Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №6

з дисципліни «Об’єктно-орієнтоване програмування»

**Тема:** «Шаблони в С++»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Гребенюк Д.О. |
| Перевірив: асистент  Козірова Н.Л. |
|  |

Кропивницький

2024

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

**Варіант 4**

**Завдання 1**

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).

**Завдання 1**

*Лістинг bluefly.cpp*

#include "bluefly.h"  
  
BlueFly::BlueFly() : Fly(QPixmap(":/assets/blue\_fly.png")) {}

*Лістинг bluefly.h*

#ifndef BLUEFLY\_H  
#define BLUEFLY\_H  
  
#include <QObject>  
#include "fly.h"  
  
class BlueFly : public Fly  
{  
public:  
 BlueFly();  
};  
  
#endif // BLUEFLY\_H

*Лістинг classicfly.cpp*

#include "classicfly.h"  
  
ClassicFly::ClassicFly() : Fly(QPixmap(":/assets/fly.png")) {}

*Лістинг classicfly.h*

#ifndef CLASSICFLY\_H  
#define CLASSICFLY\_H  
  
#include <QObject>  
#include "fly.h"  
  
class ClassicFly : public Fly  
{  
public:  
 ClassicFly();  
  
};  
  
#endif // CLASSICFLY\_H

*Лістинг fly.cpp*

#include "fly.h"  
#include "mainwindow.h"  
  
#include <QIcon>  
  
Fly::Fly(QPixmap flyPixmap)  
{  
 setFlyPixmap(flyPixmap);  
}  
  
Fly::~Fly()  
{  
  
}  
  
QPixmap Fly::getFlyPixmap() const  
{  
 return flyPixmap;  
}  
  
void Fly::setFlyPixmap(QPixmap newFlyPixmap)  
{  
 flyPixmap = newFlyPixmap;  
}

*Лістинг fly.h*

#ifndef FLY\_H  
#define FLY\_H  
  
#include <QObject>  
#include <QPushButton>  
  
class Fly  
{  
public:  
 Fly(QPixmap flyPixmap);  
 ~Fly();  
  
 QPixmap getFlyPixmap() const;  
 void setFlyPixmap(QPixmap newFlyPixmap);  
  
private:  
 QPixmap flyPixmap;  
  
};  
  
#endif // FLY\_H

*Лістинг greenfly.cpp*

#include "greenfly.h"  
  
GreenFly::GreenFly() : Fly(QPixmap(":/assets/green\_fly.png")) {}

*Лістинг greenfly.h*

#ifndef GREENFLY\_H  
#define GREENFLY\_H  
  
#include <QObject>  
#include "fly.h"  
  
class GreenFly : public Fly  
{  
public:  
 GreenFly();  
  
};  
  
#endif // GREENFLY\_H

*Лістинг main.cpp*

#include "mainwindow.h"  
  
#include <QApplication>  
  
int main(int argc, char \*argv[])  
{  
 QApplication a(argc, argv);  
 MainWindow w;  
 w.show();  
 return a.exec();  
}

*Лістинг mainwindow.cpp*

#include "mainwindow.h"  
#include "qpainter.h"  
#include "ui\_mainwindow.h"  
  
#include <cmath>  
#include <QRandomGenerator>  
#include <QMessageBox>  
  
#include "classicfly.h"  
#include "greenfly.h"  
#include "bluefly.h"  
  
MainWindow::MainWindow(QWidget \*parent)  
 : QMainWindow(parent)  
 , ui(new Ui::MainWindow)  
{  
 ui->setupUi(this);  
  
 flies.push\_back(ClassicFly());  
 flies.push\_back(GreenFly());  
 flies.push\_back(BlueFly());  
  
 flyInitialPos = ui->flyObject->pos();  
 flyLevel = -1;  
 updateFlyLevel();  
  
 ui->frogKillsCount->setVisible(false);  
 frogKills = 0;  
  
 this->setMouseTracking(true);  
 ui->centralwidget->setMouseTracking(true);  
 ui->flyObject->setMouseTracking(true);  
  
 flyUIObjects.push\_back(ui->flyObject);  
}  
  
MainWindow::~MainWindow()  
{  
 delete ui;  
 delete frogTimer;  
}  
  
bool MainWindow::event(QEvent \*event)  
{  
 if (event->type() != QEvent::MouseMove)  
 return QMainWindow::event(event);  
  
 if (flyLevel == 2)  
 return QMainWindow::event(event);  
  
