**об’єктно-орієнтоване програмування**

**Лабораторна робота №6**

Шаблони в С++

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

***Теоретична частина***

Шаблони є потужним інструментом у мові програмування C++. Вони дозволяють програмістам створювати загальні, параметризовані класи та функції, які можуть працювати з різними типами даних. Використання шаблонів дозволяє писати більш загальні й універсальні програми, що збільшує перевикористання коду, покращує його зрозумілість та підтримуваність.

Шаблони класів дозволяють визначити загальну структуру класу, незалежну від конкретного типу даних. Вони визначаються за допомогою ключового слова "template" і параметрів шаблону, які можуть бути типами даних, константами або іншими шаблонами. Наприклад, ось простий шаблон класу "Stack", який реалізує структуру стеку:

template <typename T>

class Stack {

private:

std::vector<T> elements; // Зберігає елементи стеку

public:

void push(const T& element) {

elements.push\_back(element);

}

void pop() {

elements.pop\_back();

}

T& top() {

return elements.back();

}

bool empty() const {

return elements.empty();

}

};

У цьому прикладі тип даних "T" є параметром шаблону. Клас "Stack" може бути використаний для створення стеку будь-якого типу даних. Наприклад:

Stack<int> intStack; // Стек цілих чисел

Stack<std::string> stringStack; // Стек рядків

Крім шаблонів класів, в C++ також є шаблони функцій. Вони дозволяють визначати функції, які працюють з різними типами даних. Шаблони функцій використовують таку саму синтаксичну конструкцію з ключовим словом "template" і параметрами шаблону. Наприклад, ось проста шаблонна функція "max", яка повертає більший із двох переданих значень:

template <typename T>

T max(T a, T b) {

return (a > b) ? a : b;

}

Ця функція може бути використана для порівняння значень будь-якого типу даних, для якого визначені оператори порівняння.

int result1 = max(10, 20); // Поверне 20

double result2 = max(3.14, 2.71); // Поверне 3.14

Шаблони в C++ також підтримують спеціалізацію, що дозволяє визначати варіанти шаблону для конкретних типів даних або спеціальних випадків. Зазвичай спеціалізацію використовують для оптимізації або визначення специфічної поведінки для певного типу даних.

Шаблони в C++ - це потужний інструмент, який дозволяє створювати загальні та універсальні програми. Вони забезпечують перевикористання коду, покращують зрозумілість і підтримуваність програм, а також дають можливість програмістам працювати з різними типами даних без необхідності дублювання коду.

***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***

**Варіант 1**

Створи програму для керування бібліотекою книжок. Кожна книжка має заголовок, автора та рік видання. Необхідно створити клас Book, що містить ці дані.

Крім того, створи шаблонний клас Library, який представляє колекцію книжок. Цей клас має методи для додавання книжки, видалення книжки за заголовком, виведення списку всіх книжок та пошуку книжок за автором.

Забезпеч універсальність класу Library, використовуючи шаблонний параметр для типу книжки. Це дозволить тобі працювати з різними типами книжок.

Для демонстрації роботи програми:

1. Створи об'єкт Library для зберігання книжок.
2. Додай до бібліотеки декілька книжок різних типів, наприклад, художніх, наукових, документальних тощо.
3. Виведи на екран список всіх книжок в бібліотеці.
4. Знайди та виведи на екран усі книжки певного автора.
5. Видали з бібліотеки книжку за заголовком.
6. Знову виведи на екран оновлений список книжок в бібліотеці.

Не забудь використовувати шаблонний клас Library для зберігання книжок будь-якого типу. Також, забезпеч, щоб клас Book мав методи доступу до своїх полів (заголовок, автор, рік видання).

**Варіант 2**

Створи програму для керування студентським списком. Кожен студент має ім'я, прізвище та середній бал. Необхідно створити клас Student, який містить ці дані.

