

Реферат до проєкту Shop

Виконала:
студентка 2 курсу
спеціальності
Комп'ютерна математика 1
Ковальчук Анна

2024

1. Вступ

Сучасний світ диктує необхідність автоматизації різноманітних процесів, включаючи комерційну діяльність. Розробка програмного забезпечення для моделювання роботи магазинів є важливим напрямком, який дозволяє оптимізувати продажі, керувати запасами товарів та аналізувати взаємодію між клієнтами та працівниками.

Метою даного проєкту є створення програмного додатка для імітації роботи магазину. Це включає моделювання основних бізнес-процесів: облік товарів, взаємодія з клієнтами та продавцями, реалізація системи знижок та бонусів. Для розробки програми було використано дві мови програмування: C та C++. Це дало змогу порівняти підходи до реалізації алгоритмів у процедурному та об'єктно-орієнтованому стилях.

Актуальність цього проєкту зумовлена потребою у створенні універсальних рішень для малого та середнього бізнесу, які дозволяють автоматизувати торгівельну діяльність. Реалізована програма є гнучким інструментом, що може бути адаптований під конкретні потреби користувачів.

Результати цієї роботи можуть бути використані для навчальних цілей, а також для створення базової версії комерційного програмного забезпечення для управління магазинами.

2. Огляд функціоналу програми

Програмний додаток **Shop** призначений для моделювання роботи магазину, зокрема обслуговування покупців, управління товарами та аналізу продажів. Основні функціональні можливості програми включають:

Робота з базою даних товарів, покупців та продавців

Програма реалізує систему зберігання даних, яка дозволяє ефективно управляти інформацією про товари, покупців і працівників магазину. У базі даних можна додавати, редагувати та видаляти записи, що забезпечує гнучкість у роботі з великою кількістю даних.

Моделювання купівлі та продажу товарів

Програма імітує процеси покупки та продажу товарів. Покупець має змогу обирати товари з доступного асортименту, визначати кількість та здійснювати покупку. У свою чергу, продавці обслуговують клієнтів та реєструють операції продажу. Система підтримує облік змін стану складу після кожної операції.

Застосування уцінки товарів та знижок для постійних клієнтів

Для підвищення лояльності покупців і стимулювання продажів програма передбачає автоматичну уцінку товарів на основі фіксованого відсотка. Крім того, постійні клієнти отримують додаткові знижки на покупки, а також накопичують бонусні бали, які можна використовувати для подальших транзакцій.

Збереження та завантаження даних з бінарних файлів

Програма забезпечує можливість збереження всіх даних магазину у вигляді бінарних файлів, що дозволяє зберігати інформацію про стан складу, працівників та покупців між сесіями роботи програми. Також передбачена функція завантаження даних, що дає змогу відновлювати попередній стан магазину.

Інтерактивний інтерфейс користувача

Програма пропонує простий та зрозумілий текстовий інтерфейс, який дозволяє зручно управляти всіма функціями магазину. Чітка структура меню сприяє легкому доступу до всіх основних операцій.

Усі ці можливості роблять програму **Shop** ефективним інструментом для моделювання торговельних процесів та навчання основам програмування.

3. Структура програми

Програма **Shop** побудована за допомогою об'єктно-орієнтованого підходу. Основою її структури є система класів, що моделюють різні аспекти роботи магазину. Нижче наведено короткий опис основних класів та їх взаємозв'язків.

Класи та їхні взаємозв'язки

1. Клас Person

Цей базовий клас описує загальні характеристики людини. Основні поля:

- Ім'я (**name**).

Основні методи:

- Конструктор для ініціалізації.
- Метод для отримання інформації про людину.

2. Клас Customer

Клас-нащадок від **Person**, що описує покупця. Основні поля:

- Баланс (**balance**).
- Кількість бонусних балів (**bonusPoints**).
- Статус (**status**), який визначає, чи є клієнт постійним.

