Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

**ЛАБОРАТОРНА РОБОТА №5**

**Тема:** Перевантаження операторів

з дисципліни: Об’єктно-орієнтоване програмування

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Царенко С. О. |
| Перевірив: асистент  Козірова Н. Л. |
|  |

Кропивницький

2024

**Тема:** Перевантаження операторів

**Мета:** ознайомитись з поняттям перевантаження операторів та навчитись їх програмно реалізовувати мовою С++.

**Варіант 5**

***Завдання 1:***

Перевантажте оператори для завдання № 2 з лабораторної роботи №2.

***Завдання 2:***

Продовжить розробку гри «Спіймай муху», тепер ваша муха має стати класом, реалізуйте декілька рівнів гри, на кожному наступному рівні, має з’являтися нова муха, яка буде класом наслідником. Також, з другого рівня на вашій формі має з’явитися вікно, муха має намагатися вилетіти в це вікно, задача гравця відігнати муху від вікна і загнати її в пастку.

***Завдання 3:***

**Варіант 5**

Створіть клас Fraction, який представляє дріб у вигляді чисельника та знаменника. В класі Fraction перевантажте наступні оператори:

1. Оператор + для додавання двох дробів.
2. Оператор - для віднімання двох дробів.
3. Оператор \* для множення двох дробів.
4. Оператор / для ділення одного дробу на інший.
5. Оператор == для порівняння двох дробів на рівність.
6. Оператор != для порівняння двох дробів на нерівність.
7. Оператор << для виводу дробу у форматі "чисельник/знаменник".

Додайте в клас також необхідні конструктори, деструктор та інші методи, які можуть знадобитись для роботи з дробами.

Напишіть програму, де ви використовуєте цей клас та перевірте роботу всіх перевантажених операторів. Створіть декілька об'єктів класу Fraction і виконайте з ними операції додавання, віднімання, множення, ділення, порівняння на рівність та виведення на екран.

**Завдання 1.**

***MyString.h:***

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

class MyString {

private:

string str;

public:

MyString(const string& input = "");

MyString operator+(const MyString& other) const;

void removeCharAt(int pos);

bool operator==(const MyString& other) const;

friend ostream& operator<<(ostream& os, const MyString& obj);

bool operator<(const MyString& other) const;

static void sortArr(vector<MyString>& strings);

~MyString();

};

***MyString.cpp:***

#include "MyString.h"

MyString::MyString(const string& str) : str(str) {}

MyString MyString::operator+(const MyString& other) const {

return MyString(str + other.str);

}

void MyString::removeCharAt(int pos) {

str.erase(pos, 1);

}

bool MyString::operator==(const MyString& other) const {

return str == other.str;

}

MyString::~MyString() {}

void MyString::sortArr(vector<MyString>& strings) {

sort(strings.begin(), strings.end());

cout << "Sorted strings:" << endl;

for (const auto& str : strings) {

cout << str << endl;

}

}

bool MyString::operator<(const MyString& other) const {

return str < other.str;

}

ostream& operator<<(ostream& os, const MyString& obj) {

os << obj.str;

return os;

}

***Task1.cpp:***

#include <iostream>

#include <string>

#include <vector>

#include "MyString.h"

using namespace std;

int main() {

MyString s1("Alex");

MyString s2("Roberto");

cout << "String s1: " << s1 << endl;

cout << "String s2: " << s2 << endl;

MyString s3 = s1 + s2;

cout << "After concatenate: " << s3 << endl;

s3.removeCharAt(1);

cout << "After delete symb: " << s3 << endl;

if (s1 == s2) {

cout << "Strings are equal." << endl;

}

else {

cout << "Strings are not equal." << endl;