Крім того, створи шаблонний клас StudentList, який представляє колекцію студентів. Цей клас має методи для додавання студента, видалення студента за прізвищем, виведення списку всіх студентів та пошуку студентів за середнім балом.

Забезпеч універсальність класу StudentList, використовуючи шаблонний параметр для типу студента. Це дозволить тобі працювати з різними типами студентів (наприклад, інженерний факультет, медичний факультет тощо).

Для демонстрації роботи програми:

1. Створи об'єкт StudentList для зберігання студентів.
2. Додай до списку декілька студентів різних факультетів.
3. Виведи на екран список всіх студентів у списку.
4. Знайди та виведи на екран усіх студентів з певним середнім балом.
5. Видали зі списку студента за прізвищем.
6. Знову виведи на екран оновлений список студентів у списку.

Не забудь використовувати шаблонний клас StudentList для зберігання студентів будь-якого типу. Також, забезпеч, щоб клас Student мав методи доступу до своїх полів (ім'я, прізвище, середній бал).

**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).

**Варіант 4**

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).

**Варіант 5**

Створи програму для керування рестораном. У ресторані є різні страви, які клієнти можуть замовити. Кожна страва має назву, опис та ціну. Необхідно створити клас Dish, який містить ці дані.

Крім того, створи шаблонний клас Menu, який представляє меню ресторану. Цей клас має методи для додавання страви до меню, видалення страви за назвою, виведення списку всіх страв та пошуку страви за ціною.

Забезпеч універсальність класу Menu, використовуючи шаблонний параметр для типу страви. Це дозволить тобі працювати з різними типами страв (наприклад, страви із м'ясом, страви для вегетаріанців тощо).

Для демонстрації роботи програми:

1. Створи об'єкт Menu для зберігання страв.
2. Додай до меню декілька різних страв.
3. Виведи на екран список всіх страв у меню.
4. Знайди та виведи на екран всі страви за певною ціною.
5. Видали з меню страву за назвою.
6. Знову виведи на екран оновлений список страв у меню.

Не забудь використовувати шаблонний клас Menu для зберігання страв будь-якого типу. Також, забезпеч, щоб клас Dish мав методи доступу до своїх полів (назва, опис, ціна).

**Варіант 6**

Створи програму для керування туристичним агентством. Туристичне агентство пропонує різні тури та послуги для подорожей. Кожен тур має назву, опис, ціну та тривалість. Необхідно створити клас Tour, який містить ці дані.

Крім того, створи шаблонний клас TravelAgency, який представляє туристичне агентство. Цей клас має методи для додавання туру до агентства, видалення туру за назвою, виведення списку всіх доступних турів та пошуку туру за ціною.

Забезпеч універсальність класу TravelAgency, використовуючи шаблонний параметр для типу туру. Це дозволить тобі працювати з різними типами турів (наприклад, пляжні тури, екскурсійні тури, гірськолижні тури тощо).

Для демонстрації роботи програми:

1. Створи об'єкт TravelAgency для зберігання турів.
2. Додай до агентства декілька різних турів.
3. Виведи на екран список всіх доступних турів.
4. Знайди та виведи на екран всі тури за певною ціною.
5. Видали з агентства тур за назвою.
6. Знову виведи на екран оновлений список доступних турів.

Не забудь використовувати шаблонний клас TravelAgency для зберігання турів будь-якого типу. Також, забезпеч, щоб клас Tour мав методи доступу до своїх полів (назва, опис, ціна, тривалість).

**Варіант 7**

Створи програму для керування автомобільним сервісним центром. Сервісний центр займається обслуговуванням автомобілів та наданням різних послуг. Кожен автомобіль має марку, модель та рік випуску. Необхідно створити клас Car, який містить ці дані.

Крім того, створи шаблонний клас ServiceCenter, який представляє сервісний центр. Цей клас має методи для додавання автомобіля до списку обслуговування, видалення автомобіля за маркою, виведення списку всіх автомобілів у сервісному центрі та пошуку автомобіля за роком випуску.