Основні методи:

- Метод для здійснення покупки.
- Метод для накопичення бонусних балів.

3. Клас Seller

Клас-нащадок від **Person**, що описує продавця. Основні поля:

- Стаж (**experience**).
- Кількість обслугованих клієнтів (**clientsServed**).

Основні методи:

- Метод для обліку обслуговування клієнтів.

4. Клас Manager

Клас-нащадок від **Person**, що відповідає за керування магазином. Основні поля:

- Рівень відповідальності (**responsibilityLevel**).

Основні методи:

- Метод для управління складом і персоналом.

5. Клас Item

Цей клас моделює товар. Основні поля:

- Назва товару (**name**).
- Ціна (**price**).
- Кількість на складі (**quantity**).

Основні методи:

- Метод для уцінки товару.

6. Клас Shop

Цей клас об'єднує всі елементи магазину. Основні поля:

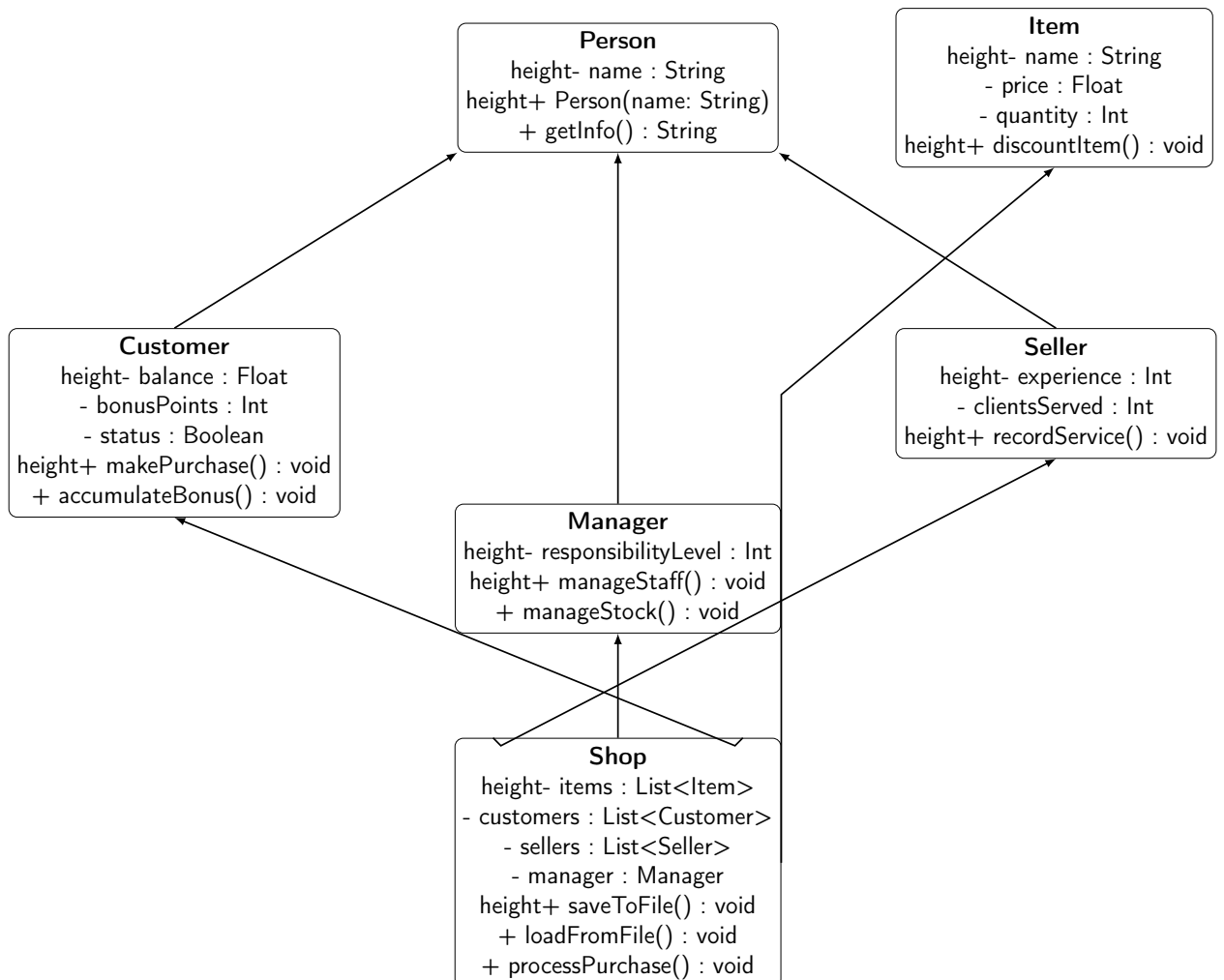
- Список товарів (**items**).
- Список покупців (**customers**).
- Список продавців (**sellers**).
- Керівник магазину (**manager**).

