

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ім. Т. Г. Шевченка
Кафедра інформаційних систем та технологій

Пояснювальна записка

до курсової роботи

З дисципліни “Об’єктно-орієнтоване програмування”

З теми “Програмний застосунок магазину обліку одягу ARMANI”

студент (ка) 1 курсу групи ІР-12

Керівник курсової роботи:

Ампілогов Ігор Андрійович
(прізвище, ім’я та по батькові)

к.т.н. техн. наук доц. к. інф. с. та т.
(науковий ступінь, вчене звання або посада)

(дата здачі) (підпис)

Лісневський Ростислав Валерійович
(прізвище, ім’я та по батькові)

ЗМІСТ КУРСОВОЇ РОБОТИ

Пояснювальна записка включає в себе наступні розділи:

Вступ (ст. 3)

Розділ 1 “Постановка задачі, аналіз предметної області, вибір методів та технологій” (ст. 4 – 13)

Розділ 2 “Проектування архітектури та реалізація програмного застосунку” (ст. 14 – 31)

Розділ 3 “Тестування та дослідження отриманих результатів” (ст. 32 – 36)

Список джерел та використаної літератури у курсовій роботі (ст. 39)

Додатки(ст. 40 – 48)

Додатки включатимуть в себе:

Технічне обладнання

Код програми

Інструкція користувача

Мова програмування, що буде використана при написанні коду програми:

C++

ВСТУП

Потреби сучасної людини з кожним днем все більше і більше потребують занурення у цифровий світ. Якщо приблизно двадцять років тому похід до магазину або до відділення банку було звичайною справою, то сьогодні людина має змогу зробити все це, лежачи на дивані у своєму будинку.

Людина має змогу обрати, забронювати, замовити, оплатити, та навіть повернути товар з онлайн - магазину одяжі, навіть якщо він знаходиться в тисячах кілометрів від фізичного відділення магазину. Все це стало можливим завдяки прогресу в галузі інформаційних систем та технологій.

Тож, дивлячись на всі ці винаходи, я вирішив власноруч написати та реалізувати сайт магазину обліку одяжі брендового одягу ARMANI. В моєму застосунку користувач зможе продивитися наявний одяг, замовити та придбати його, а також багато інших функцій. У даній курсовій роботі я опишу процес створення та реалізації програмного застосунку онлайн магазину одяжі ARMANI, яка буде написана використовуючи високу мову програмування, таку як C++.



РОЗДІЛ 1

Вибір теми.

Тему курсової роботи “Програмний застосунок магазину обліку одєжі ARMANI”, я обрав через те, що особисто мені цікаво розробити програмний застосунок, який буде використовуватись багатьма людьми, особливо зараз, коли інтернет-технології розвиваються величезними кроками, і програмні застосунки стають все більш і більш затребуваними у суспільстві.

Прогрес не стоїть на місці, і якщо приблизно двадцять років тому, люди і уявити не могли, як можна купувати речі не виходячи за межі свого дому, то зараз словосполучення “Онлайн - шопінг” є абсолютно доречним, і логічним. Також ще одним важливим фактором вибору цієї теми є зручність. Якщо буденний похід до магазину може викликати купу незручностей, такі як: великі витрати часу як для того, щоб банально дістатися до магазину, так і для вибору необхідного товару; складність у знаходженні потрібних речей, бо як правило при відвідуванні магазинів одєжі, великою складністю для нас стає знайти “той самий” товар, який нас цікавить; при купівлі багатьох речей, стає логічне питання - “Як мені дійти до дому?”, носіння великих пакетів та сумок з речами, дуже сильно обмежують твої дії, не кажучи все про простий дискомфорт та тяжкість.

На фоні цих проблем, дуже яскраво і чітко видно переваги програмного застосунку. Застосунок буквально дозволяє здійснити купівлю товарів майже з будь-якого місця на планеті, бо це все абсолютно не важливо, так як твій товар все одно буде доставлено саме до того місця, яке обираєш саме ти. Проблема з знаходженням конкретного товару також вирішується сама собою, бо абсолютно весь товар, що маєтсья у наявності в магазині, буде вміщений у маленький екран. Там ви не тільки маєте змогу знайти

потрібну вам річ, але й продивитися наявні кольори, розмір, та ціну товару, і одним рухом руки цей товар може бути переміщений у кошик для його купівлі.

Розділ 1.1

Метою даної курсової роботи є написання застосунку магазину обліку одяжі ARMANI.

Основні завдання, які включають в себе такі пункти:

1. Розробка алгоритму виконання логіки застосунку.
2. Написання та розробка коду для реалізації застосунку.
3. Тестування та перевірка на працездатність коду.
4. Оформлення звіту курсової роботи.

Опис предметної області:

Програмний застосунок магазину обліку одягу ARMANI - це комп'ютерна програма, розроблена для полегшення процесу обліку та управління продажами одягу в магазинах ARMANI. Вона дозволяє вести облік всіх товарів, які продаються в магазині, контролювати залишки, керувати процесом закупівель та продажу.

Застосунок магазину обліку одягу ARMANI містить в собі різноманітні функції, такі як ведення каталогу товарів, керування замовленнями та постачанням, та багато інших корисних функцій. Програма дозволяє ефективно керувати процесами магазину і забезпечити високу якість обслуговування клієнтів.

У загальному, програмний застосунок магазину обліку одягу ARMANI є потужним інструментом, що дозволяє підвищити ефективність діяльності.

Для створення даного застосунку я буду використовувати технології об'єктно-орієнтованого програмування, та мову програмування високого рівня C ++.

Розділ 1.2

Аналіз та вибір математичних методів та технологій програмування.