Забезпеч універсальність класу ServiceCenter, використовуючи шаблонний параметр для типу автомобіля. Це дозволить тобі працювати з різними типами автомобілів (наприклад, легкові, вантажні, спортивні тощо).

Для демонстрації роботи програми:

1. Створи об'єкт ServiceCenter для зберігання автомобілів.
2. Додай до сервісного центру декілька різних автомобілів.
3. Виведи на екран список всіх автомобілів у сервісному центрі.
4. Знайди та виведи на екран всі автомобілі за певним роком випуску.
5. Видали з сервісного центру автомобіль за маркою.
6. Знову виведи на екран оновлений список автомобілів у сервісному центрі.

Не забудь використовувати шаблонний клас ServiceCenter для зберігання автомобілів будь-якого типу. Також, забезпеч, щоб клас Car мав методи доступу до своїх полів (марка, модель, рік випуску).

**Варіант 8**

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).

**Варіант 9**

Створи програму для керування спортивним клубом. Спортивний клуб має різні секції, кожна з яких спеціалізується на певному виді спорту. Кожна секція має назву, тренера та кількість учасників. Необхідно створити клас Section, який містить ці дані.

Крім того, створи шаблонний клас SportsClub, який представляє спортивний клуб. Цей клас має методи для додавання секції до клубу, видалення секції за назвою, виведення списку всіх доступних секцій та пошуку секції за кількістю учасників.

Забезпеч універсальність класу SportsClub, використовуючи шаблонний параметр для типу секції. Це дозволить тобі працювати з різними видами спорту (наприклад, футбол, баскетбол, плавання тощо).

Для демонстрації роботи програми:

1. Створи об'єкт SportsClub для зберігання секцій.
2. Додай до клубу декілька різних секцій.
3. Виведи на екран список всіх доступних секцій у клубі.
4. Знайди та виведи на екран всі секції з певною кількістю учасників.
5. Видали з клубу секцію за назвою.
6. Знову виведи на екран оновлений список доступних секцій у клубі.

Не забудь використовувати шаблонний клас SportsClub для зберігання секцій будь-якого типу. Також, забезпеч, щоб клас Section мав методи доступу до своїх полів (назва, тренер, кількість учасників).

**Варіант 10**

Створи програму для керування онлайн-курсами. Кожен курс має назву, викладача та кількість студентів. Необхідно створити клас Course, який містить ці дані.

Крім того, створи шаблонний клас OnlineLearningPlatform, який представляє платформу для онлайн-навчання. Цей клас має методи для додавання курсу до платформи, видалення курсу за назвою, виведення списку всіх доступних курсів та пошуку курсу за кількістю студентів.

Забезпеч універсальність класу OnlineLearningPlatform, використовуючи шаблонний параметр для типу курсу. Це дозволить тобі працювати з різними типами курсів (наприклад, програмування, мови, маркетинг тощо).

Для демонстрації роботи програми:

1. Створи об'єкт OnlineLearningPlatform для зберігання курсів.
2. Додай до платформи декілька різних курсів.
3. Виведи на екран список всіх доступних курсів на платформі.
4. Знайди та виведи на екран всі курси з певною кількістю студентів.
5. Видали з платформи курс за назвою.
6. Знову виведи на екран оновлений список доступних курсів на платформі.

Не забудь використовувати шаблонний клас OnlineLearningPlatform для зберігання курсів будь-якого типу. Також, забезпеч, щоб клас Course мав методи доступу до своїх полів (назва, викладач, кількість студентів).

**Варіант 11**

Розроби програму для керування готельним бронюванням. Готель має декілька типів номерів, кожен з яких має свою кількість ліжок, ціну та доступність. Необхідно створити клас Room, який містить ці дані.

Створи клас HotelBookingSystem, який буде відповідати за керування бронюванням номерів готелю. Клас має методи для додавання номеру до системи, видалення номеру за його ідентифікатором, виведення списку доступних номерів та пошуку вільних номерів за кількістю ліжок.