Основні методи:

- Метод для збереження даних у файл.
- Метод для завантаження даних із файлу.
- Метод для здійснення купівлі товарів.

Діаграма класів (UML)

article tikz



4. Функціональність програми

4.1. Основні методи програми

- **Додавання та видалення товарів:** Програма дозволяє додавати нові товари до асортименту магазину, включаючи вказівку їхнього найменування, ціни, кількості та інших характеристик. Також передбачена можливість видалення товарів зі складу, зокрема тих, що більше не доступні для продажу.

```
// C-приклад: додавання товару
void addItem(Item item) {
    items.push_back(item);
}

// C++-приклад: видалення товару за назвою
void removeItem(std::string itemName) {
    for (auto it = items.begin(); it != items.end(); ++it) {
        if (it->getName() == itemName) {
            items.erase(it);
            break;
        }
    }
}
```

- **Застосування уцінки товарів:** Кожен товар у магазині може бути уцінений відповідно до певного відсотка. Уцінка застосовується для стимулювання продажу товарів, які знаходяться на складі протягом тривалого часу або мають дефекти. Після застосування уцінки змінюється ціна товару, що відображається під час процесу покупки.

```
// C-приклад: застосування уцінки товару
void applyDiscount(Item &item, float discount) {
    float newPrice = item.getPrice() * (1 - discount);
    item.setPrice(newPrice);
}

// C++-приклад: застосування знижки до товару
void applyDiscount(Item &item, float discount) {
```

```

        item.setPrice(item.getPrice() * (1 - discount));
    }

```

- **Процес купівлі товарів покупцем:**

- **Урахування статусу клієнта:** Покупець може бути звичайним чи постійним клієнтом. Постійні клієнти отримують знижки на основі їхнього статусу, що відображається на вартості покупки.

```

// C-приклад: перевірка статусу клієнта
void processPurchase(Customer &customer, Item &item, int quantity) {
    float totalPrice = item.getPrice() * quantity;
    if (customer.isRegular()) {
        totalPrice *= 0.9; // 10% знижка для постійних клієнтів
    }
    customer.updateBalance(-totalPrice);
}

```

```

// C++-приклад: застосування знижки для постійних клієнтів
void processPurchase(Customer &customer, Item &item, int quantity) {
    float totalPrice = item.getPrice() * quantity;
    if (customer.isRegular()) {
        totalPrice *= 0.9; // 10% знижка для постійних клієнтів
    }
    customer.updateBalance(totalPrice);
}

```

- **Обчислення підсумкової вартості зі знижкою:** Програма обчислює загальну вартість покупки з урахуванням знижок для постійних клієнтів, а також знижок на товари в залежності від уцінки.

```

// C-приклад: обчислення підсумкової вартості покупки
float calculateTotalPrice(Customer &customer, std::vector<Item> &items) {
    float total = 0;
    for (auto &item : items) {
        total += item.getPrice();
    }
    if (customer.isRegular()) {
        total *= 0.9; // Знижка для постійних клієнтів
    }
}