 QPoint mousePos = ui->centralwidget->mapFromGlobal(QCursor::pos());  
  
 for (auto flyUI : flyUIObjects)  
 {  
 QPoint flyPos = flyUI->pos();  
 double distance = calculateDistance(flyPos, mousePos);  
 moveFly(flyUI, distance);  
  
 flyPos = ui->centralwidget->mapFromGlobal(  
 flyUI->mapToGlobal(flyUI->rect().center())  
 );  
  
 if (ui->trapPlantObject->geometry().contains(flyPos)) {  
 QMessageBox msgBox;  
 msgBox.setIcon(QMessageBox::Information);  
 msgBox.setText("Муха спіймана, спробуйте ще раз :)");  
 msgBox.exec();  
  
 for (auto flyUI : flyUIObjects)  
 {  
 flyUI->move(flyInitialPos);  
 }  
 }  
  
 if (ui->windowObject->geometry().contains(flyPos, true)) {  
 QMessageBox msgBox;  
 msgBox.setIcon(QMessageBox::Information);  
 msgBox.setText(QString("Побіг вдався 😁\nВи пройшли: %1/%2.").arg(QString::number(flyLevel + 1), QString::number(MAX\_LEVEL)));  
 msgBox.exec();  
  
 switch (flyLevel) {  
 case 0: {  
 updateFlyLevel();  
 break;  
 }  
 case 1: {  
 updateFlyLevel();  
  
 QPixmap frogPixmap(":/assets/frog.png");  
 ui->frogObject->setIcon(QIcon(frogPixmap));  
 ui->frogObject->setIconSize(QSize(FLY\_OBJECT\_WIDTH, FLY\_OBJECT\_HEIGHT));  
  
 frogTimer = new QTimer(this);  
 frogTimer->setInterval(32);  
 connect(frogTimer, &QTimer::timeout, this, &MainWindow::frogTimerCallback);  
 frogTimer->start();  
  
 ui->frogKillsCount->setVisible(true);  
  
 break;  
 }  
 default:  
 break;  
 }  
  
 for (auto flyUI : flyUIObjects)  
 {  
 flyUI->move(flyInitialPos);  
 }  
 }  
 }  
  
 return QMainWindow::event(event);  
}  
  
void MainWindow::moveFly(QPushButton\* flyUI, double distance)  
{  
 if (distance > FLY\_MOVE\_DIST\_CONDITION)  
 return;  
  
 QPoint flyPos = flyUI->pos();  
 QPoint newFlyPos(0, 0);  
  
 if (flyLevel == 2) {  
 QPoint frogPosCenter = ui->centralwidget->mapFromGlobal(  
 ui->frogObject->mapToGlobal(ui->frogObject->rect().center())  
 );  
  
 if (flyUI->geometry().contains(frogPosCenter)) {  
 frogKills += 1;  
 ui->frogKillsCount->setText(QString("Рахунок: %1").arg(frogKills));  
 }  
 }  
  
 while (true) {  
 int randomX = generateRandomXY() + 10;  
 int randomY = generateRandomXY();  
  
 newFlyPos.setX(flyPos.x() + randomX);  
 newFlyPos.setY(flyPos.y() + randomY);  
  
 int windowMoveLimitForX = this->size().width() - flyUI->iconSize().width();  
 if (newFlyPos.x() < 0 || newFlyPos.x() > windowMoveLimitForX)  
 continue;  
  
 int windowMoveLimitForY = this->size().height() - flyUI->iconSize().height();  
 if (newFlyPos.y() < 0 || newFlyPos.y() > windowMoveLimitForY)  
 continue;  
  
 break;  
 }  
  
 flyUI->move(newFlyPos);  
}  
  
int MainWindow::generateRandomXY() {  
 return QRandomGenerator::global()->bounded(-FLY\_MOVE\_DIST\_CONDITION, FLY\_MOVE\_DIST\_CONDITION);  
}  
  
double MainWindow::calculateDistance(const QPoint &firstPoint, const QPoint &secondPoint) {  
 return sqrt(pow(secondPoint.x() - firstPoint.x(), 2) + pow(secondPoint.y() - firstPoint.y(), 2));  
}  
  
void MainWindow::updateFlyLevel()  
{  
 flyLevel += 1;  
  
 if (flyLevel >= flies.size())  
 return;  
  
 if (flyLevel != 1) {  
 ui->flyObject->setIcon(QIcon(flies[flyLevel].getFlyPixmap()));  
 ui->flyObject->setIconSize(QSize(FLY\_OBJECT\_WIDTH, FLY\_OBJECT\_HEIGHT));  
 }  
  