Для розробки програмного застосунку магазину обліку одягу ARMANI я буду використовувати високу мову програмування C++, та бібліотеки, які наявні в цій мові, також будуть описані математичні методи, які можна доречно використати у створенні застосунку по темі курсової роботи.

Мій застосунок може вести облік усіх товарів, наявних в магазині, окремий облік залишків товарів в магазині. Також оформлювати замовлення на товари, керувати процесом отримання товару постачальниками.

Користуючись різними бібліотеками, що наявні в мові C ++, з'являється можливість графічної реалізації інтерфейсу.

Мною буде використаний графічний інтерфейс для C++ під назвою SFML.

Дана графічна бібліотека допоможе мені реалізувати графіку у моїй курсовій роботі, та зв'язати її з об'єктно-орієнтованою частиною мого застосунку.

При виборі графічної бібліотеки я відокремив для себе дві - SFML та qt, які на мій погляд найкраще б впоралися з умовами та потребами моєї курсової. При детальному вивченні обох бібліотек я вирішив обрати саме SFML, бо ця бібліотека набагато легша в освоєнні, ніж qt, та найголовніше це те, що для мого програмного застосунку цілком вистачить функціоналу даної бібліотеки.

Інтерфейс даної бібліотеки дозволяє створювати та додавати прості геометричні фігури на екран, такі як коло, квадрат, n-кутник та робити з ними різні дії, такі як зміна кольору, додавання рамок, розтягування в висоту та ширину, збільшення та зменшення їх розмірів, накладання картинки, зміна прозорості, та багато чого іншого.

Також, важливою особливістю SFML, є можливість реалізації декількох сторінок, між якими користувач матиме можливість перемикатись за допомогою т. н. “кнопок”. Оформлення даних вікон буде відбуватись таким чином: Користувач наводить курсор миші на кнопку “Каталог”, і при наведенні курсору на границі або на саму кнопку, SFML в змозі це передбачити, та передати дані про положення курсору, і кнопка може на це якось відреагувати, наприклад, змінити колір або свій розмір. Те ж саме і відбувається при натисканні лівою кнопкою миші на кнопку, бібліотека передає та отримує ці дані,

Після запуску моєї програми користувач зможе подивитися весь товар, що мається у наявності на даний момент. Може обрати щось із запропонованого списку, та “купити” цей товар у той самий момент. Також буде реалізовано вивід у файл “квитанції” після оформлення покупки, у якому буде наявна інформація про товар: ціна на момент покупки, дата та

час операції. Квитанція буде доступна у форматі txt, яку при бажанні користувач зможе зберегти.

При написанні програмного застосунку буде доречно звернути увагу на деякі методи, такі як логістика. Врахування логістики при створенні застосунку магазину обліку речей є дуже важливим, бо саме завдяки цьому методу, можливо розрахувати час, за який певний товар дістанеться до свого потенціального покупця зі складу. Використовуючи метод логістики, також можна розрахувати скільки разів на місяць буде можливість завозити новий товар до магазину, поповнювати асортимент, та багато чого іншого.

Також корисно та доречно у застосунку використати метод сортування товарів за типом, ціною, виробником та іншими критеріями. Це допоможе швидше і легше знайти потрібну річ для потенційних покупців. Так, наприклад, якщо дівчина хоче знайти для себе недорогий верхній одяг, наприклад куртки до 5000 грн, вона може використати спеціальні фільтри, в яких вкаже свою стать, тип одягу, та ціну. І після вибору, заздалегідь відсортовані речі будуть показані покупцю. Цей метод дозволяє поділити товари на “умовні” групи, завдяки чому пошук потрібних речей буде займати набагато менше часу. Також, це дозволить вести облік речей по групам, розуміти, які речі продаються краще, або навпаки гірше. З’являється можливість бачити, скільки було продано чоловічого верхнього одягу, як часто люди купували взуття вартістю більше 7500 гривень, та багато іншого.

Також окремої уваги потребує метод обліку речей, який є основним при написанні мого програмного застосунку, бо суть завдання полягає саме в цьому. Метод обліку речей включає в себе ведення повного списку всіх речей ARMANI, що мають у наявності у даний момент часу. При купівлі

речі, список обліку речей буде оновлюватись, та зменшуватись на кількість яка буде дорівнювати числу проданих товарів. Список речей буде поповнюватись завдяки поставкам товарів зі складів або з заводу, які будуть відбуватись деяку кількість разів на місяць, в чому також буде приймати свою участь метод логістики, що був описаний та прокоментований вище. Для більш зручного ведення обліку мною було прийнято рішення вести не один великий список речей, а розбити його на декілька підгруп, наприклад: взуття, аксесуари, верхній одяг, нижня білизна, та тощо. Це робиться для кращого розуміння, які ж саме типи та види речей були продані, щоб потім ефективно поповнювати списки цих речей, а при використанні одного великого списку речей, я не мав би об'єктивної можливості побачити і оцінити, які ж саме товари користуються попитом, і які треба поповнювати.

Також, важливою частиною даного методу, є можливість рахувати місячний дохід з продажу товарів, та мати об'єктивну оцінку по фінансовій складовій даного магазину, наскільки магазин вийшов в “плюс” або “мінус”.

І взагалі облік речей є дуже корисним та дієвим методом, який дозволяє знати у будь-який момент кількість наявного товару на складі, ціну, за яку продаються товари, та запобігає краді товарів, адже зі списку товар не може просто зникнути, як це може відбутися на складі.

Тож я вважаю, що реалізація застосунку на мові програмування є досить доречною ідеєю, як і використання методів, що були описані мною вище.

