та навчитись використовувати віртуальні функції для досягнення поліморфізму. Також вивчити принципи обробки винятків у мові C++.