 if (flyLevel == 0)  
 {  
 flyUIObjects.clear();  
 flyUIObjects.push\_back(ui->flyObject);  
  
 ui->duplicateFly->setVisible(false);  
 }  
 else if (flyLevel == 1)  
 {  
 flyUIObjects.clear();  
 flyUIObjects.push\_back(ui->flyObject);  
 flyUIObjects.push\_back(ui->duplicateFly);  
  
 ui->duplicateFly->setIcon(QIcon(flies[flyLevel].getFlyPixmap()));  
 ui->duplicateFly->setIconSize(QSize(FLY\_OBJECT\_WIDTH, FLY\_OBJECT\_HEIGHT));  
 ui->duplicateFly->move(ui->flyObject->pos());  
  
 ui->duplicateFly->setVisible(true);  
 }  
 else if (flyLevel == 2)  
 {  
 flyUIObjects.clear();  
 flyUIObjects.push\_back(ui->flyObject);  
  
 ui->duplicateFly->setVisible(false);  
 }  
}  
  
void MainWindow::frogTimerCallback()  
{  
 double speed = 2.0;  
  
 QPoint flyPosCenter = ui->centralwidget->mapFromGlobal(  
 ui->flyObject->mapToGlobal(ui->flyObject->rect().center())  
 );  
 QPoint frogPosCenter = ui->centralwidget->mapFromGlobal(  
 ui->frogObject->mapToGlobal(ui->frogObject->rect().center())  
 );  
  
 QPoint flyPos = ui->flyObject->pos();  
 QPoint frogPos = ui->frogObject->pos();  
  
 moveFly(ui->flyObject, calculateDistance(flyPos, frogPosCenter));  
  
 double dx = flyPosCenter.x() - frogPos.x();  
 double dy = flyPosCenter.y() - frogPos.y();  
 double distance = calculateDistance(frogPos, flyPosCenter);  
  
 QPoint newFrogPos(  
 frogPos.x() + ((dx / distance) \* speed),  
 frogPos.y() + ((dy / distance) \* speed)  
 );  
 ui->frogObject->move(newFrogPos.x(), newFrogPos.y());  
}  
  
void MainWindow::paintEvent(QPaintEvent \*event)  
{  
 QPainter painter(this);  
 QPixmap background((flyLevel == 2) ? ":/assets/room\_bg\_no\_window.png" : ":/assets/room\_bg.png");  
  
 painter.drawPixmap(0, 0, width(), height(), background);  
  
 ui->centralwidget->update();  
 QMainWindow::paintEvent(event);  
}

*Лістинг mainwindow.h*

#ifndef MAINWINDOW\_H  
#define MAINWINDOW\_H  
  
#include <QMainWindow>  
#include <QTimer>  
#include "fly.h"  
  
#define FLY\_OBJECT\_WIDTH 48  
#define FLY\_OBJECT\_HEIGHT 48  
  
#define FLY\_MOVE\_DIST\_CONDITION 70  
  
#define MAX\_LEVEL 3  
  
QT\_BEGIN\_NAMESPACE  
namespace Ui {  
class MainWindow;  
}  
QT\_END\_NAMESPACE  
  
class MainWindow : public QMainWindow  
{  
 Q\_OBJECT  
  
public:  
 MainWindow(QWidget \*parent = nullptr);  
 ~MainWindow();  
  
private:  
 Ui::MainWindow \*ui;  
  
 QPoint flyInitialPos;  
  
 int flyLevel;  
  
 std::vector<Fly> flies;  
  
 QTimer \*frogTimer;  
  
 int frogKills;  
  
 std::vector<QPushButton\*> flyUIObjects;  
  
 bool event(QEvent \* event) override;  
  
 void moveFly(QPushButton\* flyUI, double distance);  
  
 int generateRandomXY();  
 double calculateDistance(const QPoint &firstPoint, const QPoint &secondPoint);  
  
 void updateFlyLevel();  
  
private slots:  
 void frogTimerCallback();  
  
protected:  
 void paintEvent(QPaintEvent \*event) override;  
  
};  
  