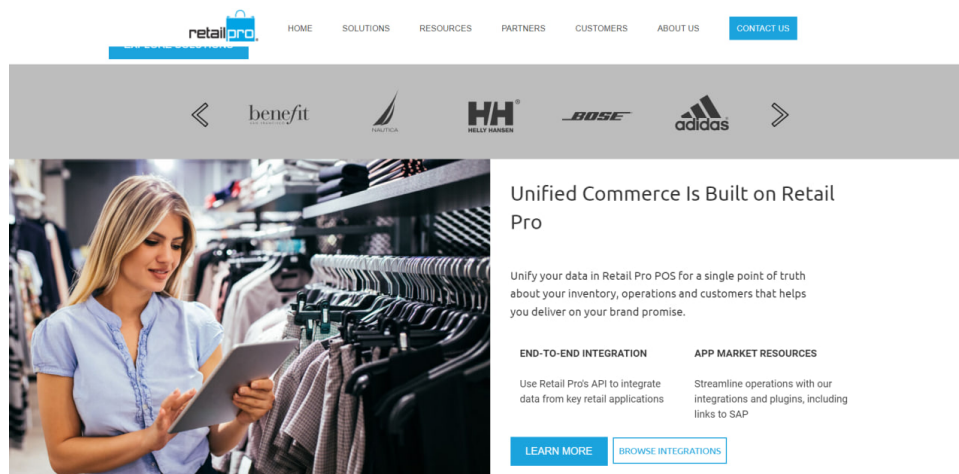
Все це дасть мені змогу не тільки написати та розробити працюючий ПЗ, а й розвине та покращить мої навички у цій галузі, які 100% допоможуть мені у майбутньому.

Розділ 1.3

Порівняння з існуючими аналогами:

Цілком очевидно, що ідеєю розробки програмного застосунку обліку речей, я зацікавився не перший, і на світі є багато аналогів мого застосунку, які тим чи іншим відрізняються від моєї реалізації. Але не дивлячись на це, я знайшов та порівняв існуючі аналоги, які можуть бути схожі на мою реалізацію. Але зрозуміло, що мій застосунок буде набагато простіший ніж наведені приклади, але також матиме в собі відмінні особливості, які будуть відрізняти його від інших.

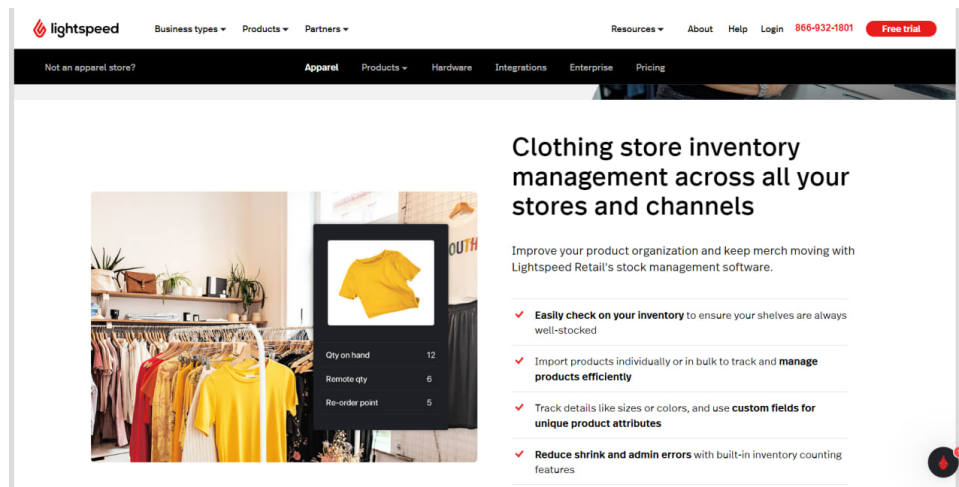
Retail Pro (2): це програмне забезпечення, яке дозволяє магазинам обліковувати продажі, управляти запасами, замовляти товари, проводити аналіз продажів та вести клієнтську базу даних, роботу даного сайту можна побачити на (мал. 1.1).



Мал. 1.1 “Дизайн сайту Retail Pro”

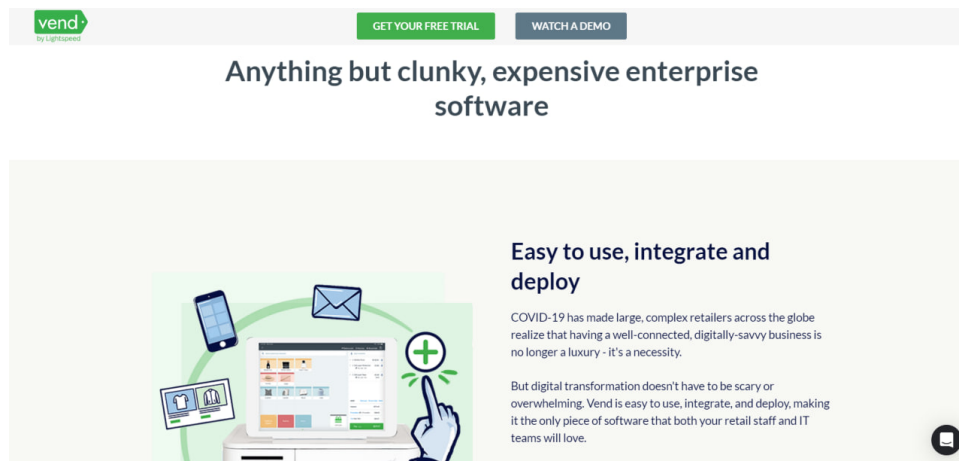
Lightspeed (3): це веб-програмне забезпечення, яке надає можливість магазинам вести електронний каталог товарів, керувати запасами, проводити операції з продажу та повернення товарів, аналізувати продажі

та управляти клієнтськими даними, роботу даного сайту можна побачити на (мал. 1.2).



