Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Курсова робота**

з предмету: «Програмування, частина 2 (Об’єктно-орієнтоване програмування)»

на тему :«Базові принципи об’єктно-орієнтованого програмування»

Індивідуальне завдання: «Аптека»

Виконав:

ст. гр. КІ-15

Зектер В.О.

Прийняв:

Козак Н. Б.

Львів-2020

# **Анотація**

Дана курсова робота була підготовлена задля демонстрації знань та навичок, які були отримані під час проходження курсу "Програмування, частина 2 (Об'єктно-орієнтовне програмування)".

Мета даної курсової роботи – засвоєння основних методів та алгоритмів об’єктно-орієнтованих структур.

У цій роботі я програмно реалізував модель аптеки за допомогою методів об’єктно-орієнтованого програмування. Для цього я використав середовище Microsoft Visual Studio 2019, об’єктно-орієнтовану мову програмування С++ та файли формату txt в якості баз даних.

Зміст

[Анотація 1](#_Toc28519)

[Вступ 3](#_Toc10256)

[1. Огляд та обгрунтування вибору технологій об’єктно-орієнтованого програмування 4](#_Toc27436)

[2.1. Опис алгоритму та оцінка його складності. 7](#_Toc10304)

[2.2 Аналіз особливостей алгоритму. 7](#_Toc13496)

[2.3 Опис алгоритму у псевдокоді 8](#_Toc32610)

[2.4.Розробка граф-схеми алгоритму 9](#_Toc23970)

[Пояснення до граф-схеми алгоритму: 10](#_Toc7385)

[3. Програмна реалізація алгоритму згідно індивідуального завдання 10](#_Toc21309)

[3.1 Опис структур даних програми 10](#_Toc3623)

[3.2. Діаграма класів 12](#_Toc21657)

[3.3. Опис реалізації основних функцій/методів 16](#_Toc10086)

[4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання 17](#_Toc22919)

[5.Аналіз продуктивності (часу роботи) та структурної складності реалізованої програм згідно індивідуального завдання 18](#_Toc11449)

[Висновок 21](#_Toc11509)

[Список літератури 22](#_Toc29708)

[Додатки 23](#_Toc27197)

[Лістинг модулів програми згідно індивідуального завдання 23](#_Toc12022)

# **Вступ**

Перша проблема, яка постає перед виконанням завдання – це вибір мови програмування та методи програмування. Мова програмування - набір ключових слів і система правил для конструювання операторів, що складаються з груп або рядків чисел, літер, знаків пунктуації та інших символів, за допомогою яких люди можуть повідомляти комп'ютеру набір команд. З моменту створення комп'ютера створено більше 1000 мов, що відрізняються своїм призначенням і можливостями. Найбільш значимими є: **Машинно-орієнтовані мови** - мови, орієнтовані на певний набір команд процесора - Асемблери. **Мови професійної спрямованості й наукових розрахунків** - Фортран (математичні розрахунки), Кобол (економічні задачі), Алгол (наукові розрахунки). **Універсальні процедурні мови** - Бейсік, Паскаль, С, РL/1, Ада. Універсальні мови відрізняються високою функціональністю й формальністю (відходом від машинної логіки). Вони все більше наближені до природних мов. На них можна створювати будь-який клас задач - ігрові, навчальні, прикладні, системні додатки тощо. **Об'єктно-орієнтовані мови** - мови, що прийшли на зміну процедурним мовам - С++, Оbject Раsсаl. Підтримуючи традиції й можливості процедурних мов, об'єктно-орієнтовані мови додають новий, полегшуючий процес створення великих і професійних програм.

Поставлена у курсовій роботі задача є дуже важкою через складність структури програми. Головна складність полягає у зрозумінні та створенні алгоритму, а також у баготоразовому повторенні однакових операцій. У моделі Аптека ми маємо багато моделей реальних об’єктів, таких як аптекар, медичний препарат чи покупець. Для того щоб зручно керувати цими об’єктами, нам потрібно зосередитися на найважливіших особливостях об’єктів, відкидаючи всі інші, абстрагувати їх. Саме тому у цьому випадку доцільно буде використати методи об’єктно-орієнтованого програмування.

**Об'є́ктно-орієнто́ване програмува́ння**(ООП) — одна з  парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами).

Одною з проблем є вибір реальних об’єктів, які мені потрібно буде абстрагувати у своїй програмі, щоб було зручніше створити реалізацію заданої моделі. Для вирішення цієї проблеми потрібно застосувати об’єктне мисленні.

**Метою виконання курсової роботи** є засвоєння основних методів та алгоритмів об’єктно-орієнтованих структур, принципів побудови відповідних структур, набуття початкових практичних навиків проектування таких засобів.

# **Огляд та обгрунтування вибору технологій об’єктно-орієнтованого програмування**

**Об'єктно - орієнтоване програмування (ООП)** – це модель програмування яка базується на стверджені того, що програма це сукупність об’єктів які взаємодіють між собою. Кожен об’єкт в цій моделі є незалежним, і він здатний отримувати, обробляти дані та відправляти ці дані іншим об’єктам. В ООП використано моделі успадкування, модульності, поліморфізму та інкапсуляції.

Основним поняттям ООП є об'єкт. **Об'єкт** можна визначити як певну сукупність даних(характеристик об'єкта) та методів роботи з ними. Для класифікації об'єктів у ООП використовують класи. **Клас** служить зразком для створення обєкту, тобто об'єкт є нічим іншим, ніж копією класу.

Кожен об'єкт має процедури і функції(те що він уміє виконувати, наприклад,завантажувати файл, відображати картинку і т.д.), які служать для роботи з даними об'єкта. Ці процедури і функції називаються **методами**. Такий спосіб допоможе мені розробити модель покупця, модель адміністратора та моделі ліків, які будуть продаватися у моїй аптеці. Це дасть мені змогу легко створити алгоритм програми та виконувати операції над цими об’єктами. Також я значно спрощу код програми, зроблю його зрозумілішим та заощаджу час на написанні коду.