#endif // MAINWINDOW\_H

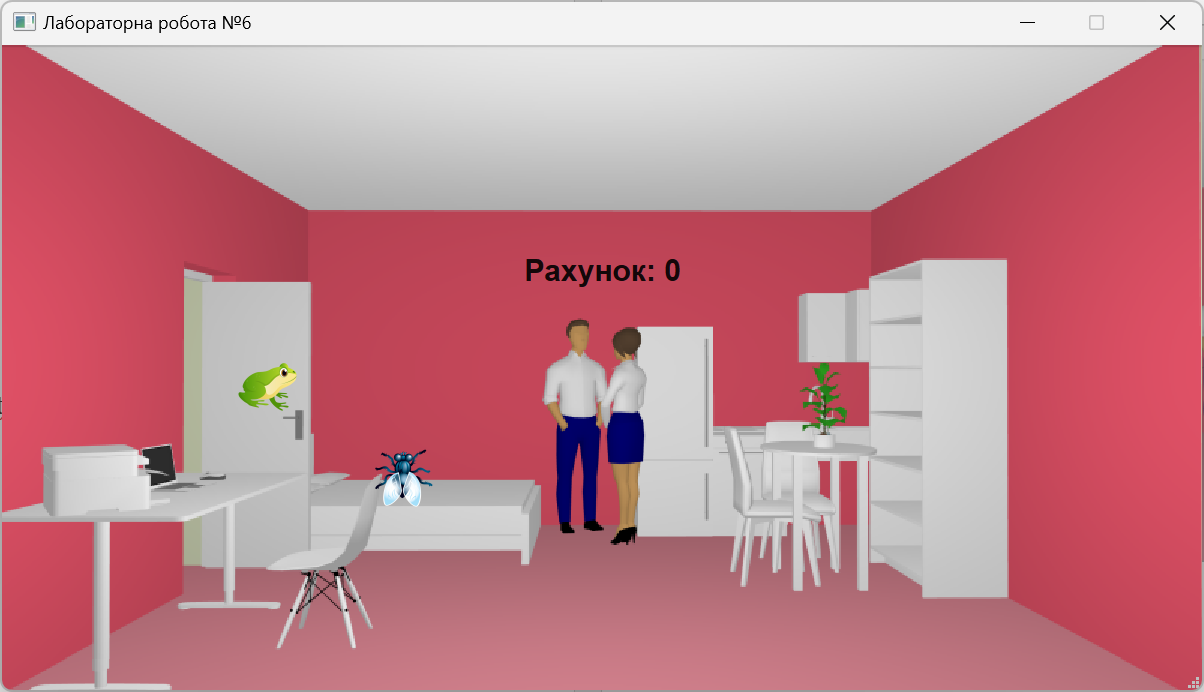
*Лістинг mainwindow.ui*

<?xml version="1.0" encoding="UTF-8"?>  
<ui version="4.0">  
 <class>MainWindow</class>  
 <widget class="QMainWindow" name="MainWindow">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>800</width>  
 <height>430</height>  
 </rect>  
 </property>  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>800</width>  
 <height>430</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>800</width>  
 <height>430</height>  
 </size>  
 </property>  
 <property name="windowTitle">  
 <string>Лабораторна робота №6</string>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <widget class="QPushButton" name="flyObject">  
 <property name="geometry">  
 <rect>  
 <x>110</x>  
 <y>190</y>  
 <width>42</width>  
 <height>48</height>  
 </rect>  
 </property>  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">background-color: none;  
border: none;</string>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="windowObject">  
 <property name="geometry">  
 <rect>  
 <x>620</x>  
 <y>140</y>  
 <width>61</width>  
 <height>161</height>  
 </rect>  
 </property>  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Minimum" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>0</width>  
 <height>0</height>  
 </size>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">background-color: none;  
border: none;</string>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="trapPlantObject">  
 <property name="geometry">  
 <rect>  
 <x>570</x>  
 <y>210</y>  
 <width>51</width>  
 <height>71</height>  
 </rect>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>0</width>  
 <height>0</height>  
 </size>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">background-color: none;  
border: none;</string>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="duplicateFly">  
 <property name="geometry">  
 <rect>  
 <x>10</x>  
 <y>340</y>  
 <width>42</width>  
 <height>48</height>  
 </rect>  
 </property>  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">background-color: none;  
border: none;</string>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="frogObject">  
 <property name="geometry">  
 <rect>  
 <x>110</x>  
 <y>290</y>  
 <width>42</width>  
 <height>48</height>  
 </rect>  
 </property>  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>42</width>  
 <height>48</height>  
 </size>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">background-color: none;  
border: none;</string>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 </widget>  
 <widget class="QLabel" name="frogKillsCount">  
 <property name="enabled">  
 <bool>true</bool>  
 </property>  
 <property name="geometry">  
 <rect>  
 <x>330</x>  
 <y>130</y>  
 <width>141</width>  
 <height>41</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <family>Arial</family>  
 <pointsize>15</pointsize>  
 <bold>true</bold>  
 <stylestrategy>PreferDefault</stylestrategy>  
 </font>  
 </property>  
 <property name="text">  
 <string>Рахунок: 0</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignmentFlag::AlignCenter</set>  
 </property>  
 </widget>  
 </widget>  
 <widget class="QMenuBar" name="menubar">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>800</width>  
 <height>18</height>  
 </rect>  
 </property>  
 </widget>  
 <widget class="QStatusBar" name="statusbar"/>  
 </widget>  
 <resources/>  
 <connections/>  
</ui>

