Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №6**

**Тема: «Шаблони в С++»**

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

Кропивницький 2024

**Тема:** шаблони в С++

**Мета:**ознайомитись з основними поняттями шаблони та навчитись їх програмно реалізовувати мовою С++.

**Варіант 8**

**Завдання 1**

Що ж, вітаю гра «Спіймай муху», доходить до свого логічного завершення. На даному етапі, вашій мусі нудно літати по пустій платформі, створіть для неї інтерфейс кімнати, в кімнаті уже є вікно. Також на другому рівні, має залишитись муха з першого рівня і прилетіти нова. Коли на другому рівні одна з мух вилітає у вікно або ловиться в пастку, настає третій рівень, на третьому рівні через відкриті двері до нас заскакує клас «Жаба», котра намагається спіймати муху, при цьому вікно закривається, а об’єкт «пастка» знищується . Додайте лічильник, щоб було цікавіше.

**Завдання 2**

Створи програму для керування музичним магазином. Музичний магазин продає різні музичні інструменти та аксесуари. Кожен інструмент має назву, тип та ціну. Необхідно створити клас Instrument, який містить ці дані.

Крім того, створи шаблонний клас MusicStore, який представляє музичний магазин. Цей клас має методи для додавання інструменту до асортименту магазину, видалення інструменту за назвою, виведення списку всіх доступних інструментів та пошуку інструменту за типом.

Забезпеч універсальність класу MusicStore, використовуючи шаблонний параметр для типу інструменту. Це дозволить тобі працювати з різними типами інструментів (наприклад, гітари, барабани, клавішні тощо).

Для демонстрації роботи програми:

1. Створи об'єкт MusicStore для зберігання інструментів.
2. Додай до магазину декілька різних інструментів.
3. Виведи на екран список всіх доступних інструментів у магазині.
4. Знайди та виведи на екран всі інструменти певного типу.
5. Видали з магазину інструмент за назвою.
6. Знову виведи на екран оновлений список доступних інструментів у магазині.

Не забудь використовувати шаблонний клас MusicStore для зберігання інструментів будь-якого типу. Також, забезпеч, щоб клас Instrument мав методи доступу до своїх полів (назва, тип, ціна).

**Завдання 1**

**Лістинг cathflygame.cpp:**

#include "catchflygame.h"

#include <QMessageBox>

#include <QRandomGenerator>

CatchFlyGame::CatchFlyGame(QWidget \*parent)

: QMainWindow(parent), level(1), score(0)

{

setFixedSize(800, 600);

setWindowTitle("Fly Game");

room = new QLabel(this);

room->setPixmap(QPixmap(":/photo/room.png").scaled(size(), Qt::KeepAspectRatioByExpanding));

room->setGeometry(0, 0, width(), height());

room->lower();

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();

door = new QLabel(this);

door->setPixmap(QPixmap(":/photo/door.png").scaled(150, 200, Qt::KeepAspectRatio));

door->setGeometry(100, 400, 150, 200);

door->hide();

scoreLabel = new QLabel(this);

scoreLabel->setText("Score: 0");

scoreLabel->setGeometry(10, 10, 100, 30);

timer = new QTimer(this);

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

startLevel(1);

}

void CatchFlyGame::mousePressEvent(QMouseEvent \*event)

{

for (Fly \*fly : flies)

{

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

{

score++;

updateScore();

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::startLevel(int level)

{

this->level = level;

qDeleteAll(flies);

flies.clear();

delete frog;

frog = nullptr;

window->setVisible(level > 1);

door->setVisible(level == 3);

trap->setVisible(level < 3);

if (level == 1)

{

Fly \*fly = new Level1Fly(this);

int x = QRandomGenerator::global()->bounded(width() - 50);

int y = QRandomGenerator::global()->bounded(height() - 50);

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

flies.append(fly);

fly->show();

}

else if (level == 2)

{

Fly \*fly1 = new Level1Fly(this);

int x1 = QRandomGenerator::global()->bounded(width() - 50);

int y1 = QRandomGenerator::global()->bounded(height() - 50);

fly1->setGeometry(x1, y1, 50, 50);

flies.append(fly1);

fly1->show();