Абстракція - спосіб виділення найбільш значущих характеристик об'єкта, при цьому менш значущі відкидаються. В ООП абстракція - робота тільки із значимими характеристиками. Суть цього принципу в тому, щоб відокремити складові об'єкти, що складаються з «менших» об'єктів, від ци самих об'єктів, тобто від їх складових. Такий підхід дозволяє працювати безпосередньо з об'єктом, не вдаючись у подробиці, з чого ж він складається і як працює. Абстрагуючи свої об’єкти, я буду вибирати лише ті характеристики, які знадобляться мені для реалізації моделі аптеки. Наприклад, для медичних препаратів я виокремлю лише потрібні мені їх властивості, такі як назву, ціну, кількість, людський орган, на який вони впливають. Але я не буду зважати на зайву інформацію про них, таку як, речовини, які входять до їхнього складу.

Існування ООП можливе завдяки трьом основним парадигмам **на яких базується саме ООП:**

- **Інкапсуляція**. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкта, замість цього надаючи інтерфейс взаємодії з ним. Інкапсуляцію застосовують:

* коли потрібно зберегти деяку ділянку коду без змін з боку користувача;
* коли потрібно обмежити доступ до коду - в зв'язку з унікальністю використовуваних технік, які автор хоче залишити «при собі»;
* коли зміна коду спричинить за собою непрацездатність програми або її злом.

- **Успадкування.** Це означає, що об'єкти (класи) можуть переймати деякі властивості у своїх прабатьків. Це залежить від тієї мови, на якому пишеться програма. Це призводить до повторного використання вже написаного одного разу коду. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюється ієрархія з класів, де від основного класу(так званого, предка) походять усі інші класи. Приклад такого розгалуження зображено на схемі.

- **Поліморфізм** означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по різному на однакові повідомлення. Це спричинене зміною в одного з класів якогось методу(процедури, функції), шляхом запису іншого алгоритму. Поліморфізм дозволяє підвищити відсоток повторного використання коду і скоротити тим самим розмір програми і часові витрати на її написання.

На сьогоднішній день найпопулярніші мови програмування - це об'єктно-орієнтовані, наприклад, С++ і Java. Також існують мови, які не передбачають написання програмних операторів, а програмування відбувається у вигляді візуального проектування за допомогою інтерфейсу мови. Прикладом таких систем є VisualBasic, Delphiі C ++ Builder.

Об'єктна декомпозиція дасть можливість створювати програмні системи меншого розміру шляхом використання загальних механізмів, що забезпечують необхідну економію виражальних засобів. Використання об'єктного підходу істотно підвищить рівень уніфікації розробки і придатність для повторного використання проектів. Системи часто виходять більш компактними, ніж їх структурні еквіваленти, що означає зменшення обсягу програмного коду. Об'єктна модель цілком природна, оскільки в першу чергу орієнтована на людське сприйняття світу, а не на комп'ютерну реалізацію. Само тому для створення цієї програми буде ефективно застосувати методи об’єктно-орієнтованого програмування[1]. Для створення своєї програми я буду використовувати мову програмування С++ та [інтегроване середовище розробки](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) програмного забезпечення Code::Blocks 17.12.

**C++**(**Сі-плюс-плюс**) — мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом (англ. *Bjarne Stroustrup*) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові С. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014.

У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм таких як відеоігри. С++ суттєво вплинула на інші, популярні сьогодні, мови програмування: С# та Java.

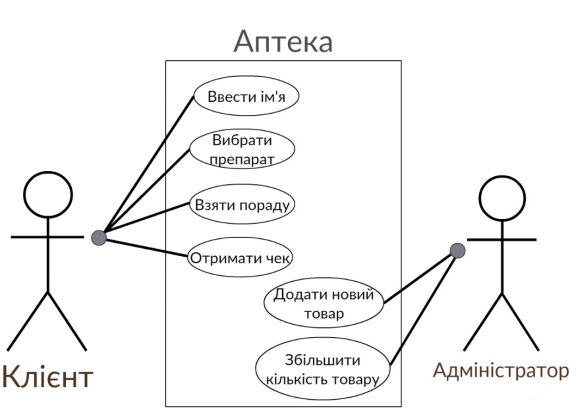
Переваги мови C++:

* **Швидкодія**. Швидкість роботи програм на С++ практично не поступається програмам на С, хоча програмісти отримали в свої руки нові можливості і нові засоби.
* **Масштабованість**. На мові C++ розробляють програми для найрізноманітніших платформ і систем.
* Можливість роботи на **низькому рівні** з пам'яттю, адресами, портами. (Що, при необережному використанні, може легко перетворитися на недолік.)
* Можливість створення **узагальнених алгоритмів** для різних типів даних, їхня спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів.
* Підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, мета програмування (шаблони, макроси)[2].

Для виконання курсової роботи я вибрав мову С++ тому, що вона дозволяє зручно використовувати методи об’єктно орієнтованого програмування

**2. Аналіз та розробка алгоритму згідно індивідуального завдання**

# **2.1. Опис алгоритму та оцінка його складності.**



*Рис.2.1. Діаграма прецедентів програми*

Актори:

Клієнт:

* Ввести ім’я;
* Вибрати препарат;
* Взяти пораду;
* Отримати чек.

Адміністратор:

* Додати новий товар;
* Збільшити кількість товару.

Як видно на діаграмі, адміністратор та клієнт не мають спільних можливостей.

# **2.2 Аналіз особливостей алгоритму.**

Розглянемо детальніше кожну функцію нашої діаграми.

1. Функція Ввести ім’я – доступна лише клієнту. Вона дозволяє клієнту вводити своє ім’я у програму для того, щоб програма могла звертатися до покупця за іменем. Також кожен виданий чек зберігається у пам’яті з ім’ям клієнта, якому він був виданий.
2. Функція Вибрати препарат – доступна лише клієнту. За допомогою цієї функції клієнт може переглянути всі препарати, що є на складі у аптеки, їх ціни та вибрати потрібний собі товар й додати його у чек.
3. Функція Взяти пораду – доступна лише клієнту. Ця функція дає змогу дізнатися клієнту про органи, які лікує кожен з препаратів, які є у наявності. Дізнавшись який препарат потрібен клієнту, він може його вибрати та додати у чек.
4. Функція Отримати чек – доступна лише клієнту. Ця функція завершає покупку клієнта та видає йому чек. У чеку клієнт може побачити список товарів, які він придбав, їхню ціну та загальну суму коштів за всі придбані товари.
5. Функція Додати новий товар – доступна лише адміністратору. За допомогою цієї функції адміністратор має змогу додавати на склад аптеки нові унікальні товари. Для цього він має записати назву, кількість, ціну нового товару та орган, який лікує цей препарат.
6. Функція Збільшити кількість товару – доступна лише адміністратору. Вона дозволяє адміністратору збільшити кількість одиниць вже існуючих товарів. Для цього він повинен вибрати товар, кількість якого він збирається збільшити та ввести кількість одиниць цього товару, на яку він збирається його збільшити.