*Результат виконання*







**Завдання 2**

*Лістинг clothingproduct.h*

#ifndef CLOTHINGPRODUCT\_H  
#define CLOTHINGPRODUCT\_H  
  
#include "product.h"  
  
#include <iostream>  
  
class ClothingProduct : public Product {  
 std::string size;  
  
public:  
 ClothingProduct(const std::string& name\_, double price\_, int quantity\_, const std::string& size\_)  
 : Product(name\_, price\_, quantity\_), size(size\_) {}  
  
 void display() const override {  
 std::cout << "Назва: " << name << " (Одяг)"  
 << ", Ціна: " << price << ", Кількість: " << quantity  
 << ", Розмір: " << size << std::endl;  
 }  
};  
  
#endif // CLOTHINGPRODUCT\_H

*Лістинг electronicproduct.h*

#ifndef ELECTRONICPRODUCT\_H  
#define ELECTRONICPRODUCT\_H  
  
#include "product.h"  
  
#include <iostream>  
  
class ElectronicProduct : public Product {  
 int warrantyPeriod;  
  
public:  
 ElectronicProduct(const std::string& name\_, double price\_, int quantity\_, int warrantyPeriod\_)  
 : Product(name\_, price\_, quantity\_), warrantyPeriod(warrantyPeriod\_) {}  
  
 void display() const override {  
 std::cout << "Назва: " << name << " (Електроніка)"  
 << ", Ціна: " << price << ", Кількість: " << quantity  
 << ", Гарантійний період: " << warrantyPeriod << " місяців" << std::endl;  
 }  
};  
  
#endif // ELECTRONICPRODUCT\_H

*Лістинг foodproduct.h*

#ifndef FOODPRODUCT\_H  
#define FOODPRODUCT\_H  
  
#include "product.h"  
  
#include <iostream>  
  
class FoodProduct : public Product {  
 std::string expirationDate;  
  
public:  
 FoodProduct(const std::string& name\_, double price\_, int quantity\_, const std::string& expirationDate\_)  
 : Product(name\_, price\_, quantity\_), expirationDate(expirationDate\_) {}  
  
 void display() const override {  
 std::cout << "Назва: " << name << " (Продукти харчування)"  
 << ", Ціна: " << price << ", Кількість: " << quantity  
 << ", Термін придатності: " << expirationDate << std::endl;  
 }  
};  
  
#endif // FOODPRODUCT\_H

*Лістинг inventory.h*

#ifndef INVENTORY\_H  
#define INVENTORY\_H  
  
#include <iostream>  
#include <string>  
#include <vector>  
  
template <typename T>  
class Inventory {  
 std::vector<T> products;  
  
public:  
 void addProduct(const T& product) {  
 products.push\_back(product);  
 }  
  
 void removeProductByName(const std::string& name) {  
 for (auto it = products.begin(); it != products.end(); ++it) {  
 if ((\*it)->getName() == name) {  
 products.erase(it);  
 std::cout << "Товар \"" << name << "\" було видалено з інвентаря.\n";  
 return;  
 }  
 }  
 std::cout << "Товар з назвою \"" << name << "\" не знайдено.\n";  
 }  
  
 void displayAllProducts() const {  
 for (const auto& product : products) {  
 product->display();  
 }  
 }  
  
 void findProductsByPrice(double price) const {  
 bool found = false;  
 for (const auto& product : products) {  
 if (product->getPrice() == price) {  
 product->display();  
 found = true;  
 }  
 }  
 if (!found) {  
 std::cout << "Товари з ціною " << price << " не знайдено.\n";  
 }  
 }  
};  
  
#endif // INVENTORY\_H

*Лістинг main.cpp*

#include <iostream>  
  
#include "inventory.h"  
#include "product.h"  
#include "foodproduct.h"  
#include "electronicproduct.h"  
#include "clothingproduct.h"  
  
using namespace std;  
  
int main()  
{  
 system("chcp 65001 & cls");  
  