Забезпеч універсальність класу HotelBookingSystem, використовуючи шаблонний параметр для типу номеру. Це дозволить тобі працювати з різними типами номерів (наприклад, одномісні, двомісні, люкси тощо).

Для демонстрації роботи програми:

1. Створи об'єкт HotelBookingSystem для керування готельними номерами.
2. Додай до системи декілька різних типів номерів.
3. Виведи на екран список доступних номерів у готелі.
4. Знайди та виведи на екран всі вільні номери з певною кількістю ліжок.
5. Видали з системи номер за його ідентифікатором.
6. Знову виведи на екран оновлений список доступних номерів у готелі.

Не забудь використовувати шаблонний клас HotelBookingSystem для зберігання номерів будь-якого типу. Також, забезпеч, щоб клас Room мав методи доступу до своїх полів (кількість ліжок, ціна, доступність).

**Варіант 12**

Розроби програму для керування замовленнями в ресторані. Ресторан має різні страви у своєму меню, кожна з яких має назву, опис та ціну. Необхідно створити клас Dish, який містить ці дані.

Створи клас OrderManagementSystem, який буде відповідати за керування замовленнями у ресторані. Клас має методи для додавання страви до замовлення, видалення страви за її назвою, виведення списку всіх страв у замовленні та обчислення загальної суми замовлення.

Забезпеч універсальність класу OrderManagementSystem, використовуючи шаблонний параметр для типу страви. Це дозволить тобі працювати з різними типами страв (наприклад, перші страви, основні страви, десерти тощо).

Для демонстрації роботи програми:

1. Створи об'єкт OrderManagementSystem для керування замовленнями в ресторані.
2. Додай до замовлення декілька різних страв.
3. Виведи на екран список всіх страв у замовленні.
4. Видали з замовлення страву за її назвою.
5. Знову виведи на екран оновлений список страв у замовленні.
6. Обчисли загальну суму замовлення та виведи її на екран.

Не забудь використовувати шаблонний клас OrderManagementSystem для зберігання страв будь-якого типу. Також, забезпеч, щоб клас Dish мав методи доступу до своїх полів (назва, опис, ціна).

**Варіант 13**

Розроби програму для керування запозиченням та поверненням велосипедів у велопрокаті. Кожен велосипед має унікальний ідентифікатор, модель та стан (доступний або запозичений).

Створи клас Bike, який містить поля для зберігання цих даних. Також, додай методи для отримання та зміни стану велосипеда.

Створи клас BikeRentalSystem, який буде відповідати за керування запозиченням та поверненням велосипедів. Клас повинен мати методи для реєстрації нового велосипеда, запозичення велосипеда за його ідентифікатором, повернення велосипеда за його ідентифікатором та виведення списку доступних велосипедів.

Забезпеч універсальність класу BikeRentalSystem, використовуючи шаблонний параметр для типу велосипеда. Це дозволить тобі працювати з різними типами велосипедів (наприклад, гірські велосипеди, шосейні велосипеди тощо).

Для демонстрації роботи програми:

1. Створи об'єкт BikeRentalSystem для керування запозиченням та поверненням велосипедів.
2. Зареєструй декілька нових велосипедів у системі.
3. Запозичи один з доступних велосипедів за його ідентифікатором.
4. Поверни запозичений велосипед за його ідентифікатором.
5. Виведи на екран список доступних велосипедів.

Не забудь використовувати шаблонний клас BikeRentalSystem для зберігання велосипедів будь-якого типу. Також, забезпеч, щоб клас Bike мав методи доступу до своїх полів (ідентифікатор, модель, стан).

**Варіант 14**

Створіть програму для керування готельним бронюванням. Кожне бронювання має включати ім'я гостя, номер кімнати, дату заїзду та виїзду. Необхідно створити клас Reservation, який містить ці дані.