# **2.3 Опис алгоритму у псевдокоді**

Моя розроблена програма для повноцінного функціонування обробляє та зберігає інформацію у двох файлах: stock.txt та checks.txt.

Файл stock.txt зберігає повну інформацію про препарати, які є у наявності в аптеці. До нього записується номер товару, його назва, кількість, ціна та орган, який лікує цей товар у такому вигляді: Номер,Ціна,Назва,Кількість,Орган.

Для коректної обробки інформації, щоб розділити властивості препарату, я використав символ «,». Для того, щоб відокремити властивості одного препарату від властивостей іншого, я використав символ «#».

Інформацію про препарати з цього файлу зчитує клас Apothecary за допомогою методу char\* readData(). Зберігає інформацію про поточний стан складу аптеки у файл stock.txt також клас Apothecary за допомою методу void rewrite().

У файл checks.txt програма зберігає чеки клієнтів, в які вона записує назви придбаних товарів, їх ціну та загальну суму, яку повинен сплатити клієнт. Ця інформація зберігається у такому форматі:

-------------------------------------------------------

Ім’яКлієнта

НазваПрепарату ------------ ЦінаПрепарату$

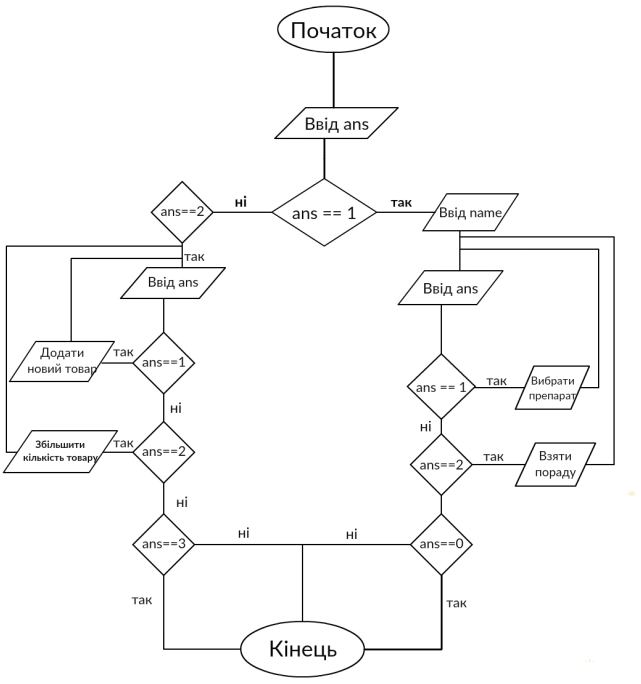
….

Total: ЗагальнаСума

--------------------------------------------------------

У файл checks.txt зберігає чеки клас customer за допомогою методу void saveCheck(char path[]).

# **2.4.Розробка граф-схеми алгоритму**



# **Пояснення до граф-схеми алгоритму:**

1. Початок
2. Вхід до головного меню
3. Вибір варіанту входу до системи: як відвідувач/як адміністратор
4. Якщо **ans==1**, вхід у якості відвідувача

4.1 Поява меню відвідувача

4.2 Ввід імені відвідувача

4.3 Ввід **ans**:

А) якщо **ans==1**, то відбудеться вибір та придбання товару, після -повернення до меню відвідувача

В) якщо **ans==2**, то відбудеться взяття поради, після - повернення до меню відвідувача

С) якщо **ans==0**, або **ans !== 1,2 та 0**, то вихід з системи(кінець)

1. Якщо ans==2, вхід у якості адміністратора
   1. Поява меню адміністратора
   2. Ввід ans:

А) якщо **ans==1**, то відбудеться додавання нового товару, після - повернення до меню адміністратора

В) якщо **ans==2**, то відбудеться збільшення кількості певного товару(поповнення запасів), після - повернення до меню адміністратора

С) якщо **ans==3**, або **ans !== 1,2 та 3**, то вихід з системи(кінець)

# **3. Програмна реалізація алгоритму згідно індивідуального завдання**

# **3.1 Опис структур даних програми**

База даних реалізована двома файлами, які містять різну інформацію. Вони мають різне призначення. Перший файл має назву stock.txt. Він є моделлю складу аптеки, на якому зберігаються медичні препарати. Файл stock.txt грає важливу роль у функціонуванні створеної аптеки, без якого продаж товару буде неможливий. Для того, щоб продати той чи інший товар клієнту, потрібно знати кількість наявного препарату, його ціну та іншу інформацію. Саме цю потрібну нам інформацію містить файл stock.txt. Тому його можна назвати базою даних наявних товарів.

Для того, щоб продати медикамент нам потрібні такі дані:

* Номер препарату(id);
* Назва;
* Ціна;
* Кількість наявних одиниць товару на складі;
* Орган, який лікує препарат.

Номер препарату (id) нам потрібен для зручного звертання до препарату та швидкого пошуку товару та інформації про нього у базі даних. Назва потрібна покупцю, який вибирає товар за його назвою. Ціна потрібна для того, щоб видати покупцю чек з сумою грошей, яку він повинен сплатити за придбаний товар. Якщо покупець не знає який препарат йому допоможе вилікуватись, він може замовити товар за назвою органу, яка теж вказана у базі даних. Файл stock.txt зберігає повну інформацію про препарати, які є у наявності в аптеці. До нього записується номер товару, його назва, кількість, ціна та орган, який лікує цей товар у такому вигляді: Номер,Ціна,Назва,Кількість,Орган#. Для коректної обробки інформації, щоб розділити властивості препарату, я використав символ «,». Для того, щоб відокремити властивості одного препарату від властивостей іншого, я використав символ «#».