 Inventory<Product\*> inventory;  
  
 inventory.addProduct(new FoodProduct("Яблука", 30.0, 100, "01.12.2024"));  
 inventory.addProduct(new ElectronicProduct("Телефон", 5000.0, 10, 24));  
 inventory.addProduct(new ClothingProduct("Футболка", 250.0, 50, "M"));  
  
 std::cout << "Список всіх товарів на складі:\n";  
 inventory.displayAllProducts();  
  
 double searchPrice = 250.0;  
 std::cout << "\nТовари з ціною " << searchPrice << ":\n";  
 inventory.findProductsByPrice(searchPrice);  
  
 std::string removeName = "Телефон";  
 std::cout << std::endl;  
 inventory.removeProductByName(removeName);  
  
 std::cout << "\nОновлений список товарів на складі:\n";  
 inventory.displayAllProducts();  
  
 return 0;  
}

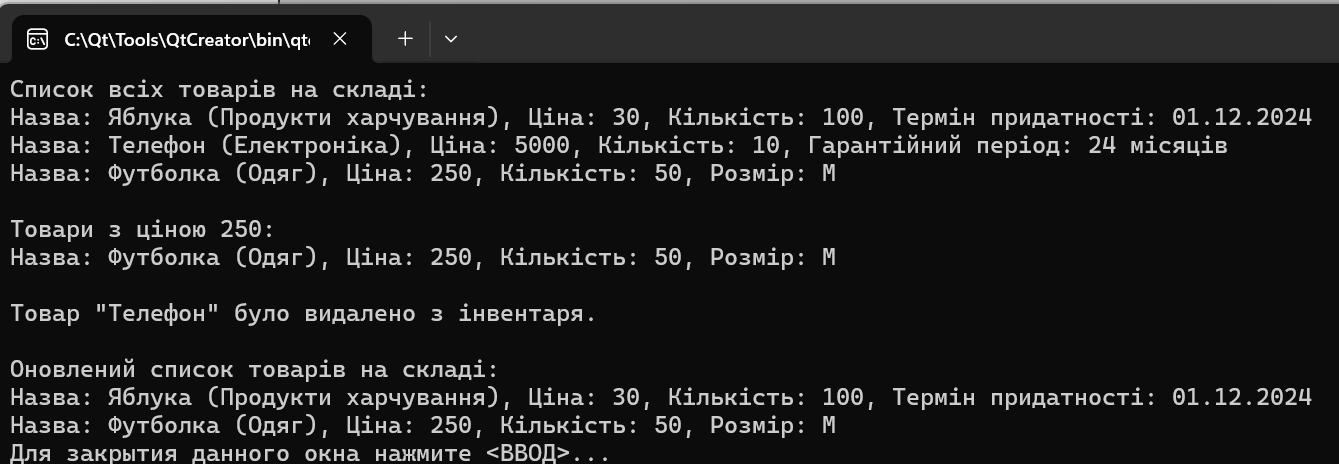
*Лістинг product.cpp*

#include "product.h"  
  
#include <iostream>  
  
std::string Product::getName() const { return name; }  
  
double Product::getPrice() const { return price; }  
  
int Product::getQuantity() const { return quantity; }  
  
void Product::display() const {  
 std::cout << "Назва: " << name << ", Ціна: " << price << ", Кількість: " << quantity << std::endl;  
}

*Лістинг product.h*

#ifndef PRODUCT\_H  
#define PRODUCT\_H  
  
#include <string>  
  
class Product {  
protected:  
 std::string name;  
 double price;  
 int quantity;  
  
public:  
 Product(const std::string& name\_, double price\_, int quantity\_)  
 : name(name\_), price(price\_), quantity(quantity\_) {}  
  
 std::string getName() const;  
 double getPrice() const;  
 int getQuantity() const;  
  
 virtual void display() const;  
};  
  
#endif // PRODUCT\_H

*Результат виконання*



**Висновок:**

Під час виконання лабораторної роботи №6 я ознайомився з принципами використання шаблонів у мові програмування C++. Виконані завдання продемонстрували універсальність та зручність використання шаблонів у створенні гнучких та ефективних рішень, які можуть працювати з різними типами даних. У другому завданні реалізовано шаблонний клас для управління інвентарем, що додало можливість роботи з різними типами товарів завдяки використанню параметрів шаблонів.

Досвід, набутий у цій роботі, поглибив мої знання в області ООП та відкрив нові можливості.