Мал 1.2 “Дизайн сайту Lightspeed”

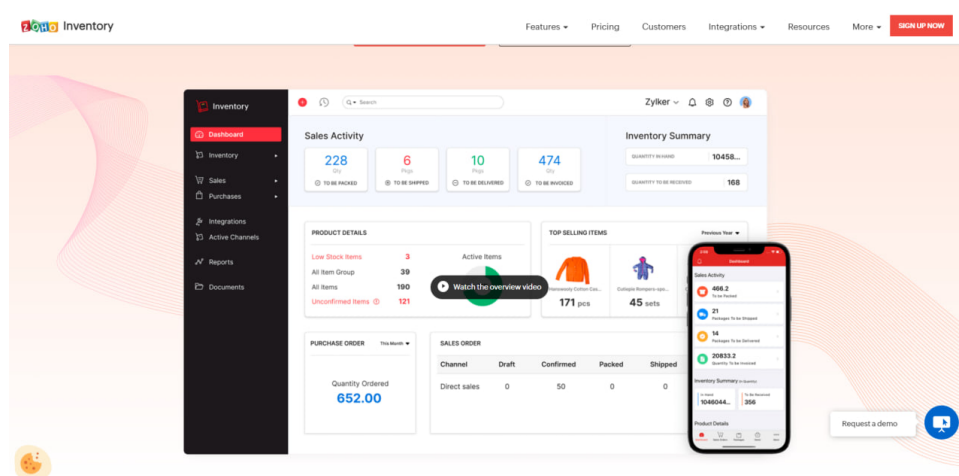
Vend (4): це хмарне програмне забезпечення, яке дозволяє магазинам керувати запасами, проводити операції з продажу та повернення товарів, вести клієнтську базу даних та аналізувати продажі, роботу даного сайту можна побачити на (мал. 1.3).



Мал. 1.3 “Дизайн сайту Vend”

Zoho Inventory (5): це програмне забезпечення для управління запасами, яке надає можливість магазинам керувати запасами, вести журнали

продажів та замовлення, проводити аналіз продажів та керувати поверненнями товарів роботу даного сайту можна побачити на (мал. 1.4).



Мал. 1.4 “Дизайн сайту Zoho Inventory”

Висновок:

У першому розділі курсової роботи торкнувся і обґрунтував такі підтеми як: вибір мною теми курсової, чому саме дана тема мене зацікавила, описав переваги програмного застосунку перед існуючими магазинами обліку одягу, також коротко поставив для себе основні завдання, які обов’язково мають бути виконані у процесі виконання. Провів опис предметної області мого застосунку, приблизно навів функції та можливості, які будуть міститися у реалізації. Проаналізував та вибрав математичні методи, обрав технології програмування, та коротко навів приклади їх роботи. Також порівняв мій застосунок з існуючими аналогами, для оцінки оригінальності мого застосунку.

РОЗДІЛ 2

“Проектування архітектури та реалізація програмного застосунку”

Розділ 2.1

Опис програмних інтерфейсів

В моїй курсовій роботі, яка має тему “Програмний застосунок магазину обліку речей” можна виділити такі інтерфейси, класи, та функції:

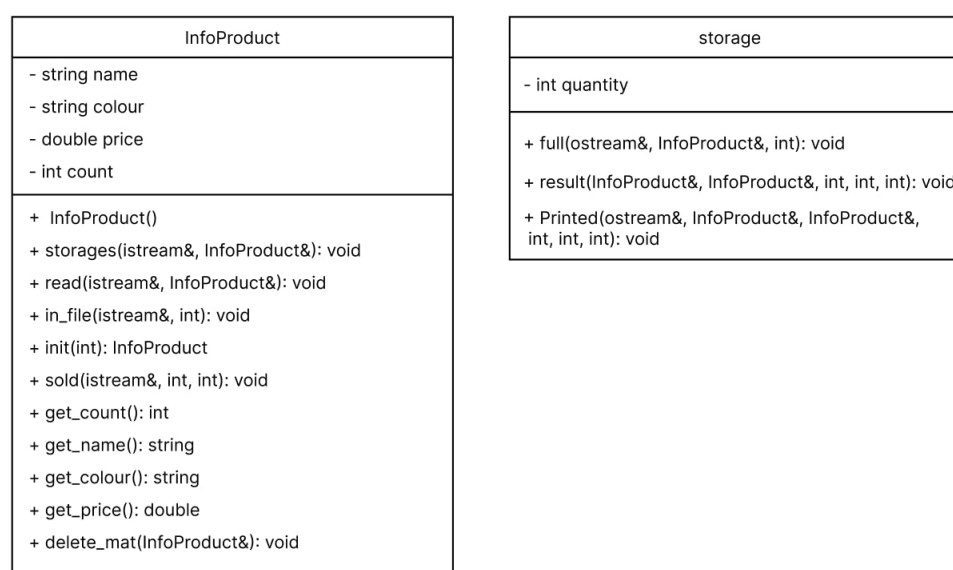
- Програмний інтерфейс SFML, його вбудовані класи, методи файли, та бібліотека <SFML/Graphics.hpp>, на якій і буде будуватись 100% графічного інтерфейсу застосунку.
- Клас InfoProduct, який містить в собі інформацію про наявний товар на складі, яку він бере з файлу (накладної)
- Клас storage, який містить в собі інформацію про поточну кількість всього товару на складі.
- Клас InfoClient, в якому зберігається та оброблюється інформація про клієнта, його ім'я, номер телефону та інше.
- Клас Order, який після купівлі товару користувачем друкує чек з інформацією про покупку.
- Клас ARMANI, який наслідує усі наявні класи, описані вище.

