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. КБ-23 |
| Захарченко А.О. |
| Перевірив: викладач  Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

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

**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.

**Реалізація:**

**GameWindow.h**

#ifndef GAMEWINDOW\_H

#define GAMEWINDOW\_H

#include <QMainWindow>

#include <QLabel>

#include <QTimer>

#include <QRandomGenerator>

#include <QMouseEvent>

#include <QElapsedTimer>

class Trap : public QLabel

{

public:

Trap(QWidget \*parent = nullptr);

};

class Fly : public QLabel

{

Q\_OBJECT

public:

Fly(QWidget \*parent = nullptr);

void setRandomPosition();

void moveToExit(int exitX, int exitY);

void repelFromCursor(int cursorX, int cursorY);

void moveFly(int cursorX, int cursorY, int exitX, int exitY);

void moveToCenter();

private:

QPoint exitPosition;

};

class Frog : public QLabel

{

Q\_OBJECT

public:

Frog(QWidget \*parent = nullptr);

void followFly(const QPoint &flyPosition);

};

class GameWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit GameWindow(int flyCount = 1, QWidget \*parent = nullptr);

~GameWindow();

protected:

void mouseMoveEvent(QMouseEvent \*event) override;

private:

void moveFly();

void checkCollision();

void showEndMessage(const QString &message, const QColor &color);

int flyCount;

Fly \*fly1;

Fly \*fly2;

Trap \*trap;

Frog \*frog;

QTimer \*flyTimer;

QLabel \*exitWindow;

QLabel \*timerLabel;

QElapsedTimer \*gameTimer;

QPoint cursorPos;

bool gameOver=false;

};

#endif // GAMEWINDOW\_H

**GameWindow.cpp**

#include "GameWindow.h"

#include <QVBoxLayout>

#include <QTimer>

#include <cmath>

#include <QMessageBox>

Trap::Trap(QWidget \*parent) : QLabel(parent)

{

setPixmap(QPixmap("D:\\task4\\gepard361o.jpg"));

setScaledContents(true);

setFixedSize(150, 50);

move(0, 0);

show();

}

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

{

setScaledContents(true);

setFixedSize(50, 50);

exitPosition = QPoint(400, 300);

setRandomPosition();

show();

}

void Fly::setRandomPosition()

{

int x = QRandomGenerator::global()->bounded(0, parentWidget()->width() - width());

int y = QRandomGenerator::global()->bounded(0, parentWidget()->height() - height());

move(x, y);

}

void Fly::moveToExit(int exitX, int exitY)

{

int deltaX = exitX - x();

int deltaY = exitY - y();

double distance = std::sqrt(deltaX \* deltaX + deltaY \* deltaY);

if (distance > 55) {

move(x() + static\_cast<int>(deltaX / distance \* 5), y() + static\_cast<int>(deltaY / distance \* 5));

}

}

void Fly::repelFromCursor(int cursorX, int cursorY)

{

int deltaX = x() - cursorX;

int deltaY = y() - cursorY;

double distance = std::sqrt(deltaX \* deltaX + deltaY \* deltaY);

if (distance > 55) {

move(x() + static\_cast<int>(deltaX / distance \* 10), y() + static\_cast<int>(deltaY / distance \* 10));

}

}

void Fly::moveFly(int cursorX, int cursorY, int exitX, int exitY)

{

repelFromCursor(cursorX, cursorY);

moveToCenter();

moveToExit(exitX, exitY);

}

void Fly::moveToCenter()

{

int centerX = parentWidget()->width() / 2;

int centerY = parentWidget()->height() / 2;

int deltaX = centerX - x();

int deltaY = centerY - y();

double distance = std::sqrt(deltaX \* deltaX + deltaY \* deltaY);

if (distance > 55) {

move(x() + static\_cast<int>(deltaX / distance \* 3), y() + static\_cast<int>(deltaY / distance \* 3));

}

}

Frog::Frog(QWidget \*parent) : QLabel(parent)

{

setPixmap(QPixmap("D:\\task4\\th-1361191126.jpg"));

setScaledContents(true);

setFixedSize(100, 100);

move(parent->width() / 2, parent->height() / 2);

show();

