Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №5**

**Тема: «Перевантаження операторів»**

|  |
| --- |
| Виконав: ст. гр. КI-23 |
| Карпець О. Р. |
| Перевірив викладач:  Козірова Н. Л. |
|  |

Кропивницький 2024

**Тема:** Перевантаження операторів

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

**Варіант 8**

**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**

Створіть клас Rectangle, який представляє прямокутник з заданими шириною та висотою. Ширина та висота повинні бути додатними цілими числами. В класі Rectangle перевантажте наступні оператори:

1. Оператор + для об'єднання двох прямокутників. Результатом має бути новий прямокутник, який охоплює обидва початкових прямокутники.
2. Оператор - для віднімання одного прямокутника від іншого. Результатом має бути новий прямокутник, який представляє різницю між початковими прямокутниками.
3. Оператор \* для масштабування прямокутника. Передавайте ціле число як аргумент, і результатом має бути новий прямокутник, який має ширину та висоту, помножені на це число.
4. Оператор / для зменшення прямокутника. Передавайте ціле число як аргумент, і результатом має бути новий прямокутник, який має ширину та висоту, поділені на це число.
5. Оператор == для порівняння двох прямокутників на рівність.
6. Оператор != для порівняння двох прямокутників на нерівність.
7. Оператор << для виводу прямокутника у зрозумілому форматі.

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

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

**Завдання 1**

**Лістинг set.cpp**

#include "set.h"

Set::Set() {}

Set::Set(string elems)

{

for (char elem : elems)

{

if (!contains(elem))

{

elements.push\_back(elem);

}

}

}

bool Set::contains(char checkElem) const

{

for (char elem : elements)

{

if (elem == checkElem)

{

return true;

}

}

return false;

}

Set Set::operator+(const Set& otherSet) const

{

Set result = \*this;

for (char elem : otherSet.elements)

{

if (!result.contains(elem))

{

result.elements.push\_back(elem);

}

}

return result;

}

Set Set::operator\*(const Set& otherSet) const

{

Set result;

for (char elem : elements)

{

if (otherSet.contains(elem))

{

result.elements.push\_back(elem);

}

}

return result;

}

Set Set::operator-(const Set& otherSet) const

{

Set result;

for (char elem : elements)

{

if (!otherSet.contains(elem))

{

result.elements.push\_back(elem);

}

}

return result;

}

Set Set::operator^(const Set& otherSet) const

{

Set result;

for (char elem : elements)

{

if (!otherSet.contains(elem))

{

result.elements.push\_back(elem);

}

}

for (char elem : otherSet.elements)

{

if (!contains(elem))

{

result.elements.push\_back(elem);

}

}

return result;

}

ostream& operator<<(ostream& os, const Set& set)

{

os << "{ ";

for (char elem : set.elements) {

os << elem << " ";

}

os << "}";

return os;

}

**Лістинг set.h**

#ifndef SET\_H

#define SET\_H

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Set {

private:

vector<char> elements;

public:

Set();

Set(string elems);

bool contains(char checkElem) const;

Set operator+(const Set& otherSet) const;

Set operator\*(const Set& otherSet) const;

Set operator-(const Set& otherSet) const;

Set operator^(const Set& otherSet) const;

friend ostream& operator<<(ostream& os, const Set& set);

};

#endif // SET\_H

**Лістинг main.cpp**

#include <iostream>

#include "set.h"

using namespace std;

int main()

{

Set set1("abc");

Set set2("bcd");

Set unionSet = set1 + set2;

Set intersectionSet = set1 \* set2;

Set differenceSet = set1 - set2;

Set symmetricDifferenceSet = set1 ^ set2;

cout << "Union: " << unionSet << endl;

cout << "Intersection: " << intersectionSet << endl;

cout << "Difference: " << differenceSet << endl;

cout << "Symmetric Difference: " << symmetricDifferenceSet << endl;

return 0;

}

**Завдання 2**

**Лістинг catchflygame.cpp**

#include "catchflygame.h"

#include <QMessageBox>

#include <QRandomGenerator>

CatchFlyGame::CatchFlyGame(QWidget \*parent)