}

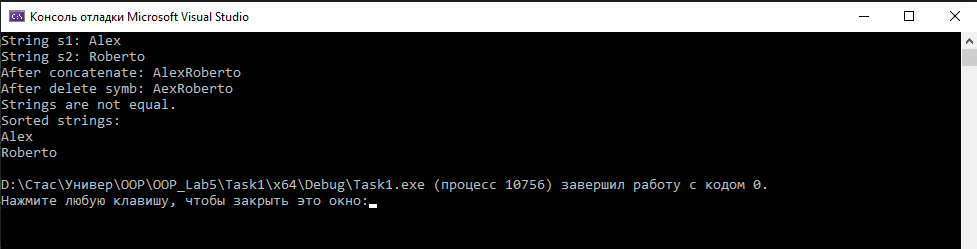
vector<MyString> strings = {s1, s2};

MyString::sortArr(strings);

return 0;

}

***Скріншот:***

****

**Завдання 2.**

***Mainwindow.h:***

*#ifndef MAINWINDOW\_H*

*#define MAINWINDOW\_H*

*#include <QMainWindow>*

*#include <QLabel>*

*#include <QTimer>*

*#include <QMouseEvent>*

*#include "level.h"*

*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 \*window;*

*QLabel \*trap;*

*Level \*currentFly;*

*QTimer \*moveTimer;*

*int currentLevel;*

*int movementPattern;*

*void moveFly();*

*bool isCaught();*

*void nextLevel();*

*};*

*#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), currentLevel(1), movementPattern(1) {

ui->setupUi(this);

setFixedSize(800, 600);

// пастка

trap = new QLabel(this);

trap->setGeometry(650, 400, 120, 80);

trap->setAlignment(Qt::AlignCenter);

QPixmap trapPixmap("D:/Стас/Универ/OOP/OOP\_Lab5/Task2/trap.jfif");

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

trap->setPixmap(trapPixmap);

// окно

window = new QLabel(this);

window->setGeometry(650, 150, 120, 80);

window->setAlignment(Qt::AlignCenter);

QPixmap windowPixmap("D:/Стас/Универ/OOP/OOP\_Lab5/Task2/window.jpg");

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

window->setPixmap(windowPixmap);

moveTimer = new QTimer(this);

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

moveTimer->start(90);

setMouseTracking(true);

currentFly = new Level(this, "D:/Стас/Универ/OOP/OOP\_Lab5/Task2/fly.png", currentLevel);

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

currentFly->show();

}

MainWindow::~MainWindow() {

delete ui;

delete trap;

delete window;

delete currentFly;

delete moveTimer;

}

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

QMainWindow::mouseMoveEvent(event);

}

void MainWindow::moveFly() {

currentFly->moveFly();

if (currentFly->geometry().intersects(window->geometry())) {

moveTimer->stop();

QMessageBox::information(this, "Гра завершена", "Ви програли! Муха вилетіла в вікно.");

close();

return;

}

if (isCaught()) {

moveTimer->stop();

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

nextLevel();

}

}

bool MainWindow::isCaught() {

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

}

void MainWindow::nextLevel() {

delete currentFly;

currentLevel++;

if (currentLevel > 5) {

QMessageBox::information(this, "Гра завершена", "Всі рівні пройдено!");

close();

return;

}

QString flyImage = QString("D:/Стас/Универ/OOP/OOP\_Lab5/Task2/fly%1.jpg").arg(currentLevel);

currentFly = new Level(this, flyImage, currentLevel);

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

currentFly->show();

moveTimer->start(90);

}

***Fly.h:***

#ifndef FLY\_H

#define FLY\_H

#include <QLabel>

#include <QString>

class Fly : public QLabel {

Q\_OBJECT

public:

explicit Fly(QWidget \*parent = nullptr, const QString &imagePath = "");

virtual ~Fly();

virtual void moveFly() = 0;

};

#endif // FLY\_H

***Fly.cpp:***

#include "fly.h"

#include <QPixmap>

Fly::Fly(QWidget \*parent, const QString &imagePath) : QLabel(parent) {

setPixmap(QPixmap(imagePath).scaled(80, 80, Qt::KeepAspectRatio));

setAlignment(Qt::AlignCenter);

}

Fly::~Fly() {}

void Fly::moveFly() {

}

***level.h:***

#ifndef LEVEL\_H

#define LEVEL\_H

#include "fly.h"

class Level : public Fly {

Q\_OBJECT

public:

explicit Level(QWidget \*parent = nullptr, const QString &imagePath = "", int level = 1);

void moveFly() override;

private:

int currentLevel;