Крім того, створіть шаблонний клас Hotel, який представляє колекцію бронювань. Цей клас має методи для додавання бронювання, видалення бронювання за номером кімнати, виведення списку всіх бронювань та пошуку бронювань за ім'ям гостя.

Забезпечте універсальність класу Hotel, використовуючи шаблонний параметр для типу бронювання. Це дозволить вам працювати з різними типами бронювань (наприклад, одномісні, двомісні, люксові тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт Hotel для зберігання бронювань.
2. Додайте до готелю декілька бронювань різних типів, наприклад, одномісних, двомісних, люксових тощо.
3. Виведіть на екран список всіх бронювань у готелі.
4. Знайдіть та виведіть на екран усі бронювання певного гостя.
5. Видаліть з готелю бронювання за номером кімнати.
6. Знову виведіть на екран оновлений список бронювань у готелі.

Не забудьте використовувати шаблонний клас Hotel для зберігання бронювань будь-якого типу. Також, забезпечте, щоб клас Reservation мав методи доступу до своїх полів (ім'я гостя, номер кімнати, дата заїзду та виїзду).

**Варіант 15**

Створіть програму для керування списком контактів. Кожен контакт має включати ім'я, прізвище та номер телефону. Необхідно створити клас Contact, який містить ці дані.

Крім того, створіть шаблонний клас ContactList, який представляє колекцію контактів. Цей клас має методи для додавання контакту, видалення контакту за номером телефону, виведення списку всіх контактів та пошуку контактів за ім'ям або прізвищем.

Забезпечте універсальність класу ContactList, використовуючи шаблонний параметр для типу контакту. Це дозволить вам працювати з різними типами контактів (наприклад, особисті, службові, родинні тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт ContactList для зберігання контактів.
2. Додайте до списку кілька контактів різних типів, наприклад, особистих, службових, родинних тощо.
3. Виведіть на екран список всіх контактів у списку.
4. Знайдіть та виведіть на екран усі контакти за певним ім'ям або прізвищем.
5. Видаліть зі списку контакт за номером телефону.
6. Знову виведіть на екран оновлений список контактів у списку.

Не забудьте використовувати шаблонний клас ContactList для зберігання контактів будь-якого типу. Також, забезпечте, щоб клас Contact мав методи доступу до своїх полів (ім'я, прізвище, номер телефону).

**Варіант 16**

Створіть програму для керування домашнім інвентарем. Кожен предмет має містити назву, опис та кількість. Необхідно створити клас Item, який містить ці дані.

Крім того, створіть шаблонний клас Inventory, який представляє колекцію предметів. Цей клас має методи для додавання предмету, видалення предмету за назвою, виведення списку всіх предметів та пошуку предметів за описом.

Забезпечте універсальність класу Inventory, використовуючи шаблонний параметр для типу предмету. Це дозволить вам працювати з різними типами предметів (наприклад, побутові, електроніка, спортивні речі тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт Inventory для зберігання предметів.
2. Додайте до інвентаря кілька предметів різних типів, наприклад, побутових, електроніки, спортивних речей тощо.
3. Виведіть на екран список всіх предметів у інвентарі.
4. Знайдіть та виведіть на екран усі предмети з певним описом.
5. Видаліть з інвентаря предмет за назвою.
6. Знову виведіть на екран оновлений список предметів у інвентарі.

Не забудьте використовувати шаблонний клас Inventory для зберігання предметів будь-якого типу. Також, забезпечте, щоб клас Item мав методи доступу до своїх полів (назва, опис, кількість).

**Варіант 17**

Створіть програму для керування списком подій. Кожна подія має містити назву, дату та час проведення. Необхідно створити клас Event, який містить ці дані.

Крім того, створіть шаблонний клас EventManager, який дозволить додавати події, видаляти події за назвою, виводити список всіх подій та шукати події за датою.