}

void Frog::followFly(const QPoint &flyPosition)

{

int deltaX = flyPosition.x() - x();

int deltaY = flyPosition.y() - y();

double distance = std::sqrt(deltaX \* deltaX + deltaY \* deltaY);

if (distance > 10) {

move(x() + static\_cast<int>(deltaX / distance \* 5), y() + static\_cast<int>(deltaY / distance \* 5));

}

}

GameWindow::GameWindow(int flyCount, QWidget \*parent)

: QMainWindow(parent), flyCount(flyCount), fly1(new Fly(this)), trap(new Trap(this)), frog(nullptr), flyTimer(new QTimer(this))

{

setWindowTitle("Піймай муху");

setFixedSize(800, 600);

gameTimer = new QElapsedTimer();

gameTimer->start();

timerLabel = new QLabel(this);

timerLabel->setStyleSheet("font-size: 16px; color: white; background-color: black; padding: 5px;");

timerLabel->setFixedSize(150, 30);

timerLabel->move(10, 10);

timerLabel->setText("Час: 0 с");

timerLabel->show();

QTimer \*updateTimer = new QTimer(this);

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

int elapsed = static\_cast<int>(gameTimer->elapsed() / 1000);

timerLabel->setText(QString("Час: %1 с").arg(elapsed));

});

updateTimer->start(1000);

exitWindow = new QLabel(this);

exitWindow->setFixedSize(100, 100);

exitWindow->setPixmap(QPixmap("D:\\task4\\window.jpg"));

exitWindow->setScaledContents(true);

exitWindow->move(350, 250);

exitWindow->show();

fly1->setPixmap(QPixmap("D:\\task4\\fly.png"));

if (flyCount > 1) {

fly2 = new Fly(this);

fly2->setPixmap(QPixmap("D:\\task4\\fly.png"));

} else {

fly2 = nullptr;

}

if (flyCount == 3) {

delete exitWindow;

exitWindow = nullptr;

frog = new Frog(this);

}

connect(flyTimer, &QTimer::timeout, this, &GameWindow::moveFly);

flyTimer->start(1000);

}

GameWindow::~GameWindow()

{

delete fly1;

if (fly2) delete fly2;

delete trap;

if (frog) delete frog;

delete timerLabel;

delete gameTimer;

}

void GameWindow::mouseMoveEvent(QMouseEvent \*event)

{

QMainWindow::mouseMoveEvent(event);

cursorPos = event->pos();

}

void GameWindow::moveFly()

{

QPoint exitCenter = exitWindow ? QPoint(exitWindow->x() + exitWindow->width() / 2, exitWindow->y() + exitWindow->height() / 2) : QPoint(0, 0);

fly1->moveFly(cursorPos.x(), cursorPos.y(), exitCenter.x(), exitCenter.y());

if (fly2) {

fly2->moveFly(cursorPos.x(), cursorPos.y(), exitCenter.x(), exitCenter.y());

}

if (frog) {

frog->followFly(fly1->pos());

if (fly2) frog->followFly(fly2->pos());

}

checkCollision();

}

void GameWindow::checkCollision()

{

if (gameOver) return;

QRect fly1Rect = fly1->geometry();

QRect trapRect = trap->geometry();

if (fly1Rect.intersects(trapRect) || (fly2 && fly2->geometry().intersects(trapRect))) {

if (flyCount == 1) {

showEndMessage("Ви зловили муху! Починаємо нову гру з двома мухами!", Qt::green);

GameWindow \*newGameWindow = new GameWindow(2);

newGameWindow->show();

close();

gameOver = true;

} else if (flyCount == 2) {

showEndMessage("Ви зловили обидві мухи! Починаємо третій рівень!", Qt::green);

GameWindow \*newGameWindow = new GameWindow(3);

newGameWindow->show();

close();

gameOver = true;

} else {

showEndMessage("Вітаємо! Ви завершили гру!", Qt::green);

gameOver = true;

}

}

if (frog && (fly1->geometry().intersects(frog->geometry()) || (fly2 && fly2->geometry().intersects(frog->geometry())))) {

showEndMessage("Жаба з'їла муху! Ви програли!", Qt::red);