};

#endif // LEVEL\_H

***level.cpp:***

#include "level.h"

#include <QCursor>

Level::Level(QWidget \*parent, const QString &imagePath, int level)

: Fly(parent, imagePath), currentLevel(level) {}

void Level::moveFly() {

static QPoint prevPos;

QPoint cursorPos = parentWidget()->mapFromGlobal(QCursor::pos());

if (prevPos == cursorPos) {

return;

}

prevPos = cursorPos;

int x = this->x();

int y = this->y();

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

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

switch (currentLevel) {

case 1:

break; // Вправо-вниз

case 2:

dx = -dx;

break; // Вліво-вниз

case 3:

dx = -dx;

dy = -dy;

break; // Вліво-вгору

case 4:

dy = -dy;

break; // Вправо-вгору

case 5:

break;

default:

break;

}

x = std::clamp(x + dx, 0, parentWidget()->width() - this->width());

y = std::clamp(y + dy, 0, parentWidget()->height() - this->height());

this->move(x, y);

}

***Main.cpp:***

#include "mainwindow.h"

#include <QApplication>

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

QApplication a(argc, argv);

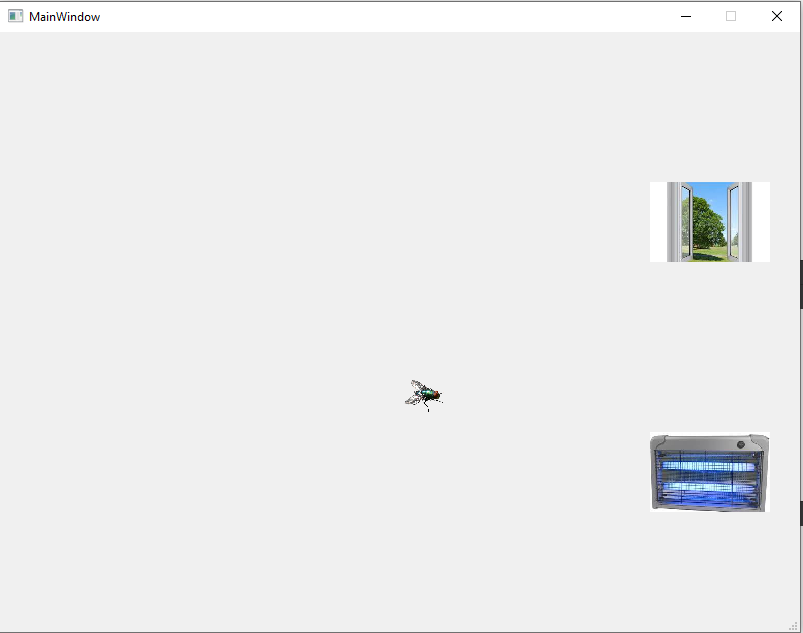
MainWindow w;