Щоб програма могла зчитувати інформацію з бази даних про наявний товар, я створив метод char\* readData(string filepath), який належить класу Administrator. Цей метод приймає в якості аргументу назву файлу у типі даних string. В методі створюється масив data типу char, в який за допомогою циклу for та функції eof() інформація з бази даних записується у масив data[]. Після цього він створює об’єкт класу medicine та додає його у масив об’єктів vector<medicine>stock. Цей об’єкт має такі поля, як id, name, cost, count та DamagedOrgan. За цим метод readData() за допомогою циклу for зчитує інформацію по одному символу до першої коми і записує її у поле створеного об’єкту id. Інформацію зчитану після першої коми до другої метод записує у поле cost того самого об’єкта, від другої до третьої – у поле name, від третьої до четвертої – у поле count і від четвертої до символу «#» - у поле DamagedOrgan. Дійшовши до «#» метод створює новий об’єкт класу medicine та таким самим способом заповнює його поля. Це буде повторюватись поки лічильник циклу не дійде до значення, яке повертає функція strlen(data), а саме – розмір масива data. Таким чином програма зчитує інформацію з бази даних та створює об’єкти препаратів.

Після виконання програми у базу даних переписується вся інформація про поточний стан товарів з створених об’єктів типу medicine назад у файл stock.txt за допомогою методу void rewrite(), який відноситься до класу Administrator. Цей метод відкриває файл stock.txt за допомогою функції ofstream fin("stock.txt"). Потім за допомогою циклу for та функції stock.size(), яка визначає кількість об’єктів препаратів у масиві stock, він отримую доступ до всіх створених об’єктів типу medicine. У циклі метод rewrite записує у файл спочатку інформацію з поля id першого об’єкта, потім записує кому, після якої записує інформацію з поля cost, ставить кому, записує поле name, знову ставить кому, переписує поле name, ставить кому, переписує поле count, ставить кому, переписує поле DamagedOrgan та вставляє символ «#». Таким чином метод переписує інформацію до файлу із всіх об’єктів.

Другий файл, що складає базу даних, має назву checks.txt. Цей файл є базою даних чеків. Він потрібен для того, щоб програма зберігала інформацію про клієнтів, які робили замовлення у моїй аптеці, та інформацію про придбані клієнтом товари. У файл stock.txt програма зберігає чеки клієнтів, в які вона записує назви придбаних товарів, їх ціну та загальну суму, яку повинен сплатити цлієнт. Ця інформація зберігається у такому форматі:

-------------------------------------------------------

Ім’яКлієнта

НазваПрепарату1 ------------ ЦінаПрепарату1$

….

НазваПрепаратуN------------- ЦінаПрепаратуN$

Total: ЗагальнаСума

--------------------------------------------------------

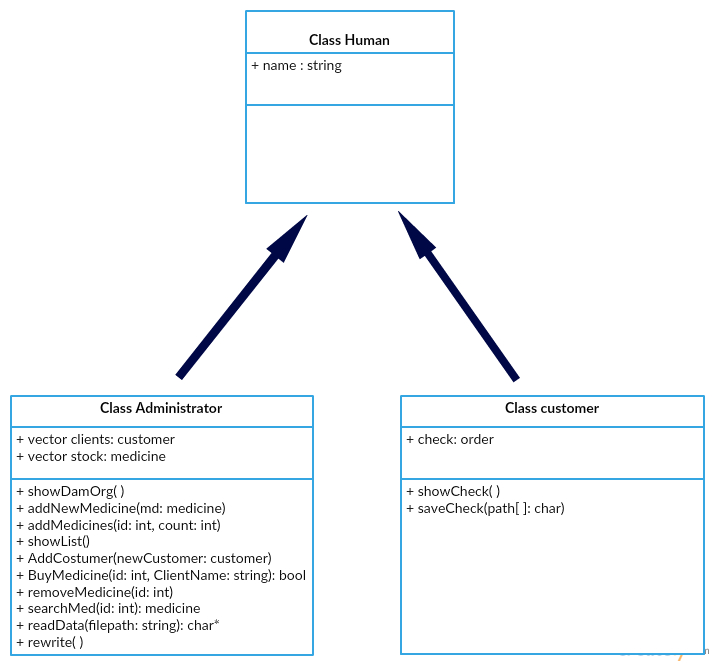
У файл checks.txt зберігає чеки клас customer за допомогою методу void saveCheck(char path[]). В якості аргументу цей метод приймає назву файлу. Коли ми викликаємо метод saveChech(), спочатку він відкриває файл за допомогою функції fin() та записує туди ім’я клієнта, який зробив замовлення. Потім він за допомогою циклу for та функції size() перебирає всі об’єкти типу medicine у масиві об’єктів drugs, який є полем класу order, та переписує з них поля name та cost у файл. При виконанні циклу поля cost додаються у змінну типу float – sum. Після проходження циклу метод записує у файл дані із змінної sum як загальну суму коштів за усе замовлення.

Таким чином програма використовує файл stock.txt для вводу та виводу даних і файл checks.txt тільки для виводу.

# **3.2. Діаграма класів**

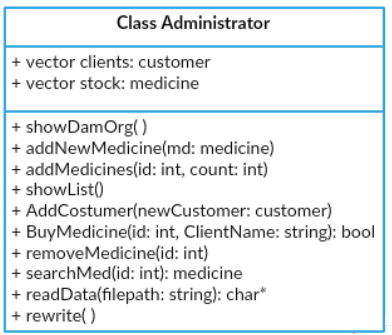
Створена програма містить в собі п'ять класів. Розглянемо детальніше структуру кожного з них.

Клас Administrator та customer мають одне спільне поле, тому можна використати один з принципів ООП – наслідування. Для цього я створив загальний абстрактний клас, який я назвав Human.



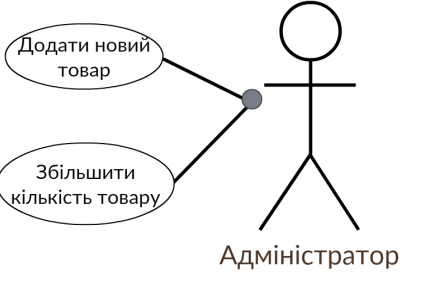
*Рис.3.2.1 Структура наслідування*

**Клас Administrator**



*Рис.3.2.2 Структура класу Administrator*

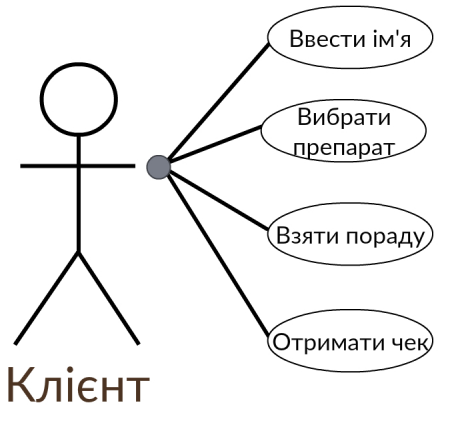
Клас Administrator є абстракцією реального адміністратора аптеки, який має усі необхідні здібності для роботи аптеки. Цей клас має поле яке є масивом об’єктів типу customer для різних клієнтів та масив об’єктів типу medicine для виконання дій з медичними препаратами.