Класи і методи, за допомогою яких реалізується графічний інтерфейс:

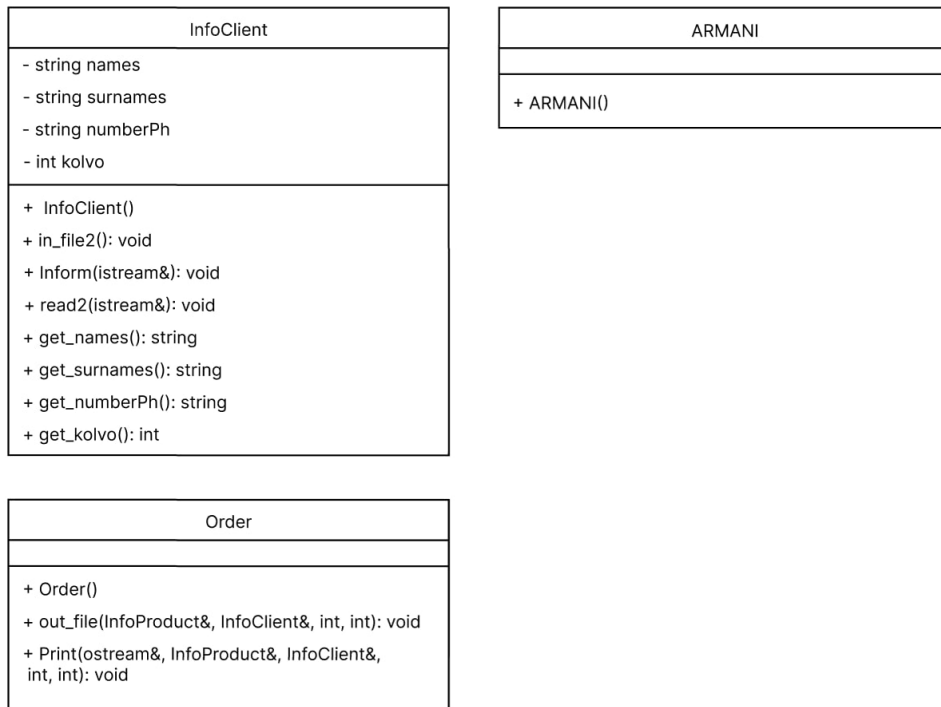
- Клас Button, який допомагає створенню кнопок, та полегшує їх адаптацію в графічний інтерфейс.
- Клас SpriteObject, клас, за допомогою якого є можливість додавати користувацькі зображення до інтерфейсу.

- Клас TextObject допомагає урізноманітнити тексти, які містяться на сторінках графічних вікон.
- Клас Rectangle, по суті, біль проста версія класу Button, який допомагає створювати прямокутники, та підлаштовувати їх під потреби застосунку.
- Функції, які допомагають перетворити значення типу int та double в string, для їх друку в графічному вікні.
- Функція Home_Page, метод, в якому графічно реалізовано головну сторінку магазину
- Функція Catalogue, одна з основних функції, в якому користувач має можливість вибрати товар, який його зацікавив.
- Функція Buy, третє графічне вікно, яке виводить інформацію про покупку користувача, враховуючи кількість товару та інше.
- Функція Final, останнє вікно, яке дякує користувачу за купівлю товару, та дає змогу дізнатись деяку інформацію про товар на складі.

UML - діаграми класів (Мал 2.1, 2.2) (Додаток Б):

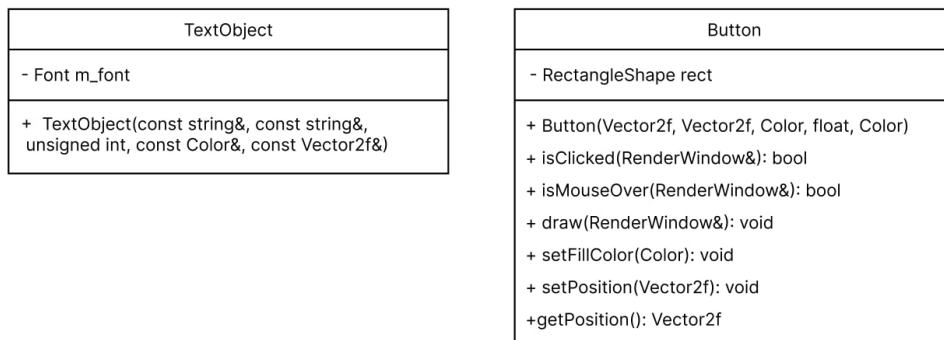


Мал 2.1



Мал 2.2

UML - діаграми класів графічного застосунку (Мал 2.3, 2.4:



Мал 2.3

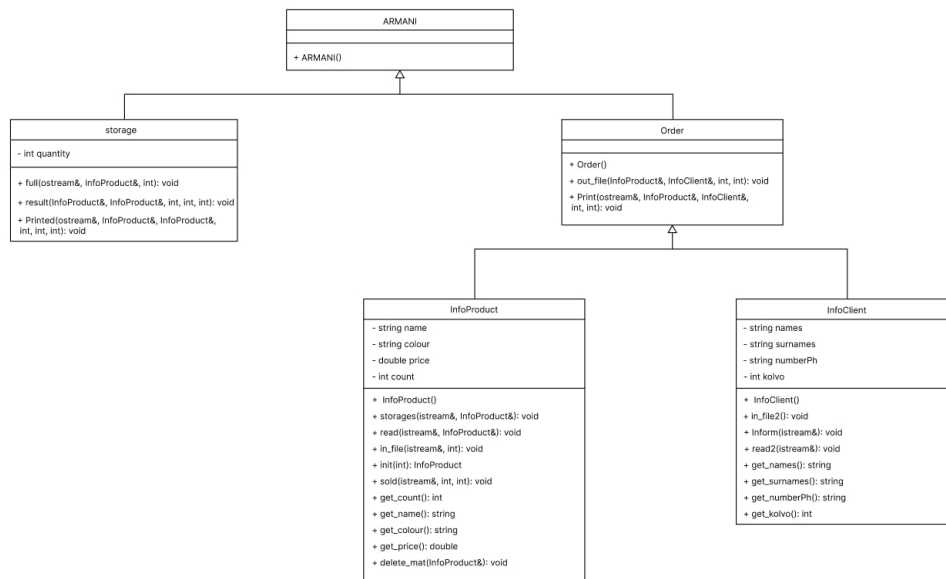
SpriteObject
+ Sprite sprite
+ SpriteObject(Vector2f, Texture&, Vector2f, Color) + getSprite(): Sprite&

Rectangle
- RectangleShape rect
+ Rectangle(Vector2f, Vector2f, Color) + draw(RenderWindow&): void + setOutlineColor(Color): void + setOutlineThickness(float): void

Мал 2.4

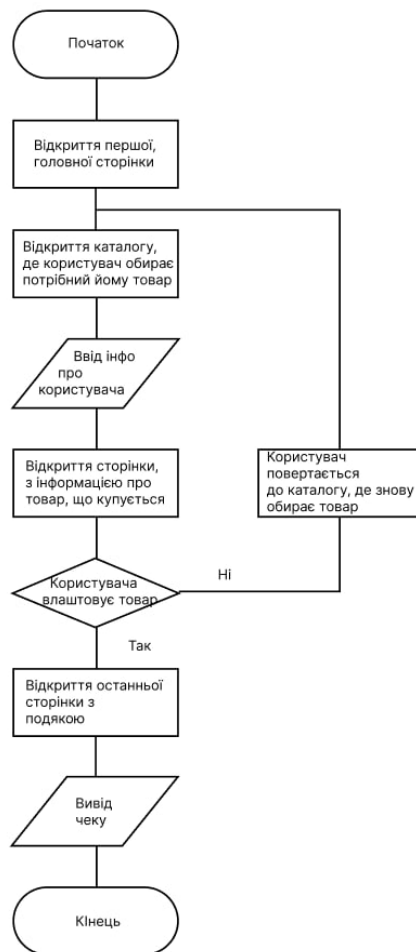
Розділ 2.2

Опис ієрархічних залежностей (Мал 2.5) (Додаток В)



Мал 2.5

Головна блок - схема застосунку (Мал 2.6) (Додаток А):



Мал 2.6

Розділ 2.3

Опис основних модулів програмної системи

Клас InfoProduct

Даний клас містить в собі інформацію, про кожен товар який міститься на складі. Задля оперування цими даними, були створені змінні, які зберігають цю інформацію в собі. Цей клас містить в собі різні методи, такі як зчитування даних з файлу-накладної, та подальша їх обробка, метод створення масиву даних, та видалення пам'яті, яка була виділена під

масив, метод оновлення кількості речей на складі, після продажу, та різноманітні геттери та сеттери.

Клас storage

Клас-склад, який оперує методами створення підсумкового чеку, в якому містяться такі важливі дані, як прибуток магазину, кількість проданих товарів, та інше.

Клас InfoClient

Клас, який має в собі поля-методи, які зберігають інформацію про клієнта, який купує товар. Із методів, що наявні в даному класі можна виділити методи зчитування та обробки даних про клієнта, та різноманітні сеттери та геттери.

Клас Order

Клас, який наслідує інформацію з інших класів, і завдяки цьому створює чек купівлі, де міститься інформація про клієнта, та про те, що він придбав.

Клас ARMANI

Збірний клас, який наслідує абсолютно всі інші класи, для більш легкого оперування ними.

Клас TextObject

Графічний клас, який допомагає у створенні, зміні, та адаптації тексту на графічному вікні, а також змінну, яка створює та змінює шрифти тексту.

Клас Button

Також графічний клас, для додавання кнопок на сторінку. Із методів, наявних у ньому, це метод реагування на клік мишкою в області кнопки, а

також наведення курсору на кнопку. І різноманітні функції, які кастомізують та змінюють зовнішній вигляд кнопок.

Клас SpriteObject

Третій клас, завдяки якому, є можливість додавати користувацькі картинки та малюнки в графічне вікно, для різнобарвлення контенту, що там міститься.

Клас Rectangle

Даний клас схожий на всі інші графічні класи, бо він допомагає у створенні прямокутників різних розмірів та кольорів і додаванню їх у вікно. Із окремих методів класа тут є створення граней прямокутника, та надання йому кольорів.

Тепер, більш розширена та конкретна інформація щодо кожного класу та кожних методів, що містяться в цих класах можна продивитись у Таблиця 3.1 “Опис класів, їх методів та функцій.”