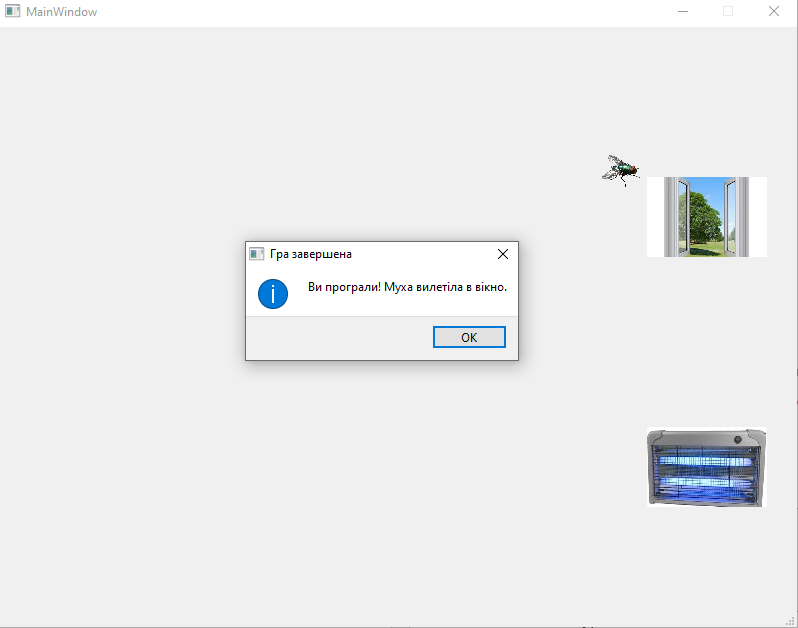
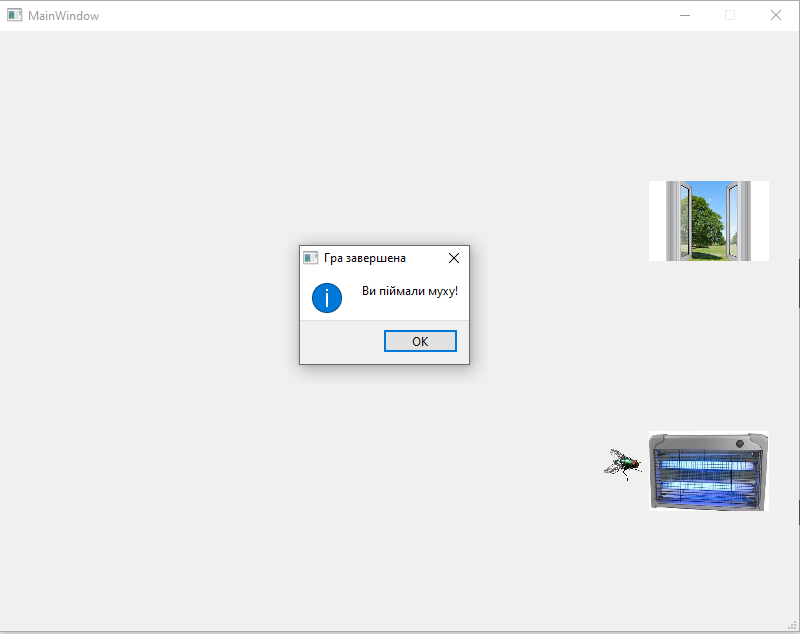
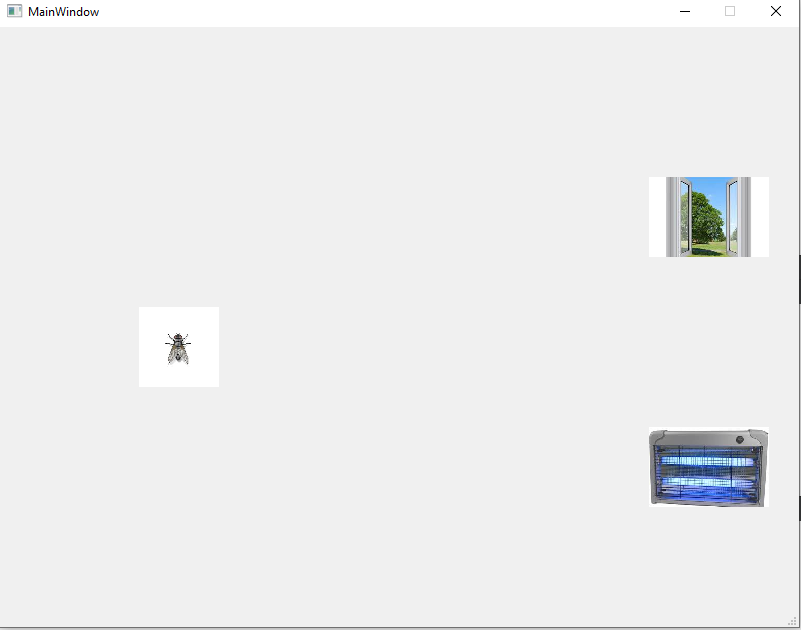
w.show();

return a.exec();