```



```

    }
    return total;
}

// C++-приклад: обчислення підсумкової вартості з бонусами
float calculateTotalPrice(Customer &customer, std::vector<Item>
    float total = 0;
    for (auto &item : items) {
        total += item.getPrice();
    }
    if (customer.isRegular()) {
        total *= 0.9; // 10% знижка для постійних клієнтів
    }
    return total;
}

```

- **Розрахунок залишків на складі:** Після покупки товарів програма автоматично зменшує кількість товару на складі, відображаючи актуальну наявність товарів у магазині.

```

// C-приклад: оновлення кількості товару після покупки
void updateStock(Item &item, int quantity) {
    item.setQuantity(item.getQuantity() - quantity);
}

// C++-приклад: оновлення кількості товару після покупки
void updateStock(Item &item, int quantity) {
    item.setQuantity(item.getQuantity() - quantity);
}

```

4.2. Робота з файлами

- **Структура збереження та завантаження даних з бінарного файлу:** Дані про товари, покупців, продавців та інші важливі аспекти зберігаються в бінарних файлах. Це дозволяє зберігати великі обсяги даних з мінімальними витратами пам'яті. Програма використовує зручні структури даних для збереження таких об'єктів, як товар, клієнт, продавець та магазин.

```

// C-приклад: збереження товару в бінарний файл

```

```

void saveItemToFile(Item &item, const char *filename) {
    FILE *file = fopen(filename, "wb");
    fwrite(&item, sizeof(Item), 1, file);
    fclose(file);
}

// C++-приклад: завантаження товару з бінарного файлу
void loadItemFromFile(Item &item, const char *filename) {
    std::ifstream inFile(filename, std::ios::binary);
    inFile.read(reinterpret_cast<char*>(&item), sizeof(Item));
    inFile.close();
}

```

- **Забезпечення коректності читання та запису:** Програма забезпечує коректність запису та читання даних з бінарних файлів за допомогою відповідних перевірок. У разі помилки під час зчитування чи запису буде виведено повідомлення про помилку.

```

// C-приклад: перевірка на успішне зчитування
void readItemFromFile(Item &item, const char *filename) {
    FILE *file = fopen(filename, "rb");
    if (file == NULL) {
        printf("Помилка відкриття файлу\n");
        return;
    }
    fread(&item, sizeof(Item), 1, file);
    fclose(file);
}

// C++-приклад: перевірка на успішне зчитування
void readItemFromFile(Item &item, const char *filename) {
    std::ifstream inFile(filename, std::ios::binary);
    if (!inFile) {
        std::cerr << "Помилка відкриття файлу!" << std::endl;
        return;
    }
    inFile.read(reinterpret_cast<char*>(&item), sizeof(Item));
    inFile.close();
}

```

4.3. Особливості моделювання

- **Використання випадкових процесів для генерації даних:** Програма генерує випадкові дані для створення покупців, продавців і товарів з певними характеристиками, що моделює реальні умови для більшої достовірності. Це дозволяє створювати різноманітні сценарії, що можуть бути використані для тестування функціональності програми.

```
// С-приклад: генерація випадкових даних
void generateRandomData() {
    srand(time(0));
    // Генерація випадкового товару
    Item newItem("Товар" + std::to_string(rand()%100), rand()%100 + 1, 1);
    items.push_back(newItem);
}

// C++-приклад: генерація випадкових покупців
void generateRandomCustomer() {
    std::string name = "Клієнт" + std::to_string(rand() % 100);
    Customer newCustomer(name, rand() % 2 == 0); // випадковий статус
    customers.push_back(newCustomer);
}
```