: QMainWindow(parent), level(1) {

setFixedSize(800, 600);

setWindowTitle("Fly Game");

trap = new QLabel(this);

trap->setPixmap(QPixmap(":/photo/trap.png").scaled(100, 100, Qt::KeepAspectRatio));

trap->setGeometry(350, 350, 100, 100);

window = new QLabel(this);

window->setPixmap(QPixmap(":/photo/window.png").scaled(150, 150, Qt::KeepAspectRatio));

window->setGeometry(600, 100, 150, 150);

window->hide();

timer = new QTimer(this);

connect(timer, &QTimer::timeout, this, &CatchFlyGame::moveFly);

startLevel(1);

}

void CatchFlyGame::startLevel(int level) {

this->level = level;

delete fly;

if (level == 1) {

fly = new Level1Fly(this);

} else if (level == 2) {

fly = new Level2Fly(this);

window->show();

}

fly->setGeometry(100, 100, 50, 50);

fly->show();

timer->start(100);

}

void CatchFlyGame::mousePressEvent(QMouseEvent \*event) {

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

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

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

fly->move(newX, newY);

}

QMainWindow::mousePressEvent(event);

}

void CatchFlyGame::moveFly() {

if (level == 2 && fly->geometry().intersects(window->geometry())) {

dynamic\_cast<Level2Fly\*>(fly)->moveToWindow(window->geometry());

} else {

fly->moveRandomly();

}

if (fly->isInTrap(trap->geometry())) {

timer->stop();

QMessageBox::information(this, "Game Over", "You caught the fly!");

if (level < 2) {

startLevel(level + 1);

} else {

QMessageBox::information(this, "Congratulations!", "You've completed all levels!");

}

}

}

**Лістинг catchflygame.h**

#ifndef CATCHFLYGAME\_H

#define CATCHFLYGAME\_H

#include <QMainWindow>

#include <QMouseEvent>

#include <QLabel>

#include <QTimer>

#include "fly.h"

class CatchFlyGame : public QMainWindow {

Q\_OBJECT

public:

CatchFlyGame(QWidget \*parent = nullptr);

void startLevel(int level);

protected:

void mousePressEvent(QMouseEvent \*event) override;

private slots:

void moveFly();

private:

Fly \*fly;

QLabel \*trap;

QLabel \*window;

QTimer \*timer;

int level;

};

#endif // CATCHFLYGAME\_H

**Лістинг fly.cpp**

#include "fly.h"

#include <QRandomGenerator>

Fly::Fly(QWidget \*parent) : QLabel(parent), speed(5) {}

void Fly::moveRandomly() {

int dx = QRandomGenerator::global()->bounded(-speed, speed);

int dy = QRandomGenerator::global()->bounded(-speed, speed);

move(x() + dx, y() + dy);

}

bool Fly::isInTrap(const QRect &trapRect) {

return this->geometry().intersects(trapRect);

}

Level1Fly::Level1Fly(QWidget \*parent) : Fly(parent) {

setPixmap(QPixmap(":/photo/fly.png").scaled(50, 50, Qt::KeepAspectRatio));

speed = 5;

}

void Level1Fly::moveRandomly() {

Fly::moveRandomly();

}

Level2Fly::Level2Fly(QWidget \*parent) : Fly(parent) {

setPixmap(QPixmap(":/photo/fly2.png").scaled(50, 50, Qt::KeepAspectRatio));

speed = 8;

}

void Level2Fly::moveRandomly() {

Fly::moveRandomly();

}

void Level2Fly::moveToWindow(const QRect &windowRect) {

int dx = (windowRect.x() - x()) / 10;

int dy = (windowRect.y() - y()) / 10;

move(x() + dx, y() + dy);

}

**Лістинг fly.h**

#ifndef FLY\_H

#define FLY\_H

#include <QLabel>

#include <QTimer>

#include <QWidget>

class Fly : public QLabel {

Q\_OBJECT

public:

Fly(QWidget \*parent = nullptr);

virtual void moveRandomly();

bool isInTrap(const QRect &trapRect);

protected:

int speed;

};

class Level1Fly : public Fly {

Q\_OBJECT

public:

Level1Fly(QWidget \*parent = nullptr);

void moveRandomly() override;

};

class Level2Fly : public Fly {

Q\_OBJECT

public:

Level2Fly(QWidget \*parent = nullptr);

void moveRandomly() override;

void moveToWindow(const QRect &windowRect);

};

#endif // FLY\_H

**Лістинг main.cpp**

#include <QApplication>

#include "catchflygame.h"

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

QApplication app(argc, argv);

CatchFlyGame game;

game.show();

return app.exec();

}

**Завдання 3**

**Лістинг rectangle.cpp**

#include "rectangle.h"

#include <stdexcept>

using namespace std;

Rectangle::Rectangle(int width, int height) : width(width), height(height) {

if (width <= 0 || height <= 0) {

throw invalid\_argument("Width and height must be positive int.");

}

}

Rectangle::Rectangle(Rectangle &other) : width(other.width), height(other.height) {}

Rectangle::~Rectangle() {}

Rectangle Rectangle::operator+(Rectangle &other) {

int newWidth = max(this->width, other.width);

int newHeight = max(this->height, other.height);

return Rectangle(newWidth, newHeight);

}

Rectangle Rectangle::operator-(Rectangle &other) {

int newWidth = max(0, this->width - other.width);

int newHeight = max(0, this->height - other.height);

return Rectangle(newWidth, newHeight);

}

Rectangle Rectangle::operator\*(int factor) {

return Rectangle(this->width \* factor, this->height \* factor);

}

Rectangle Rectangle::operator/(int divisor) {

if (divisor <= 0) throw invalid\_argument("Divisor must be a positive int.");

return Rectangle(this->width / divisor, this->height / divisor);

}

bool Rectangle::operator==(Rectangle &other) {

return this->width == other.width && this->height == other.height;

}

bool Rectangle::operator!=(Rectangle &other) {

return !(\*this == other);

}

ostream &operator<<(ostream &out, Rectangle &rect) {

out << "Rectangle(" << rect.width << " x " << rect.height << ")";

return out;

}

**Лістинг rectangle.h**

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include <iostream>

class Rectangle {

public:

Rectangle(int width = 1, int height = 1);

Rectangle(Rectangle &other);

~Rectangle();

Rectangle operator+(Rectangle &other);

Rectangle operator-(Rectangle &other);

Rectangle operator\*(int factor);

Rectangle operator/(int divisor);

bool operator==(Rectangle &other);

bool operator!=(Rectangle &other);

friend std::ostream &operator<<(std::ostream &out, Rectangle &rect);

private:

int width;

int height;

};

#endif // RECTANGLE\_H

**Лістинг main.cpp**

#include <QCoreApplication>

#include "rectangle.h"

using namespace std;

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

{

QCoreApplication a(argc, argv);

try {

Rectangle rect1(10, 5);

Rectangle rect2(7, 3);

Rectangle rect3(10, 5);

Rectangle unionRect = rect1 + rect2;

cout << "Union rect1 and rect2: " << unionRect << endl;

Rectangle diffRect = rect1 - rect2;

cout << "Difference rect1 and rect2: " << diffRect << endl;

Rectangle scaledRect = rect1 \* 2;

cout << "Scaling rect1 by factor 2: " << scaledRect << endl;

Rectangle reducedRect = rect1 / 2;

cout << "Reducing rect1 by divisor 2: " << reducedRect << endl;

cout << "rect1 is equal to rect3: " << (rect1 == rect3 ? "true" : "false") << endl;

cout << "rect1 is not equal to rect2: " << (rect1 != rect2 ? "true" : "false") << endl;

} catch (const exception &e) {

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

}

return a.exec();

}

**Висновок**

У ході лабораторної роботи я ознайомився з перевантаженням операторів у C++. Реалізував завдання з використанням перевантажених операторів для класів Set і Rectangle. Також створив гру "Спіймай муху", застосувавши принципи ООП та перевантаження функціоналу для роботи з графічними об'єктами.

Здобуті знання дозволили мені зрозуміти, як підвищити гнучкість і зручність роботи з об'єктами в C++, використовуючи перевантаження операторів.