Забезпечте універсальність класу EventManager, використовуючи шаблонний параметр для типу події. Це дозволить вам працювати з різними типами подій (наприклад, зустрічі, конференції, виставки тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт EventManager для керування списком подій.
2. Додайте до списку кілька подій різних типів, наприклад, зустрічі, конференції, виставки тощо.
3. Виведіть на екран список всіх подій.
4. Знайдіть та виведіть на екран усі події, що відбудуться в певну дату.
5. Видаліть зі списку подію за назвою.

Знову виведіть на екран оновлений список подій.

Не забудьте використовувати шаблонний клас EventManager для керування подіями будь-якого типу. Також, забезпечте, щоб клас Event мав методи доступу до своїх полів (назва, дата, час проведення).

**Варіант 18**

Створіть програму для керування списком задач. Кожна задача має містити опис, пріоритет та статус (виконано/не виконано). Необхідно створити клас Task, який містить ці дані.

Крім того, створіть шаблонний клас TaskManager, який дозволить додавати задачі, видаляти задачі за описом, виводити список всіх задач та змінювати статус задачі.

Забезпечте універсальність класу TaskManager, використовуючи шаблонний параметр для типу задачі. Це дозволить вам працювати з різними типами задач (наприклад, робочі завдання, домашні обов'язки, особисті цілі тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт TaskManager для керування списком задач.
2. Додайте до списку кілька задач різних типів, наприклад, робочі завдання, домашні обов'язки, особисті цілі тощо.
3. Виведіть на екран список всіх задач.
4. Змініть статус певної задачі на "виконано".
5. Видаліть зі списку задачу за описом.
6. Знову виведіть на екран оновлений список задач.

Не забудьте використовувати шаблонний клас TaskManager для керування задачами будь-якого типу. Також, забезпечте, щоб клас Task мав методи доступу до своїх полів (опис, пріоритет, статус).

**Варіант 19**

Створіть програму для керування списком покупок. Кожен товар має назву, кількість та ціну. Необхідно створити клас Product, який містить ці дані.

Крім того, створіть шаблонний клас ShoppingList, який дозволить додавати товари до списку покупок, видаляти товари за назвою, виводити список всіх товарів та обчислювати загальну суму покупки.

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

Для демонстрації роботи програми:

1. Створіть об'єкт ShoppingList для керування списком покупок.
2. Додайте до списку декілька товарів різних типів, наприклад, продукти харчування, одяг, електроніка тощо.
3. Виведіть на екран список всіх товарів.
4. Видаліть зі списку товар за назвою.
5. Знову виведіть на екран оновлений список товарів.
6. Обчисліть загальну суму покупки і виведіть її на екран.

Не забудьте використовувати шаблонний клас ShoppingList для керування покупками будь-якого типу товару. Також, забезпечте, щоб клас Product мав методи доступу до своїх полів (назва, кількість, ціна).

**Варіант 20**

Створіть програму для керування списком друзів і їх контактною інформацією. Кожен друг має ім'я, прізвище та номер телефону. Необхідно створити клас Friend, який містить ці дані.

Крім того, створіть клас FriendList, який дозволяє додавати друзів до списку, видаляти друзів за ім'ям або прізвищем, виводити список всіх друзів та проводити пошук друзів за номером телефону.

Забезпечте універсальність класу FriendList, використовуючи шаблонний параметр для типу друзів. Це дозволить вам працювати з різними типами друзів (наприклад, колеги, сімейні члени, однокласники тощо).

Для демонстрації роботи програми:

1. Створіть об'єкт FriendList для керування списком друзів.
2. Додайте до списку декілька друзів з різними іменами та прізвищами.
3. Виведіть на екран список всіх друзів.
4. Видаліть зі списку друга за ім'ям або прізвищем.
5. Знову виведіть на екран оновлений список друзів.
6. Здійсніть пошук друга за номером телефону і виведіть результати на екран.

Не забудьте використовувати шаблонний клас FriendList для керування списком друзів будь-якого типу. Також, забезпечте, щоб клас Friend мав методи доступу до своїх полів (ім'я, прізвище, номер телефону).