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 |
| Чепіль В.О. |
| 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:** 4

**Завдання:**

***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 складом товарів у магазині. Кожен товар має назву, ціну та кількість на складі. Необхідно створити клас Product, який містить ці дані.

Крім того, створи шаблонний клас Inventory, який представляє інвентар магазину. Цей клас має методи для додавання товару на склад, видалення товару зі складу за назвою, виведення списку всіх товарів та пошуку товару за ціною.

Забезпеч універсальність класу Inventory, використовуючи шаблонний параметр для типу товару. Це дозволить тобі працювати з різними типами товарів (наприклад, продукти харчування, електроніка, одяг тощо).

Для демонстрації роботи програми:

1. Створи об'єкт Inventory для зберігання товарів на складі.
2. Додай до інвентаря декілька товарів різних типів.
3. Виведи на екран список всіх товарів на складі.
4. Знайди та виведи на екран всі товари за певною ціною.
5. Видали з інвентаря товар за назвою.
6. Знову виведи на екран оновлений список товарів на складі.

Не забудь використовувати шаблонний клас Inventory для зберігання товарів будь-якого типу. Також, забезпеч, щоб клас Product мав методи доступу до своїх полів (назва, ціна, кількість).

**Лістинг завдання 1(task1):**

**#ifndef FLY\_H**

#define FLY\_H

#include <QLabel>

#include <QCursor>

#include <QPixmap>

#include <QPoint>

#include <QRandomGenerator>

class Fly : public QLabel

{

Q\_OBJECT

public:

Fly(QWidget\* parent = nullptr);

virtual void moveRandomly();

void moveFly();

};

#endif // FLY\_H

**#include "fly.h"**

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

setPixmap(QPixmap(":/pictures/fly.png").scaled(24, 24));

setGeometry(200, 600, 24, 24);

}

void Fly::moveRandomly() {

int X = QRandomGenerator::global()->bounded(-5, 5);

int Y = QRandomGenerator::global()->bounded(-5, 5);

int newX = qBound(0, x() + X, parentWidget()->width() - width());

int newY = qBound(0, y() + Y, parentWidget()->height() - height());

move(newX, newY);

}

void Fly::moveFly()

{

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

int mouseX = cursorPos.x();

int mouseY = cursorPos.y();

int flyX = this->x();

int flyY = this->y();

int distanceX = mouseX - flyX;

int distanceY = mouseY - flyY;

int distanceSquared = distanceX \* distanceX + distanceY \* distanceY;

if (distanceSquared < 2500)

{

flyX += (distanceX < 0) ? 10 : -10;

flyY += (distanceY < 0) ? 10 : -10;

}

else

{

this->moveRandomly();

return;

}

flyX = qBound(0, flyX, parentWidget()->width() - width());

flyY = qBound(0, flyY, parentWidget()->height() - height());

this->move(flyX, flyY);

}

**#ifndef FROG\_H**

#define FROG\_H

#include <QLabel>

#include <QTimer>

#include <QKeyEvent>

#include <QPainter>

#include <QList>

#include "fly.h"

class Frog : public QLabel

{

Q\_OBJECT

public:

explicit Frog(QWidget\* parent = nullptr);

void keyPressEvent(QKeyEvent\* event) override;

int getCaughtFlies() const;

private:

int x;

int y;

int step;

bool tongueVisible;

bool isLeft;

QLabel\* frogTongue;

QTimer tongueTimer;

int caughtFlies;

void updateTonguePosition();

void checkFlyCollision();

};

**#endif // FROG\_H**

#include "frog.h"

#include <QDebug>

#include <QPixmap>

Frog::Frog(QWidget\* parent) : QLabel(parent), x(100), y(650), step(10), tongueVisible(false), isLeft(true)