Fly \*fly2 = new Level2Fly(this);

int x2 = QRandomGenerator::global()->bounded(width() - 50);

int y2 = QRandomGenerator::global()->bounded(height() - 50);

fly2->setGeometry(x2, y2, 50, 50);

flies.append(fly2);

fly2->show();

}

else if (level == 3)

{

for (int i = 0; i < 2; ++i)

{

Fly \*fly = new Level2Fly(this);

int x = QRandomGenerator::global()->bounded(width() - 50);

int y = QRandomGenerator::global()->bounded(height() - 50);

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

flies.append(fly);

fly->show();

}

frog = new Frog(this);

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

frog->show();

QTimer \*frogMoveTimer = new QTimer(this);

connect(frogMoveTimer, &QTimer::timeout, this, &CatchFlyGame::moveFrogSlowly);

frogMoveTimer->start(200);

}

timer->start(100);

}

void CatchFlyGame::moveFlies()

{

for (Fly \*fly : flies) {

if (level == 3 && fly->geometry().intersects(door->geometry())) {

QMessageBox::information(this, "Game Over", "The fly escaped through the door!");

close();

return;

}

fly->moveRandomly();

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

timer->stop();

startLevel(level + 1);

return;

}

}

if (level == 3 && frog) {

frog->tryToCatchFly(flies);

if (frog->hasCaughtFly())

{

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

close();

}

}

}

void CatchFlyGame::moveFrogSlowly()

{

if (frog) {

int newX = frog->x() + QRandomGenerator::global()->bounded(-20, 21);

int newY = frog->y() + QRandomGenerator::global()->bounded(-20, 21);

newX = qMax(0, qMin(newX, width() - frog->width()));

newY = qMax(0, qMin(newY, height() - frog->height()));

frog->move(newX, newY);

}

}

void CatchFlyGame::updateScore()

{

scoreLabel->setText("Score: " + QString::number(score));

}

**Лістинг catchflygame.h:**

#ifndef CATCHFLYGAME\_H

#define CATCHFLYGAME\_H

#include <QMainWindow>

#include <QMouseEvent>

#include <QLabel>

#include <QTimer>

#include <QList>

#include "fly.h"

#include "frog.h"

class CatchFlyGame : public QMainWindow {

Q\_OBJECT

public:

CatchFlyGame(QWidget \*parent = nullptr);

void startLevel(int level);

void moveFrogSlowly();

protected:

void mousePressEvent(QMouseEvent \*event) override;

private slots:

void moveFlies();

private:

QList<Fly\*> flies;

QLabel \*trap;

QLabel \*window;

QLabel \*room;

QLabel \*door;

Frog \*frog;

QTimer \*timer;

int level;

int flyCount;

int score;

QLabel \*scoreLabel;

void updateScore();

};

#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);

int newX = x() + dx;

int newY = y() + dy;

if (newX < 0) newX = 0;

if (newY < 0) newY = 0;

if (newX + width() > parentWidget()->width()) newX = parentWidget()->width() - width();

if (newY + height() > parentWidget()->height()) newY = parentWidget()->height() - height();

move(newX, newY);

}

bool Fly::isInTrap(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(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(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(QRect windowRect);

};

#endif // FLY\_H

**Лістинг frog.cpp:**

#include <QRandomGenerator>

#include "frog.h"

#include "fly.h"

Frog::Frog(QWidget \*parent)

: QLabel(parent), caughtFly(false)

{

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

}

bool Frog::hasCaughtFly()

{

return caughtFly;

}

void Frog::tryToCatchFly(QList<Fly\*> flies)

{

for (Fly \*fly : flies)

{

if (this->geometry().intersects(fly->geometry()))

{

caughtFly = true;

fly->hide();

break;

}

}

}

**Лістинг frog.h:**

#ifndef FROG\_H

#define FROG\_H

#include <QLabel>

#include "fly.h"

class Frog : public QLabel {

Q\_OBJECT

public:

Frog(QWidget \*parent = nullptr);

bool hasCaughtFly();

void tryToCatchFly(QList<Fly\*> flies);

private:

bool caughtFly;

};

#endif // FROG\_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 2**

**Лістинг drum.cpp:**

#include "drum.h"

Drum::Drum(QString name, double price)

: Instrument(name, "Drum", price) {}

QString Drum::getDescription()