- **Реалізація знижок і бонусів:** Для постійних клієнтів програма реалізує систему бонусів, де клієнт накопичує бонуси після кожної покупки. Знижки можуть застосовуватися до товарів на основі їхнього статусу або залежно від кількості покупок клієнта.

```
// С-приклад: додавання бонусів для постійного клієнта
void addBonusPoints(Customer &customer, int points) {
    customer.addBonusPoints(points);
}

// C++-приклад: знижка для клієнта за бонуси
void applyBonusDiscount(Customer &customer) {
    if (customer.getBonusPoints() > 100) {
        totalPrice *= 0.85; // 15% знижка для клієнтів з більше 100 бонусів
    }
}
```

5. Відмінності між реалізаціями на С та С++

У цьому розділі розглянуті ключові відмінності між реалізаціями програм на С та С++, які стосуються модульності, роботи з класами та об'єктами, обробки файлів та можливостей для розширення функціоналу.

5.1 Модульність у С та об'єктно-орієнтований підхід у С++

- У С програмування базується на процедурному підході. Кроки коду організовуються у функції, і для поділу на модулі використовуються структури даних та файли. Однак, у С відсутня підтримка класів і об'єктів, що ускладнює масштабування програми.
- У С++ використовується об'єктно-орієнтований підхід. Завдяки підтримці класів і об'єктів можна організувати код у вигляді окремих об'єктів з чіткими інтерфейсами. Це дозволяє легше працювати з великими та складними програмами, підвищуючи модульність і зручність.

5.2 Зручність роботи з класами та об'єктами

- У С для роботи з даними використовуються структури, а функції працюють з вказівниками або масивами для доступу до даних. Такий підхід ускладнює організацію коду в більших проєктах.
- У С++ наявність класів дозволяє зручно організувати програму, створюючи об'єкти з методами, які працюють із даними. Це дає більш чисту та логічну структуру для великого коду, підвищуючи зрозумілість і гнучкість.

5.3 Різниця в обробці файлів

- У С для роботи з файлами використовуються функції з бібліотеки `stdio.h`, такі як `fopen()`, `fread()`, `fwrite()`, `fclose()`, що потребують явного управління вказівниками на файли.
- У С++ для роботи з файлами застосовується клас `fstream`, який забезпечує більш зручну і безпечну роботу з файлами, дозволяючи працювати з потоками вводу/виводу в об'єктно-орієнтованому стилі.

5.4 Переваги використання C++ для розширення функціоналу

- C++ надає набагато більше можливостей для розширення функціоналу завдяки підтримці об'єктно-орієнтованих принципів: спадкування, поліморфізму, абстракції та ін. Це дозволяє організувати код таким чином, щоб легко додавати нові функції без значного переписування старого коду.
- Крім того, C++ підтримує шаблони, що дозволяє створювати універсальні функції та класи для роботи з різними типами даних, що зменшує дублювання коду та підвищує його ефективність.

6. Приклади виконання програми

У цьому розділі наведені приклади виконання програми. Вони демонструють ключові аспекти її роботи: зміну цін товарів перед уцінкою та після, а також процес покупки товарів клієнтом.

1. До та після уцінки товарів

Перед уцінкою товари мають такі ціни:

```
Список товарів:  
- Кокос: 45.00 грн, Кількість: 10  
- Мандарини: 30.00 грн, Кількість: 20
```

Рис. 1:

Після уцінки (на 5%) ціни змінилися:

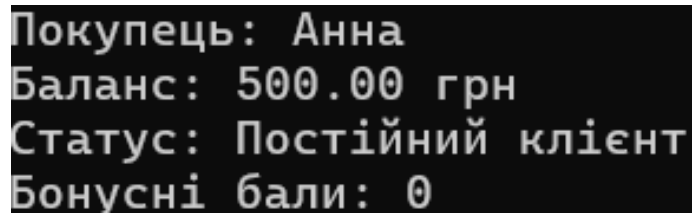
```
Початкова ціна на 'Кокос': 45.00 грн  
Ціни на товари уцінено на 5%.  
Ціна після уцінки: 42.75 грн
```