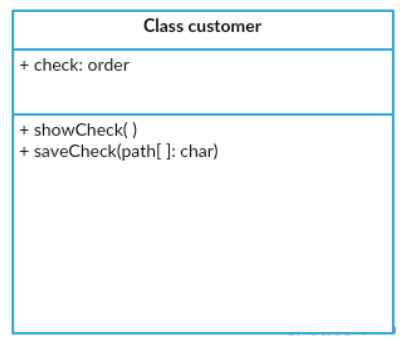
*Рис.3.2.3 Use Case класу Administrator*

**Клас customer**

Цей клас є абстрактною моделлю реального покупця, який має всі необхідні здібності для придбання товару у аптеці. Цей клас має змінну check типу order для зберігання даних про вибрані покупцем товари у аптеці.



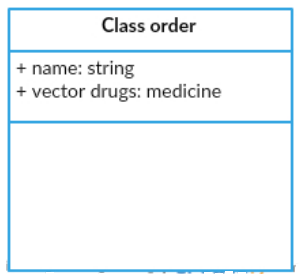
*Рис.3.2.4**Use Case класу customer*



*Рис.3.2.5 Структура класу customer*

**Клас order**

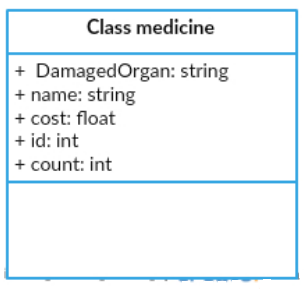
Цей клас є абстрактною моделлю замовлення клієнта. Об’єкт цього класу потрібен для зберігання переліку товарів, які куплені клієнтом, у масиві об’єктів типу medicine – drugs.



*Рис.3.2.6 Структура класу order*

**Клас medicine**

Клас medicine створює реальну модель медичних препаратів. Об’єкти цього класу містять необхідні дані про медикаменти. У полі id міститься інформація про порядковий номер товару, у cost – про ціну товару, у name – назва товару, у count – кількість наявного на складі товару, у DamagedOrgan – орган, який лікує даний медичний апарат.



*Рис.3.2.7 Структура класу medicine*

# **3.3. Опис реалізації основних функцій/методів**

У цьому розділі будуть описані алгоритми робот основних методів програми. Методи мають два класи: Administrator та customer.

**Основні методи класу Administrator**

Метод addNewMedicine( )

Цей метод дає змогу адміністратору додавати новий унікальний елемент у базу даних (у файл stock.txt). Метод приймає в якості аргументу об’єкт типу medicine та за допомогою функції push\_back( ) додає його у масив об'єктів stock, у якому зберігаються всі об’єкти наявних товарів. Метод нічого не повертає.

Метод addMedicines( )

Цей метод дає змогу адміністратору додавати будь-яку кількість одиниць товару. Метод приймає два аргументи типу int. Перший аргумент позначає порядковий номер товару, кількість якого адміністратор бажає збільшити у базі даних, а другий аргумент – кількість вибраного товару, на яку хоче його збільшити адміністратор. За допомогою циклу for та функції size( ) метод перебирає всі об’єкти типу medicine масиву об’єктів stock та шукає той, порядковий номер якого буде збігатися з тим, що приймається в якості елементу. Коли такий об’єкт буде знайдено, метод збільшить його поле count на ту кількість, яка вказана у аргументі. Якщо такого елементу не буде знайдено, то на екрані з’явиться помилка «Препарат не знайдено!». Метод не повертає жодного значення.

**Основні методи класу customer**

void saveCheck(char path[ ])

Даний метод був розроблений для зберігання чеків у базі даних клієетів. В якості аргументу цей метод приймає назву файлу. Коли ми викликаємо метод saveChech(), спочатку він відкриває файл за допомогою функції fin() та записує туди ім’я клієнта, який зробив замовлення. Потім він за допомогою циклу for та функції size() перебирає всі об’єкти типу medicine у масиві об’єктів drugs, який є полем класу order, та переписує з них поля name та cost у файл. При виконанні циклу поля cost додаються у змінну типу float – sum. Після проходження циклу метод записує у файл дані із змінної sum як загальну суму коштів за усе замовлення.

Метод void showCheck()

Цей метод дає змогу вивести чек з покупками клієнта на екран. Метод не приймає аргументів та не повертає значень. За допомогою циклу for та функції size( ) він перебирає об’єкти типу medicine у масиві check та виводить по черзі їх поля name та cost на екран у формі: «НазваПрепарату-------ЦінаПрепарату». Паралельно з тим він додає поля cost у зміну sum для того, щоб потім вивести на екран загальну суму, яку повинен сплатити клієнт.

# **4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання**

Під час написання курсової роботи виникли помилки різного характеру.

Синтаксичні помилки, викликані неправильним записом операторів мови програмування, виявляються на етапі трансляції програми. При цьому виводиться повідомлення, яке вказує оператор, в якому допущена помилка, місце в операторі, де виявлена помилка, і текст, що пояснює причину помилки. Синтаксичні помилки виникають у разі порушення правил запису мовних конструкцій.

За допомогою вбудованого у Code::Blocks зневаджувача були виправлені синтаксичні помилки. При наявності синтаксичної помилки зневаджувач повідомляв про це, відзначаючи проблемну частину коду.

Логічні помилки - це помилки в логіці роботи програми. Наприклад, програмний код виглядає абсолютно правильним, але працює неправильно. Такі помилки виявити і виправити набагато складніше. Помилки логічного типу були виправлені за допомогою статичного аналізу та покрокового тестування.