{

setFocusPolicy(Qt::StrongFocus);

setPixmap(QPixmap(":/pictures/frog to left.png").scaled(36, 36));

setGeometry(x, y, 36, 36);

frogTongue = new QLabel(parent);

frogTongue->setPixmap(QPixmap(":/pictures/frog tongue to left.png").scaled(10, 70));

frogTongue->setGeometry(x + 5, y - 36, 10, 70);

frogTongue->hide();

connect(&tongueTimer, &QTimer::timeout, [this]() {

tongueVisible = false;

frogTongue->hide();

tongueTimer.stop();

});

}

void Frog::keyPressEvent(QKeyEvent \*event)

{

if (tongueVisible)

{

return;

}

switch (event->key())

{

case Qt::Key\_W:

if (y > 590)

{

y -= step;

}

break;

case Qt::Key\_S:

if (y < (730 - height()))

{

y += step;

}

break;

case Qt::Key\_A:

if (x > 0)

{

setPixmap(QPixmap(":/pictures/frog to left.png").scaled(36, 36));

frogTongue->setPixmap(QPixmap(":/pictures/frog tongue to left.png").scaled(10, 70));

isLeft = true;

x -= step;

}

break;

case Qt::Key\_D:

if (x < (1300 - width()))

{

setPixmap(QPixmap(":/pictures/frog to right.png").scaled(36, 36));

frogTongue->setPixmap(QPixmap(":/pictures/frog tongue to right.png").scaled(10, 70));

isLeft = false;

x += step;

}

break;

case Qt::Key\_X:

if (!tongueVisible)

{

tongueVisible = true;

frogTongue->show();

checkFlyCollision();

tongueTimer.start(1000);

}

break;

default:

QLabel::keyPressEvent(event);

}

move(x, y);

updateTonguePosition();

}

void Frog::updateTonguePosition()

{

if (isLeft)

{

frogTongue->move(x - 10, y - 70);

}

else

{

frogTongue->move(x + width() - 5, y - 70);

}

}

void Frog::checkFlyCollision() {

QList<Fly\*> flies = parentWidget()->findChildren<Fly\*>();

for (Fly\* fly : flies)

{

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

{

fly->hide();

caughtFlies++;

}

}

}

int Frog::getCaughtFlies() const

{

return caughtFlies;

}

#ifndef GAME\_H

#define GAME\_H

#include <QMainWindow>

#include <QPixmap>

#include <QString>

#include "level.h"

class Game : public QMainWindow {

Q\_OBJECT

public:

Game(QWidget\* parent = nullptr);

private slots:

void onLevelCompleted();

void gameOver();

private:

int currentLevelNumber;

Level\* currentLevel = nullptr;

QLabel\* backgroundLabel;

QCursor cursor;

void changeRoomBackground(const QString& imagePath);

void startLevel(int levelNumber);

};

#endif // GAME\_H

#include "game.h"

#include <QMessageBox>

#include "QApplication"

Game::Game(QWidget\* parent) : QMainWindow(parent), currentLevelNumber(1)

{

setFixedSize(1300, 730);

backgroundLabel = new QLabel(this);

changeRoomBackground(":/pictures/room1.png");

backgroundLabel->setGeometry(0, 0, this->width(), this->height());

backgroundLabel->lower();

QPixmap cursorImage = QPixmap(":pictures/flySwatter.png");

cursor = QCursor(cursorImage);

setCursor(cursor);

startLevel(currentLevelNumber);

}

void Game::changeRoomBackground(const QString& imagePath)

{

QPixmap newBackground(imagePath);

backgroundLabel->clear();

backgroundLabel->setPixmap(newBackground.scaled(this->size(), Qt::KeepAspectRatio, Qt::SmoothTransformation));

}

void Game::startLevel(int levelNumber) {

if (currentLevel)

{

currentLevel->deleteLater();

}

currentLevel = new Level(levelNumber, this);

setCentralWidget(currentLevel);

connect(currentLevel, &Level::levelCompleted, this, &Game::onLevelCompleted);

connect(currentLevel, &Level::levelFailed, this, &Game::gameOver);

currentLevel->show();

}

void Game::onLevelCompleted()

{

switch (currentLevelNumber)

{

case 1:

QMessageBox::information(this, "Рівень 1 пройдено", "Ви зловили муху! Переходьте на наступний рівень.");

break;

case 2:

QMessageBox::information(this, "Рівень 2 пройдено", "Ви зловили всіх мух, недавши вилетіти у вікно! Переходьте на наступний рівень.");

changeRoomBackground(":/pictures/room2.png");

QMessageBox::information(this, "Рівень 3 підготовка", "Ви граєте за жабу ваше завдання зловити надоїдливих мух. "

"Рухатись можете клавішами w,a,s,d, щоб спіймати муху язиком нажміть x, "

"наганяючи мух до жаби мухобойкою");

changeRoomBackground(":/pictures/room1.png");

break;

case 3:

QMessageBox::information(this, "Рівень 3 пройдено", "Ви пройшли гру!");

QApplication::quit();

break;

}

currentLevelNumber++;

startLevel(currentLevelNumber);

}

void Game::gameOver()

{

QMessageBox::information(this, "Рівень провалено", "Ви програли");

QApplication::quit();

}

#ifndef LEVEL\_H

#define LEVEL\_H

#include <QWidget>

#include <QLabel>

#include <QTimer>

#include "fly.h"

#include "smartfly.h"

#include "frog.h"

class Level : public QWidget {

Q\_OBJECT

public:

Level(int levelNumber, QWidget\* parent = nullptr);

signals:

void levelCompleted();

void levelFailed();

private slots:

void updateFliesPosition();

private:

int level;

QTimer\* timer;

QLabel\* trapLabel;

QLabel\* windowLabel;

Fly\* fly1;

SmartFly\* fly2;

Fly\* fly3;

Frog\* frog;

QRect openWindowZone;

bool checkIfFlyCaught();

void Level1();

void Level2();

void Level3();

void stopLevel();

void keyPressEvent(QKeyEvent \*event) override;

};

#endif // LEVEL\_H

#include "level.h"

#include <QMessageBox>

#include <QPixmap>

#include <QDebug>

Level::Level(int levelNumber, QWidget\* parent)

: QWidget(parent), level(levelNumber), trapLabel(nullptr),

windowLabel(nullptr), fly1(nullptr), fly2(nullptr), fly3(nullptr), frog(nullptr)

{

setFocusPolicy(Qt::StrongFocus);

setFocus();

switch (levelNumber)

{

case 1:

Level1();

break;

case 2:

Level2();

break;

case 3:

Level3();

break;

default:

QMessageBox::warning(this, "Error", "Invalid level number");

break;

}

timer = new QTimer(this);

connect(timer, &QTimer::timeout, this, &Level::updateFliesPosition);

timer->start(50);

}

bool Level::checkIfFlyCaught()

{

return (fly1 && fly1->geometry().intersects(trapLabel->geometry())) ||

(fly2 && fly2->geometry().intersects(trapLabel->geometry())) ||

(fly3 && fly3->geometry().intersects(trapLabel->geometry()));

}

void Level::Level1()

{

trapLabel = new QLabel(this);

trapLabel->setGeometry(625, 415, 80, 80);

QPixmap trapPixmap(":/pictures/trap.png");

trapLabel->setPixmap(trapPixmap.scaled(80, 80, Qt::KeepAspectRatio));

if (!fly1)

{

fly1 = new Fly(this);

}

}

void Level::Level2()

{

Level1();

if (!windowLabel)

{

windowLabel = new QLabel(this);

windowLabel->setGeometry(945, 150, 325, 300);

QPixmap windowPixmap(":/pictures/window.png");

windowLabel->setPixmap(windowPixmap.scaled(325, 300));

openWindowZone.setRect(1100, 180, 80, 250);

}

if (!fly2)

{

fly2 = new SmartFly(this, windowLabel);

}

}

void Level::Level3()

{

Level2();

fly1->setGeometry(1000, 200, 24, 24);

if (fly2)

{

delete fly2;

fly2 = nullptr;

}

if (windowLabel)

{

delete windowLabel;

windowLabel = nullptr;

openWindowZone.setRect(0, 0, 0, 0);