GameWindow \*newGameWindow = new GameWindow(1);

newGameWindow->show();

close();

gameOver = true;

return;

}

QRect exitWindowRect = exitWindow->geometry();

if (fly1Rect.intersects(exitWindowRect)) {

showEndMessage("Муха доторкнулась до вікна! Ви програли!", Qt::red);

GameWindow \*newGameWindow = new GameWindow(1);

newGameWindow->show();

close();

gameOver = true;

}

if (fly2 && fly2->geometry().intersects(exitWindowRect)) {

showEndMessage("Муха доторкнулась до вікна! Ви програли!", Qt::red);

GameWindow \*newGameWindow = new GameWindow(1);

newGameWindow->show();

close();

gameOver = true;

}

}

void GameWindow::showEndMessage(const QString &message, const QColor &color)

{

int elapsed = static\_cast<int>(gameTimer->elapsed() / 1000);

QString finalMessage = QString("%1\nЧас: %2 секунд").arg(message).arg(elapsed);

fly1->setText(finalMessage);

fly1->setStyleSheet(QString("color: white; background-color: %1;").arg(color.name()));

fly1->setAlignment(Qt::AlignCenter);

fly1->setFixedSize(400, 100);

fly1->raise();

fly1->move((width() - fly1->width()) / 2, (height() - fly1->height()) / 2);

QMessageBox::information(this, "Гра завершена", message + "\nЧас: " + QString::number(elapsed) + " с");

}

**Main.cpp**

#include <QApplication>

#include "GameWindow.h"

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

{

QApplication a(argc, argv);

GameWindow w;

w.show();

return a.exec();

}

**Результат:**

**Завдання 2**

**Варіант 3**

Створи програму для керування банківським рахунком клієнтів. Кожен клієнт має ім'я, прізвище та баланс на рахунку. Необхідно створити клас Customer, який містить ці дані.

Крім того, створи шаблонний клас BankAccount, який представляє банківський рахунок. Цей клас має методи для додавання коштів на рахунок, зняття коштів з рахунку, переказу коштів між рахунками та виведення балансу рахунку.

Забезпеч універсальність класу BankAccount, використовуючи шаблонний параметр для типу клієнта. Це дозволить тобі працювати з різними типами клієнтів (наприклад, фізичні особи, юридичні особи тощо).

Для демонстрації роботи програми:

1. Створи об'єкт BankAccount для керування рахунком клієнта.
2. Додай кошти на рахунок за допомогою методу deposit().
3. Зніми кошти з рахунку за допомогою методу withdraw().
4. Перекажи кошти з одного рахунку на інший за допомогою методу transfer().
5. Виведи на екран баланс рахунку за допомогою методу getBalance().

Не забудь використовувати шаблонний клас BankAccount для керування рахунками різних типів клієнтів. Також, забезпеч, щоб клас Customer мав методи доступу до своїх полів (ім'я, прізвище, баланс).

**Реалізація:**

**Bank.cpp**

#include "bank.h"

Customer::Customer(const std::string& fName, const std::string& lName, double bal)

: firstName(fName), lastName(lName), balance(bal) {}

std::string Customer::getFirstName() const {

return firstName;

}

std::string Customer::getLastName() const {

return lastName;

}

double Customer::getBalance() const {

return balance;

}

void Customer::setBalance(double newBalance) {

balance = newBalance;

}

template <typename T>

BankAccount<T>::BankAccount(const T& customerData)

: customer(customerData) {}

template <typename T>

void BankAccount<T>::deposit(double amount) {

if (amount > 0) {

double currentBalance = customer.getBalance();

customer.setBalance(currentBalance + amount);

}

}

template <typename T>

bool BankAccount<T>::withdraw(double amount) {

double currentBalance = customer.getBalance();

if (amount > 0 && amount <= currentBalance) {

customer.setBalance(currentBalance - amount);

return true;

}

return false;

}

template <typename T>

bool BankAccount<T>::transfer(BankAccount<T>& otherAccount, double amount) {

if (this->withdraw(amount)) {

otherAccount.deposit(amount);

return true;