**4.1 Відлагодження та тестування окремих частин програми**

Дана курсова робота була розроблена у середовищі Code::Blocks, і має наступний функціонал: імітування справжньої аптеки, у якій ви можете виступати в якості покупця, який бажає придбати медикаменти, або в якості адміністратора, який слідкує за запасом певних препаратів та може додавати нові. Дана програма не потребує для своєї роботи середовища, у якому була розроблена, адже користувач може оперувати простим exe-файлом, для роботи якого знадобиться лише операційна система Windows. Даний варіант програми був розроблений за участю операційної системи Windows 10.

**4.2 Відлагодження роботи програми в цілому**

Після усунення всіх синтаксичних та логічних помилок, програма успішно “зібралась” та розпочала роботу, результати якої можна побачити у наступному розділі.

# **5.Аналіз продуктивності (часу роботи) та структурної складності реалізованої програм згідно індивідуального завдання**

**5.1 Опис структурної складності виконання програми**

Структурна складність програм визначається:

1. Числом взаємодіючих компонентів
2. Числом зв'язків між компонентами

3. Складністю взаємодією компонентів (кількість зв'язків)

Різноманітність поведінки програми і зв'язків між її вхідними результуючими даними визначається набором маршрутів, за якими виконується програма.

Складність програмного модуля пов'язана не стільки з розміром (кількістю команд) програми, скільки з числом маршрутів її виконання і складністю.

Маршрути можливої обробки даних визначають складність розробки програми.

Дану метрику складності можна використовувати для оцінки трудомісткості тестування і супроводу модуля, а також для оцінки потенційної надійності його функціонування.

Маршрути виконання програмного модуля поділяються на:

1. обчислювальні маршрути
2. маршрути прийняття логічних рішень

Складність обчислення маршрутів оцінюється формулою:

IMG_256, де m - кількість маршрутів виконання програми, І-число даних, які обробляються в i-му маршруті, V-число значень, оброблюваних даних j-ого типу (2 ≤ ≤ 5)

Загальна кількість арифметичних операцій не виходить за межі 5-10%, тому обчислювальні маршрути визначають структурну складність програми.

Дана курсова робота не містить у своїй структурі складних обчислень, лише невелику кількість класів, що взаємодіють між собою, а також декілька логічних маршрутів, що дозволяє сказати про те, що дана програма не є структурно складною.

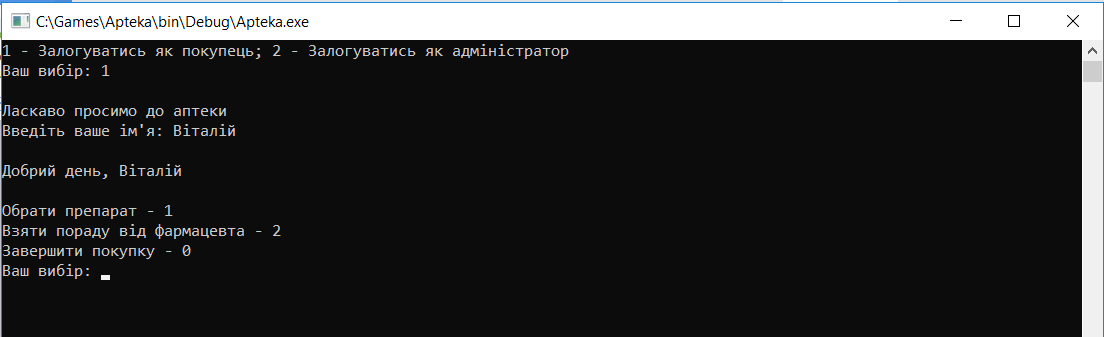
**5.2 Аналіз часу виконання програми**

У початковому меню програми ми можемо вибрати вхід від імені клієнта або від імені адміністратора.

# 

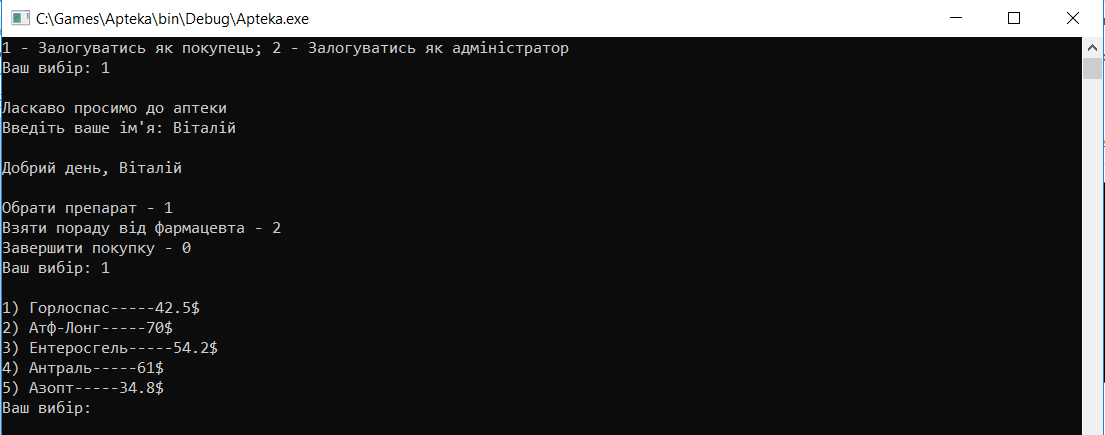
*Скріншот 5.1 Початкове меню*

Вибравши вхід від імені покупця, ми можемо обрати медичний препарат за назвою або за призначенням.