Таблиця 3.1 “Опис класів, їх методів і функцій” (Додаток Г)

№ п/п	Назва класу	Назва методу	Призначення методу	Вхідні параметри	Вихідні параметри	Заголовний файл
	InfoProduct	storages	Зчитування даних з файлу	istream& in, InfoProduct & infos	void	program .h
	InfoProduct	read	Обробка даних, що зчитались з файлу.	istream& in, InfoProduct & info	void	program .h
	InfoProduct	in_file	Надання конкретних даних для	InfoProduct *& i, int size	void	program .h

			зчитування, запис даних з файлу до масиву			
	InfoProduct	init	Створення масиву для запису даних	int size	Масив даних	program .h
	InfoProduct	sold	Оновлення даних про кількість товару на складі після продажу	InfoProduct *& i, int number, int kolvo	void	program .h
	InfoProduct	get_count	Повертає кількість конкретного товару	-	Кількість конкретног о товару	program .h
	InfoProduct	get_name	Повертає назву конкретного товару	-	Назва конкретног о товару	program .h
	InfoProduct	get_colour	Повертає колір конкретного товару	-	Колір конкретног о товару	program .h
	InfoProduct	get_price	Повертає ціну конкретного товару	-	Ціну конкретног о товару	program .h
	InfoProduct	delete_mat	Видаляє масив	InfoProduct *& i	void	program .h
	storage	result	Надання конкретних даних для запису	InfoProduct *& i, InfoProduct & d, int number, int kolvo, int size	void	program .h

	storage	Printed	Запис даних до файлу	ostream& out, InfoProduct *& i, InfoProduct & d, int number, int kolvo, int size	void	program.h
	InfoClient	in_file2	Надання конкретних даних щодо зчитування з файлу	-	void	program.h
	InfoClient	Inform	Зчитування з файлу	istream& in	void	program.h
	InfoClient	read2	Обробка даних після зчитування	istream& in	void	program.h
	InfoClient	get_names	Повертає ім'я користувача	-	Ім'я користувача	program.h
	InfoClient	get_surnames	Повертає прізвище користувача	-	Прізвище користувача	program.h
	InfoClient	get_numberPh	Повертає номер мобільного телефону користувача	-	Номер телефону користувача	program.h
	InfoClient	get_kolvo	Повертає кількість товару, яку хоче придбати користувач	-	Кількість товару, яку хоче придбати користувач	program.h
	Order	out_file	Надання конкретних	InfoProduct *& i,	void	program.h

			даних щодо запису до файлу	InfoClient& lol, int number, int kolvo		
	Order	Print	Запис даних до файлу	ostream& out, InfoProduct *& i, InfoClient& lol, int number, int kolvo	void	program.h
	Button	isClicked	Зчитування кліку мишею на границі кнопки, або в них	RenderWindow& window	Повертає true, якщо клік мишкою відбувся, false, якщо ні	graphic.h
	Button	isMouseOver	Зчитування положення курсору, коли він потрапляє на кнопку	RenderWindow& window	Повертає true, якщо курсор знаходиться в границі кнопки, false, якщо ні	graphic.h
	Button	draw	Вивід кнопки у вікно	RenderWindow& window	void	graphic.h
	Button	setFillColor	Зміна кольору кнопки	Color color	void	graphic.h
	Rectangle	draw	Вивід прямокутника у вікно	RenderWindow& window	void	graphic.h
	Rectangle	setOutlineColor	Надання кольору обрамленню	Color outlineColor	void	graphic.h

	Rectangle	setOutline Thickness	Надання прямокутнику обрамлення	float thickness	void	graphic. h
--	-----------	-------------------------	---------------------------------------	--------------------	------	---------------

Таблиця 3.2 “Опис функцій” (Додаток Д)

	-	Home_Page	Створення першого графічного окна, та вивід всіх графічних об’єктів в це вікно.	ARMANI&a, InfoProduct *& i, int size	void	graphic. h
	-	Catalogue	Друге графічне вікно з вибором товару, та з виводом всіх графічних об’єктів в це вікно.	ARMANI&a, InfoProduct *& i, int size	void	graphic. h
	-	Buy	Третє графічне вікно, з підтвердженн ям купівлі товару користувачем.	ARMANI&a, InfoProduct *& i, int number, int size	void	graphic. h
	-	Final	Четверте графічне вікно з подякою користувачу за купівлю, та з деякою іншою інформацією	ARMANI&a, InfoProduct *& i, int number, int size	void	graphic. h
	-	string_coun t	Перетворення кількості (тип int) в тип string.	InfoProduct *& i, int index	Повертає модифіков аний текст	graphic. h

	-	string_price	Перетворення ціни (тип double) в тип string.	InfoProduct *& i, int index	Повертає модифікований текст	graphic.h
	-	string_fullprice	Перетворення повної ціни (тип double) в тип string.	InfoClient& a, InfoProduct *& i, int index	Повертає модифікований текст	graphic.h
	-	string_kolvo	Перетворення кількості (тип int) в string.	InfoClient& s	Повертає модифікований текст	graphic.h

Розділ 2.4

Опис програмних рішень та інтерфейсу користувача

На самому початку користувач має скомпілювати програму в одному з можливих компіляторів, які підтримують мову програмування C++, та встановлену до цієї мови бібліотеку SFML. Після компіляції користувач побачить перед собою Головну сторінку нашого застосунку, яке у коді реалізовано за допомогою різноманітних класів і зібрано в купу у функції Home_Page, дана сторінка має такий вигляд:



Мал 2.6 “Сторінка Home Page”

Даний клас у більшості містить суто дизайн сторінки, але при цьому він містить дві кнопки: перша кнопка “HOME PAGE”, зроблена для оновлення поточної сторінки, а друга кнопка “to collection” закриває дане графічне вікно, і відразу відкриває друге, що має назву “Catalogue”. Таким чином в нашому застосунку реалізовано перехід між сторінками.