}

if (trapLabel)

{

delete trapLabel;

trapLabel = nullptr;

}

if (!fly3)

{

fly3 = new Fly(this);

fly3->setGeometry(600, 300, 24, 24);

}

if (!frog)

{

frog = new Frog(this);

frog->setGeometry(100, 650, 36, 36);

}

}

void Level::stopLevel()

{

timer->stop();

if (fly1)

{

delete fly1;

fly1 = nullptr;

}

if (fly2)

{

delete fly2;

fly2 = nullptr;

}

if (fly3)

{

delete fly3;

fly3 = nullptr;

}

if (windowLabel)

{

delete windowLabel;

windowLabel = nullptr;

}

}

void Level::keyPressEvent(QKeyEvent \*event)

{

if (frog)

{

frog->keyPressEvent(event);

}

}

void Level::updateFliesPosition()

{

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

QRect gameRect = parentWidget()->rect();

if (!gameRect.contains(cursorPos))

{

if (fly1)

{

fly1->moveRandomly();

}

if (fly2)

{

fly2->moveFly();

}

if (fly3)

{

fly3->moveRandomly();

}

return;

}

if (fly2 && openWindowZone.width() > 0 && fly2->geometry().intersects(openWindowZone))

{

emit levelFailed();

stopLevel();

return;

}

if (fly1)

{

fly1->moveFly();

}

if (fly2)

{

fly2->moveFly();

}

if (fly3)

{

fly3->moveFly();

}

if (trapLabel != nullptr)

{

if (checkIfFlyCaught())

{

emit levelCompleted();

stopLevel();

}

}

if (frog && frog->getCaughtFlies() == 2)

{

emit levelCompleted();

stopLevel();

}

}

#ifndef SMARTFLY\_H

#define SMARTFLY\_H

#include "fly.h"

class SmartFly : public Fly

{

public:

SmartFly(QWidget\* parent, QLabel\* target);

void moveRandomly() override;

void moveFly();

private:

QLabel\* windowLabel;

};

#endif // SMARTFLY\_H

#include "smartfly.h"

SmartFly::SmartFly(QWidget\* parent, QLabel\* target) : Fly(parent), windowLabel(target) {}

void SmartFly::moveRandomly() {

int X = QRandomGenerator::global()->bounded(-5, 5);

int Y = QRandomGenerator::global()->bounded(-5, 5);

int newX = qBound(0, x() + X, parentWidget()->width() - width());

int newY = qBound(0, y() + Y, parentWidget()->height() - height());

move(newX, newY);

}

void SmartFly::moveFly()

{

QPoint windowPos = parentWidget()->mapFromGlobal(windowLabel->mapToGlobal(windowLabel->rect().center()));

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

int windowX = windowPos.x();

int windowY = windowPos.y();

int mouseX = cursorPos.x();

int mouseY = cursorPos.y();

int flyX = this->x();

int flyY = this->y();

int distanceWindowX = windowX - flyX;

int distanceWindowY = windowY - flyY;

int distanceCursorX = mouseX - flyX;

int distanceCursorY = mouseY - flyY;

int distanceSquared = distanceCursorX \* distanceCursorX + distanceCursorY \* distanceCursorY;

if (distanceSquared < 2500)

{

flyX += (distanceCursorX < 0) ? 5 : -5;

flyY += (distanceCursorY < 0) ? 5 : -5;

}

else

{

flyX += (distanceWindowX > 0) ? 5 : -5;

flyY += (distanceWindowY > 0) ? 5 : -5;

}

flyX = qBound(0, flyX, parentWidget()->width() - width());

flyY = qBound(0, flyY, parentWidget()->height() - height());

this->move(flyX, flyY);

this->moveRandomly();

}

#include <QApplication>

#include "game.h"

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

QApplication app(argc, argv);

Game game;

game.setWindowTitle("Гра 'Спіймай муху'");

game.show();

return app.exec();