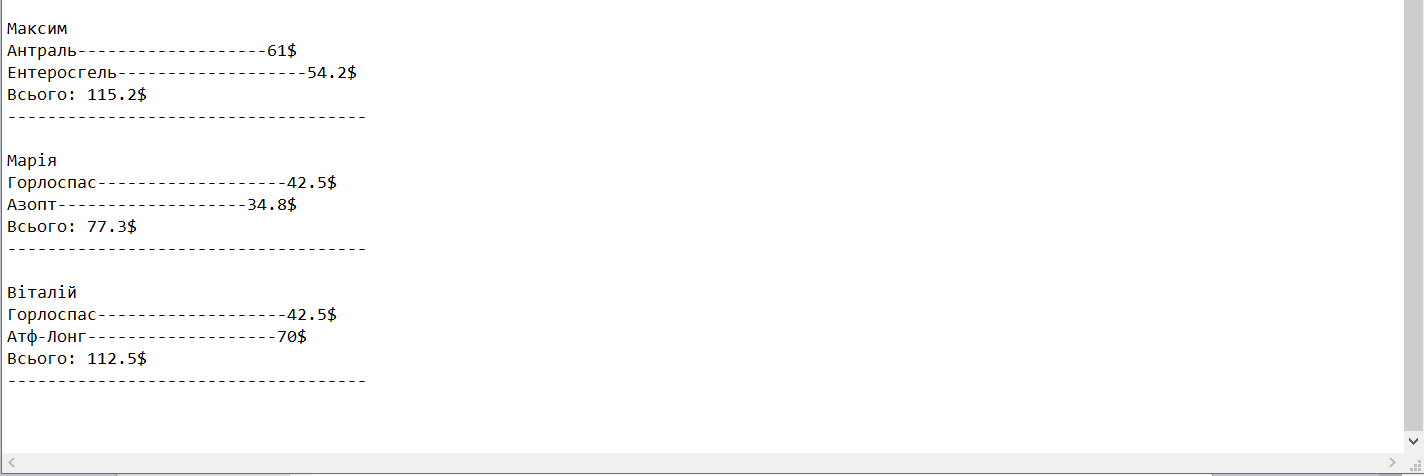
# *Скріншот 5.2 Меню клієнта*

Якщо клієнт бажає обрати товар, то на екран виводиться список доступних медичних препаратів. На цьому етапі клієнт повинен обрати потрібний йому товар.



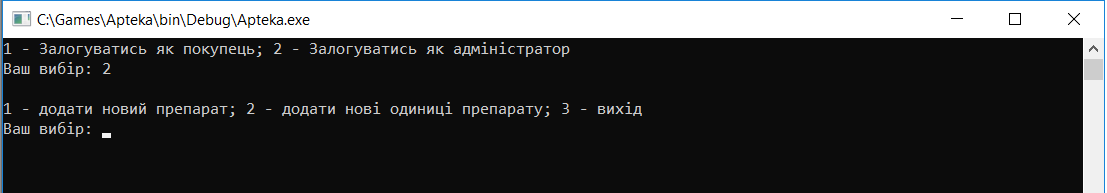
*Скріншот 5.3 Список препаратів*

Закінчивши покупку, у базі даних про клієнтів записується чек клієнта.



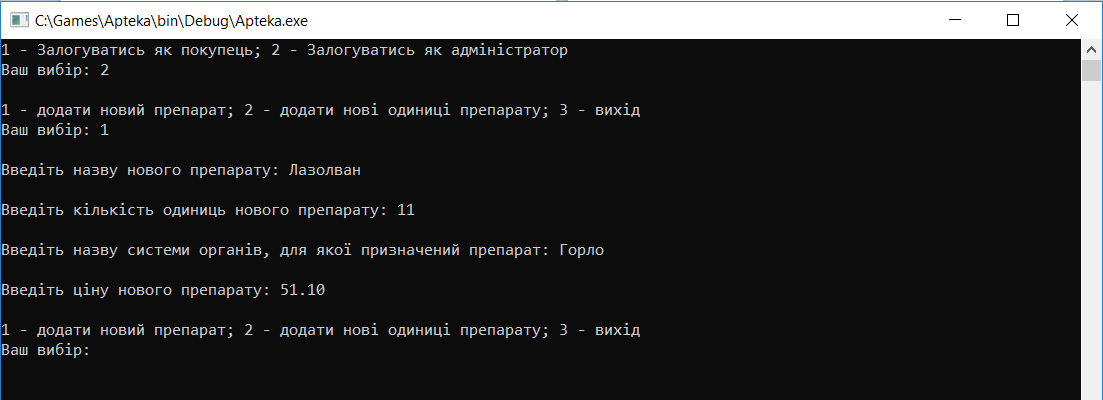
*Скріншот 5.4 Вміст файлу checks.txt*

Якщо користувач зробить вхід від імені адміністратора, то в його меню з’являться такі дії: додати новий унікальний товар, додати кількість існуючого товару.



*Скріншот 5.5 Меню адміністратора*

Щоб додати новий унікальний товар, адміністратор повинен ввести його назву, ціну та кількість.



*Скріншот 5.6 Додавання нового препарату*

“Збирання” проекту, тобто створення exe-файлу, відбувається доволі швидко, в межах від 3 до 10 секунд. Повторне завантаження програми, вже з існуючим exe-файлом, відбувається майже миттєво.

# **Висновок**

В ході курсової роботи я на практиці застосував свої знання з об’єктно-орієнтованого програмування. Поглибив розуміння основних принципів об’єктно-орієнтованої ідеології програмування. Відпрацював на практиці основні методи та засоби об’єктно-орієнтованого програмування. Навчився розробляти ієрархію класів. Практично застосував такі поняття, як інкапсуляція, наслідування, перевантаження та перевизначення функцій. Набув практичних навичок роботи.

# **Список літератури**

1. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. Пер. с англ. - М. Бином
2. [Т. Вельдхаузен. Techniques for scientific C++](http://osl.iu.edu/~tveldhui/papers/techniques/)
3. Прата С. Язык программирования C++. Лекции и упражнения. 6-е издание / Стивен Прата; пер. с англ. – М.: ООО «И.Д. Вильямс», 2012. – 1248 с.: ил. – Парал. тит. англ.
4. Полный справочник по C++, 4-е издание / Герберт Шилдт — М.: «Вильямс», 2011. - 800 с.

5. Chris Giametta «Pro Flex on Spring», 2009.- p.445.

# **Додатки**

# **Лістинг модулів програми згідно індивідуального завдання**

Class Administrator

#include<string>

#include<vector>

#include"medicine.h"

#include"order.h"

#include<iostream>

#include<fstream>

#include"customer.h"

#include"windows.h"

#include<cstdlib>

using namespace std;

class Apothecary

{

public:

vector<customer>clients;

vector<medicine>stock;

void showDamOrg(){

for(int i = 0; i < stock.size(); i++){

cout<<endl<<stock[i].id<<") "<<stock[i].DamagedOrgan<<"--"<<stock[i].name;}

}

void addNewMedicine(medicine md){

this->stock.push\_back(md);