Рис. 2:

2. Процес покупки товарів клієнтом

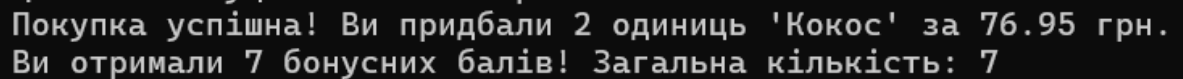
Приклад виконання покупки товарів:

Після завершення покупки:



```
Покупець: Анна
Баланс: 500.00 грн
Статус: Постійний клієнт
Бонусні бали: 0
```

Рис. 3:

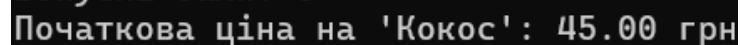


```
Покупка успішна! Ви придбали 2 одиниць 'Кокос' за 76.95 грн.
Ви отримали 7 бонусних балів! Загальна кількість: 7
```

Рис. 4:

3. До та після уцінки товарів

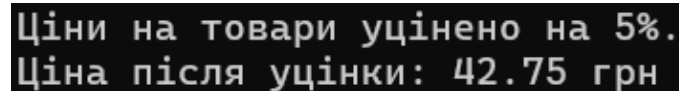
Перед уцінкою товари мають такі ціни:



```
Початкова ціна на 'Кокос': 45.00 грн
```

Рис. 5:

Після уцінки (на 5%) ціни змінилися:



```
Ціни на товари уцінено на 5%.
Ціна після уцінки: 42.75 грн
```

Рис. 6:

7. Висновок

1. Підсумок роботи

У ході виконання даного проєкту було розроблено програму для моделювання процесу купівлі та продажу товарів у магазині. Програма включає основні функції, такі як додавання товарів до складу, купівля товарів клієнтами, а також обчислення знижок та виведення інформації про залишки товарів на складі. Реалізовані можливості програми дозволяють ефективно керувати товарообігом і забезпечувати коректну обробку покупок, як для звичайних, так і для постійних клієнтів.

Усі поставлені цілі, включаючи створення базових класів для товарів і клієнтів, реалізацію покупки товарів та впровадження системи знижок, були успішно досягнуті. Програма пройшла тестування, під час якого були перевірені всі основні функції, а також взаємодія між класами. Результати тестування показали, що система працює стабільно, всі функції виконуються коректно, а покупка товарів обробляється без помилок.

2. Можливі покращення

Програма має певний потенціал для подальшого розвитку і вдосконалення. Ось кілька можливих напрямів для покращення:

- **Інтеграція графічного інтерфейсу:** Замість текстового вводу та виводу можна додати графічний інтерфейс, що зробить програму більш зручною для користувачів.
- **Розширення функціоналу магазину:** Можна додати можливість онлайн-замовлення товарів, а також інтеграцію з платіжними системами для реалізації функції оплати через інтернет.
- **Покращення системи знижок:** Замість фіксованих знижок можна розробити більш складну систему знижок, наприклад, залежно від обсягу покупки або частоти покупок клієнтом.
- **Розширена база даних:** Для зберігання даних можна використувати більш складну систему бази даних, що дозволить зберігати інформацію про більшу кількість товарів, клієнтів і транзакцій.

3. Перспективи використання проєкту

Цей проєкт має значний потенціал для застосування в реальних умовах. Він може бути використаний як основа для розробки програмного забезпечення для невеликих магазинів або торгових платформ, де важлива обробка замовлень та управління товарними запасами. Завдяки можливості розширення функціоналу, цей проєкт можна адаптувати під більш складні торгові системи або інтегрувати з іншими програмними рішеннями для повноцінної роботи в електронній комерції.

Програма також може бути корисною для навчальних цілей, оскільки дає змогу вивчати основи програмування на C/C++ та розробки об'єктно-орієнтованих програм, а також принципи організації роботи в магазині та управління товарообігом.