}

***Скріншот:***

****

**  
  
** ****

**Завдання 3.**

***Fraction.h:***

#ifndef FRACTION\_H

#define FRACTION\_H

#include <iostream>

using namespace std;

class Fraction {

private:

int numerator;

int denominator;

public:

Fraction(int num = 1, int denom = 2);

~Fraction();

Fraction operator+(const Fraction& other) const;

Fraction operator-(const Fraction& other) const;

Fraction operator\*(const Fraction& other) const;

Fraction operator/(const Fraction& other) const;

bool operator==(const Fraction& other) const;

bool operator!=(const Fraction& other) const;

friend ostream& operator<<(ostream& os, const Fraction& frac);

};

#endif // FRACTION\_H

***Fraction.cpp:***

#include "Fraction.h"

#include <stdexcept>

Fraction::Fraction(int num, int denom) : numerator(num), denominator(denom) {

if (denominator == 0)

throw invalid\_argument("Denominator cannot be zero.");

}

Fraction::~Fraction() {}

Fraction Fraction::operator+(const Fraction& other) const {

return Fraction(numerator \* other.denominator + other.numerator \* denominator,

denominator \* other.denominator);

}

Fraction Fraction::operator-(const Fraction& other) const {

return Fraction(numerator \* other.denominator - other.numerator \* denominator,

denominator \* other.denominator);

}

Fraction Fraction::operator\*(const Fraction& other) const {

return Fraction(numerator \* other.numerator, denominator \* other.denominator);

}

Fraction Fraction::operator/(const Fraction& other) const {

if (other.numerator == 0)

throw invalid\_argument("Cannot divide by zero fraction.");

return Fraction(numerator \* other.denominator, denominator \* other.numerator);

}

bool Fraction::operator==(const Fraction& other) const {

return numerator == other.numerator && denominator == other.denominator;

}

bool Fraction::operator!=(const Fraction& other) const {

return !(\*this == other);

}

ostream& operator<<(ostream& os, const Fraction& frac) {

os << frac.numerator << '/' << frac.denominator;

return os;

}

***Task3.cpp:***

#include <iostream>

#include "Fraction.h"

using namespace std;

int main() {

try {

Fraction f1(1, 2);

Fraction f2(3, 4);

Fraction f3(-2, 5);

cout << "f1: " << f1 << "\n";

cout << "f2: " << f2 << "\n";

cout << "f3: " << f3 << "\n";

cout << "f1 + f2 = " << f1 + f2 << "\n";

cout << "f1 - f2 = " << f1 - f2 << "\n";

cout << "f1 \* f3 = " << f1 \* f3 << "\n";

cout << "f1 / f2 = " << f1 / f2 << "\n";

if (f1 == f2) {

cout << "f1 == f2: true\n";

}

else {

cout << "f1 == f2: false\n";

}

if (f1 != f3) {

cout << "f1 != f3: true\n";

}

else {

cout << "f1 != f3: false\n";

}

}

catch (const exception& e) {

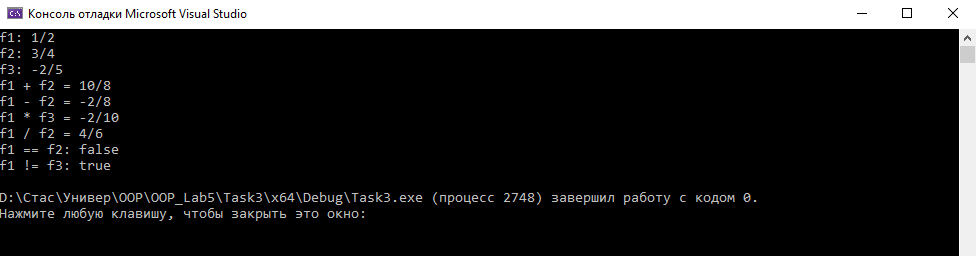
cerr << "Error: " << e.what() << "\n";

}

return 0;

}

***Скріншот:***

****

**Висновок:**

У ході виконання лабораторної роботи №5 “ Перевантаження операторів ”. Я ознайомився з поняттям перевантаження операторів та навчився їх програмно реалізовувати мовою С++.