}

**Лістинг завдання 2(task2):**

#ifndef INVENTORY\_H

#define INVENTORY\_H

#include <vector>

#include <string>

#include <algorithm>

#include <iostream>

#include <iomanip>

using namespace std;

template <typename T>

class Inventory

{

private:

vector<T> items;

public:

void addItem(const T& item);

void removeItemByName(const string& name);

void displayAllItems() const;

void findItemsByPrice(double price) const;

};

#include "inventory.cpp"

#endif // INVENTORY\_H

#ifndef INVENTORY\_CPP

#define INVENTORY\_CPP

#include "inventory.h"

template <typename T>

void Inventory<T>::addItem(const T& item) {

items.push\_back(item);

}

template <typename T>

void Inventory<T>::removeItemByName(const string& name) {

auto it = remove\_if(items.begin(), items.end(),

[&name](const T& item) { return item.getName() == name; });

if (it != items.end()) {

items.erase(it, items.end());

cout << "Item \"" << name << "\" removed successfully.\n";

} else {

cout << "Item \"" << name << "\" not found.\n";

}

}

template <typename T>

void Inventory<T>::displayAllItems() const {

if (items.empty()) {

cout << "Inventory is empty.\n";

return;

}

cout << left << setw(15) << "Name"

<< setw(10) << "Price"

<< setw(10) << "Quantity" << endl;

cout << "------------------------------------\n";

for (const auto& item : items) {

item.display();

}

}

template <typename T>

void Inventory<T>::findItemsByPrice(double price) const {

bool found = false;

for (const auto& item : items) {

if (item.getPrice() == price) {

if (!found) {

cout << "Items with price " << price << ":\n";

cout << left << setw(15) << "Name"

<< setw(10) << "Price"

<< setw(10) << "Quantity" << std::endl;

cout << "------------------------------------\n";

}

item.display();

found = true;

}

}

if (!found) {

cout << "No items found with price " << price << ".\n";

}

}

#endif // INVENTORY\_CPP

#ifndef PRODUCT\_H

#define PRODUCT\_H

#include <string>

using namespace std;

class Product {

private:

string name;

double price;

int quantity;

public:

Product(const string& name, double price, int quantity);

string getName() const;

double getPrice() const;

int getQuantity() const;

void setQuantity(int newQuantity);

void display() const;

};

#endif // PRODUCT\_H

#include "product.h"

#include <iostream>

#include <iomanip>

Product::Product(const string& name, double price, int quantity)

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

string Product::getName() const { return name; }

double Product::getPrice() const { return price; }

int Product::getQuantity() const { return quantity; }

void Product::setQuantity(int newQuantity) { quantity = newQuantity; }

void Product::display() const {

cout << left << setw(15) << name

<< setw(10) << price

<< setw(10) << quantity << endl;

}

#include "Product.h"

#include "Inventory.h"

using namespace std;

int main()

{

Inventory<Product> inventory;

inventory.addItem(Product("Apple", 0.5, 100));

inventory.addItem(Product("Laptop", 1000.0, 5));

inventory.addItem(Product("T-shirt", 20.0, 50));

inventory.addItem(Product("Banana", 0.5, 150));

cout << "All items in inventory:\n";

inventory.displayAllItems();

cout << "\n";

double searchPrice = 0.5;

cout << "Searching for items with price " << searchPrice << ":\n";

inventory.findItemsByPrice(searchPrice);

cout << "\n";

string nameToRemove = "Apple";

cout << "Removing item \"" << nameToRemove << "\":\n";

inventory.removeItemByName(nameToRemove);

cout << "\n";

std::cout << "Updated inventory:\n";

inventory.displayAllItems();

return 0;

}

**Висновок:**

У ході виконання лабораторної роботи було вивчено основи використання шаблонів у мові програмування C++. Шаблони є потужним інструментом, що дозволяє створювати параметризовані класи та функції, які можуть працювати з різними типами даних, забезпечуючи гнучкість і універсальність програмного забезпечення. Було розглянуто приклади реалізації шаблонів класів і функцій, а також основи спеціалізації шаблонів для оптимізації під конкретні типи даних.