}

void addMedicines(int id, int count){

bool findItem = false;

for(int i = 0;i < stock.size();i++){

if(stock[i].id == id){

stock[i].count += count;

findItem = true;

break;}

}

if (findItem == false){

cout<<endl<<"Препарат не знайдено!";}

}

void showList(){

for(int i = 0; i < stock.size(); i++){

cout<<endl<<stock[i].id<<") "<<stock[i].name<<"-----"<<stock[i].cost<<"$";}

cout<<endl;

}

void AddCostumer(customer newCustomer){

this->clients.push\_back(newCustomer);

}

bool BuyMedicine(int id, string ClientName){

int index;

for(int i = 0;i < clients.size(); i++){

if(clients[i].name == ClientName){

index = i;

clients[i].check.name = ClientName;

break;}}

medicine m = searchMed(id);

if(m.id != -1){

clients[index].check.drugs.push\_back(m);

removeMedicine(id);

return true;

}

else return false;

}

void removeMedicine(int id){

for(int i = 0; i < stock.size();i++){

if(stock[i].id == id){

if(stock[i].count > 1){

stock[i].count--;}

else stock.erase(stock.begin() + i);

break;}}

}

medicine searchMed(int id){

for(int i = 0;i < stock.size();i++){

if(stock[i].id == id)

return stock[i];}

medicine m;

m.id = -1;

return m;

}

char\* readData(){

char data[20000] = "";

ifstream fin("stock.txt");

for(int i = 0;!fin.eof();i++){

fin>>data[i];

data[i + 1] = '\n';}

fin.close();

string temp = "";

int count = 0;

medicine m = medicine();

for(int i = 0;i < strlen(data);i++){

if(data[i] == '#'){

m.DamagedOrgan = temp;

temp = "";count = 0;

this->stock.push\_back(m);}

else if(data[i] == ','){

switch(count){

case 0:{

m.id = stoi(temp);

break;}

case 1:{

m.cost = stof(temp);

break;}

case 2:{

m.name = temp;

break;}

case 3:{

m.count = stoi(temp);

break;}

}

temp = "";count++;}

else{

temp += data[i];}}

return data;

}

void rewrite(){

ofstream fin("stock.txt");

for(int i = 0; i < stock.size(); i++){

fin<<stock[i].id<<','<<stock[i].cost<<','<<stock[i].name<<','<<stock[i].count<<','<<stock[i].DamagedOrgan<<'#'<<endl;}

fin.close();}

};

Клас Customer

#include<string>

#include<vector>

#include"medicine.h"

#include"order.h"

#include<iostream>

#include<fstream>

#include"windows.h"

using namespace std;

class customer

{

public:

string name;

order check;

customer(string name) { this->name = name; }

void showCheck(){

float sum = 0;

for(int i = 0; i < check.drugs.size(); i++){

cout<<endl<<check.drugs[i].name<<"-------------------"<<check.drugs[i].cost<<"$";

sum+=check.drugs[i].cost;}

cout<<endl<<"Всього: "<<sum<<"$";}

void saveCheck(char path[]){

ofstream fin(path, ios\_base::app);

fin<<name;

float sum = 0;

for(int i = 0;i < check.drugs.size();i++){

fin<<endl<<check.drugs[i].name<<"-------------------"<<check.drugs[i].cost<<"$";

sum+=check.drugs[i].cost;}

fin<<endl<<"Всього: "<<sum<<"$"<<endl<<"------------------------------------"<<endl<<endl;

fin.close();}

};

Клас Medicine

#include<string>

using namespace std;

class medicine

{

public:

string DamagedOrgan;

string name;

float cost;

int id;

int count;

medicine() {}

medicine(int id, float cost, string name, int count) { this->cost = cost; this->count = count; this->id = id; this->name = name; }

};

Клас Order

#include<string>

#include<vector>

#include"medicine.h"

using namespace std;

class order

{

public:

string name;

vector<medicine>drugs;

};

Функція int main()

#include<iostream>

#include<string>

#include<vector>

#include<fstream>

#include<conio.h>

#include"medicine.h"

#include"order.h"

#include"customer.h"

#include"apothecary.h"

#include"windows.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Apothecary d1;

d1.readData();

int ans;

cout<<"1 - Залогуватись як покупець; 2 - Залогуватись як адміністратор\n";

cout<<"Ваш вибір: ";

cin>>ans;

cout<<endl;

if(ans == 1){

cout<<"Ласкаво просимо до аптеки\n";

cout<<"Введіть ваше ім'я: ";

string name;

cin>>name;

cout<<endl;

d1.AddCostumer(customer(name));

cout<<"Добрий день, "<<name;

cout<<endl;

while(true){

cout<<endl<<"Обрати препарат - 1 "<<endl<<"Взяти пораду від фармацевта - 2"<<endl<<"Завершити покупку - 0\n";

cout<<"Ваш вибір: ";

cin>>ans;

if(ans == 1){

int idi;

d1.showList();

cout<<"Ваш вибір: ";

cin>>idi;

d1.BuyMedicine(idi, d1.clients[0].name);}

else if(ans == 2){

int idi;

d1.showDamOrg();

cout<<endl<<"Ваш вибір: ";

cin>>idi;

d1.BuyMedicine(idi, d1.clients[0].name);}

else if(ans == 0){

d1.clients[0].showCheck();

d1.rewrite();

\_getch();

break;}

system("cls");}

cout<<endl<<"Гарного дня, "<<name<<'!'<<endl;

d1.clients[0].saveCheck("checks.txt");}

else if(ans == 2){

while(1){

cout<<"1 - додати новий препарат; 2 - додати нові одиниці препарату; 3 - вихід\n";

cout<<"Ваш вибір: ";

cin>>ans;

if(ans == 1){

medicine m;

m.id = d1.stock.size()+1;

cout<<endl<<"Введіть назву нового препарату: ";

cin>>m.name;

cout<<endl<<"Введіть кількість одиниць нового препарату: ";

cin>>m.count;

cout<<endl<<"Введіть назву системи органів, для якої призначений препарат: ";

cin>>m.DamagedOrgan;

cout<<endl<<"Введіть ціну нового препарату: ";

cin>>m.cost;

d1.addNewMedicine(m);

cout<<endl;}

else if(ans == 2){

d1.showList();

int idi;

int count;

cout<<endl<<"Введіть id препарату: ";

cin>>idi;

cout<<endl<<"Введіть кількість: ";

cin>>count;

d1.addMedicines(idi, count);

cout<<endl;}

else if(ans == 3){

d1.rewrite();

break;}

}

}

system("pause");

}