}

return false;

}

template <typename T>

void BankAccount<T>::printBalance() const {

std::cout << customer.getFirstName() << " " << customer.getLastName()

<< "'s Balance: $" << customer.getBalance() << std::endl;

}

template class BankAccount<Customer>;

**Bank.h**

#ifndef BANK\_H

#define BANK\_H

#include <string>

#include <iostream>

class Customer {

private:

std::string firstName;

std::string lastName;

double balance;

public:

Customer(const std::string& fName, const std::string& lName, double bal);

std::string getFirstName() const;

std::string getLastName() const;

double getBalance() const;

void setBalance(double newBalance);

};

template <typename T>

class BankAccount {

private:

T customer;

public:

BankAccount(const T& customerData);

void deposit(double amount);

bool withdraw(double amount);

bool transfer(BankAccount<T>& otherAccount, double amount);

void printBalance() const;

};

#endif // BANK\_H

**Task6.cpp**

#include <iostream>

#include <limits>

#include "bank.h"

void showMenu() {

std::cout << "\n---- Bank Account Menu ----\n";

std::cout << "1. Deposit money\n";

std::cout << "2. Withdraw money\n";

std::cout << "3. Transfer money\n";

std::cout << "4. Show balance\n";

std::cout << "5. Exit\n";

std::cout << "Enter your choice: ";

}

int main() {

std::string firstName, lastName;

double initialBalance;

std::cout << "Enter first name: ";

std::getline(std::cin, firstName);

std::cout << "Enter last name: ";

std::getline(std::cin, lastName);

std::cout << "Enter initial balance: ";

std::cin >> initialBalance;

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

Customer customer(firstName, lastName, initialBalance);

BankAccount<Customer> account(customer);

int choice;

double amount;

std::string recipientFirstName, recipientLastName;

double transferAmount;

do {

showMenu();

std::cin >> choice;

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

switch (choice) {

case 1: {

std::cout << "Enter deposit amount: $";

std::cin >> amount;

account.deposit(amount);

std::cout << "Deposit successful.\n";

break;

}

case 2: {

std::cout << "Enter withdrawal amount: $";

std::cin >> amount;

if (account.withdraw(amount)) {

std::cout << "Withdrawal successful.\n";

}

else {

std::cout << "Insufficient funds.\n";

}

break;

}

case 3: {

std::cout << "Enter recipient first name: ";

std::getline(std::cin, recipientFirstName);

std::cout << "Enter recipient last name: ";

std::getline(std::cin, recipientLastName);

std::cout << "Enter transfer amount: $";

std::cin >> transferAmount;

Customer recipient(recipientFirstName, recipientLastName, 0);

BankAccount<Customer> recipientAccount(recipient);

if (account.transfer(recipientAccount, transferAmount)) {

std::cout << "Transfer successful.\n";

}

else {

std::cout << "Transfer failed. Check your balance.\n";

}

break;

}

case 4: {

account.printBalance();

break;

}

case 5: {

std::cout << "Exiting...\n";

break;

}

default:

std::cout << "Invalid choice. Please try again.\n";

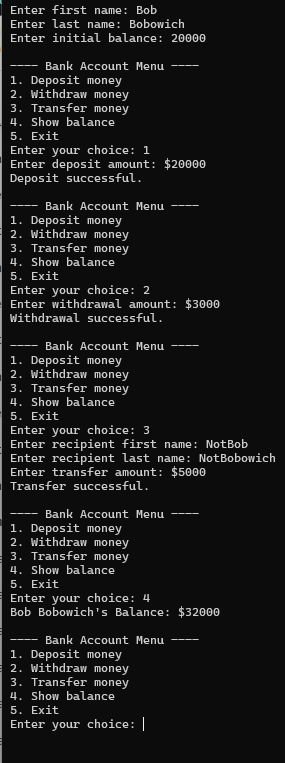
}

} while (choice != 5);

return 0;

}

**Результат:**

****

**Висновок:** в ході роботи було опрацьовано поняття шаблонів в С++ та використав їх на практиці виконавши задане завання. Також хочу зауважити що була дороблена гра: Спімай муху.