{

return QString("Drum: %1 - $%2").arg(name).arg(price);

}

Drum::Drum() {}

**Лістинг drum.h:**

#ifndef DRUM\_H

#define DRUM\_H

#include "instrument.h"

class Drum : public Instrument {

public:

Drum(QString name, double price);

QString getDescription() override;

Drum();

};

#endif // DRUM\_H

**Лістинг guitar.cpp:**

#include "guitar.h"

Guitar::Guitar(QString name, double price)

: Instrument(name, "Guitar", price) {}

QString Guitar::getDescription()

{

return QString("Guitar: %1 - $%2").arg(name).arg(price);

}

Guitar::Guitar() {}

**Лістинг guitar.h:**

#ifndef GUITAR\_H

#define GUITAR\_H

#include "instrument.h"

class Guitar : public Instrument {

public:

Guitar(QString name, double price);

QString getDescription() override;

Guitar();

};

#endif // GUITAR\_H

**Лістинг instrument.cpp:**

#include "instrument.h"

Instrument::Instrument(QString name, QString type, double price)

: name(name), type(type), price(price) {}

QString Instrument::getName()

{

return name;

}

QString Instrument::getType()

{

return type;

}

double Instrument::getPrice()

{

return price;

}

QString Instrument::getDescription()

{

return QString("%1 (%2) - $%3").arg(name).arg(type).arg(price);

}

Instrument::Instrument() {}

**Лістинг instrument.h:**

#ifndef INSTRUMENT\_H

#define INSTRUMENT\_H

#include <QString>

class Instrument {

protected:

QString name;

QString type;

double price;

public:

Instrument(QString name, QString type, double price);

QString getName();

QString getType();

double getPrice();

virtual QString getDescription();

virtual ~Instrument() {}

Instrument();

};

#endif // INSTRUMENT\_H

**Лістинг musicstore.h:**

#ifndef MUSICSTORE\_H

#define MUSICSTORE\_H

#include <QList>

#include <QString>

#include <QDebug>

template <typename T>

class MusicStore

{

public:

void addInstrument(T\* instrument)

{

instruments.append(instrument);

}

void removeInstrument(QString name)

{

for (int i = 0; i < instruments.size(); ++i)

{

if (instruments[i]->getName() == name)

{

delete instruments.takeAt(i);

return;

}

}

}

QList<T\*> getInstrumentsByType(QString type)

{

QList<T\*> result;

for (const auto& instrument : instruments)

{

if (instrument->getType() == type)

{

result.append(instrument);

}

}

return result;

}

void listAllInstruments()

{

for (const auto& instrument : instruments)

{

qDebug() << instrument->getDescription();

}

}

~MusicStore()

{

qDeleteAll(instruments);

}

private:

QList<T\*> instruments;

};

#endif // MUSICSTORE\_H

**Лістинг main.cpp:**

#include <QCoreApplication>

#include "musicstore.h"

#include "guitar.h"

#include "drum.h"

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

QCoreApplication a(argc, argv);

MusicStore<Instrument> store;

store.addInstrument(new Guitar("Fender Stratocaster", 1200.00));

store.addInstrument(new Drum("Yamaha Drum Set", 800.00));

store.addInstrument(new Guitar("Gibson Les Paul", 1500.00));

qDebug() << "All Instruments:";

store.listAllInstruments();

qDebug() << "\nGuitars:";

auto guitars = store.getInstrumentsByType("Guitar");

for (const auto& guitar : guitars)

{

qDebug() << guitar->getDescription();

}

store.removeInstrument("Fender Stratocaster");

qDebug() << "\nAfter removing Fender Stratocaster:";

store.listAllInstruments();

return a.exec();

}

**Висновок**

У ході виконання лабораторної роботи ознайомився з шаблонами в C++ та їх використанням для створення універсальних рішень. Я реалізував шаблонний клас MusicStore, який забезпечує гнучке управління музичними інструментами різних типів (гітари, барабани).

Під час роботи:

* Застосував принципи ООП (інкапсуляція, успадкування).
* Освоїв створення та використання шаблонів класів.
* Переконався в ефективності шаблонів для написання гнучкого та перевикористовуваного коду.

Шаблони значно підвищують гнучкість і якість програмного забезпечення.