Графічне вікно “Catalogue”

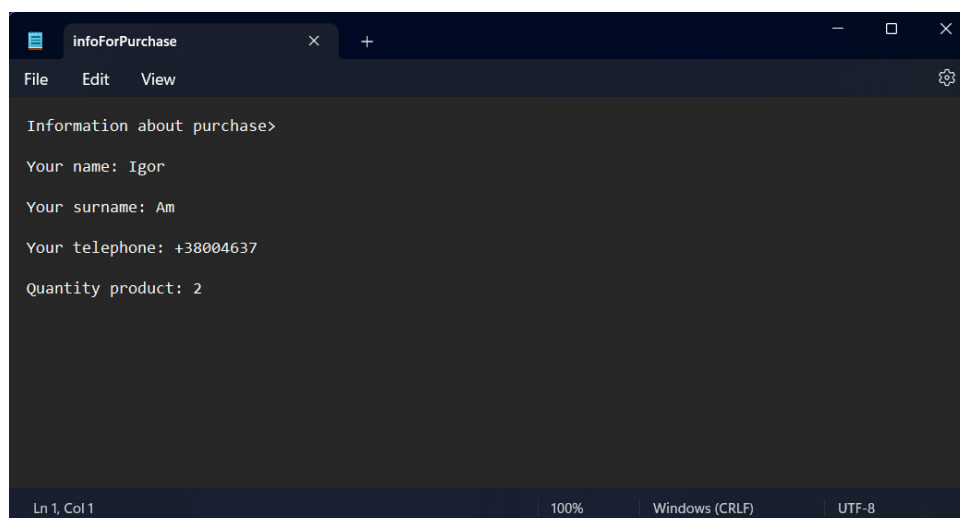
Після натискання кнопки відкривається друге графічне вікно з імпровізованим каталогом, де користувач бачить перед собою інформацію про товари, такі як: назва товару, його колір, його стокова ціна та кількість його на складі у даний момент. Всі ці дані зчитуються з файлу-накладної, яка робить логічним появу товару на складі, і при бажанні ці дані можуть бути змінені або замінені, і програма продовжить функціонувати без помилок. У графічному вікні користувач може бачити такі функціонуючі кнопки: Home Page, це кнопка, яка повертає нас на головну сторінку. Після її натискання закривається поточне вікно, і користувач знову потрапляє на першу сторінку. І інші 5 кнопок схожі одне на одну, бо кожна з них відповідає за купівлю свого, “окремого” товару. Кожна з цих кнопок передає подальшу інформацію про тип товару, який був обраний, і у

подальшому ця інформація використовується для друку чеку. Дизайн цього вікна притримується єдиного стилю всіх вікон, і схожий на попередню сторінку. Він має такий вигляд:



Мал 2.7 “Сторінка Catalogue”

Також, обов’язковою дією перед купівлею товару є введення користувачем даних про себе у окремий файл. Він має ввести своє ім’я, прізвище, свій номер телефону, та кількість товару, який він хоче придбати. Це необхідно для подальшої роботи програми.

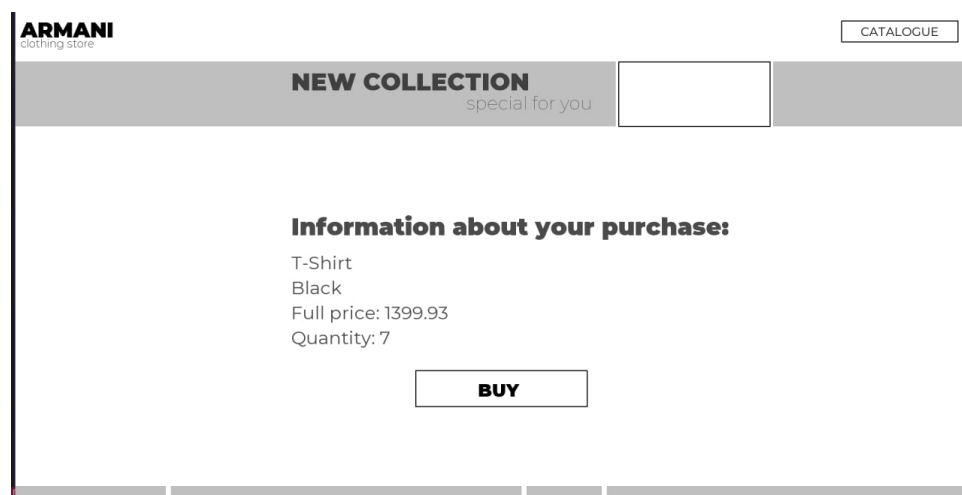


Мал 2.8 “Ввід інформації про користувача”

Після заповнення всіх даних покупець обирає товар, що зацікавив його, і має натиснути кнопку I WANT, і після цього знову ж таки поточне вікно закривається, і одразу після цього відкривається третє на черзі вікно, з назвою “Buy”

Графічне вікно “Buy”

Дане вікно є також не дуже великим по обсягу дій, які користувач може зробити, але воно виконує дуже важливу функцію - показує користувачу товар, який він обрав, підраховує повну ціну товарів, з урахуванням даних, які він ввів у файлі. Основна функція, яку має виконувати дане вікно є таке “підтвердження товару”, показати, що саме обрав покупець, і на яку ціну він має розраховувати при оплаті. Якщо ж покупець згоден і готовий купити цей товар, він має натиснути кнопку “Buy”. Якщо ж покупець не готовий купувати даний товар з різних причин, завжди можна натиснути кнопку “CATALOGUE”, яка поверне його на друге вікно, щоб він ще раз мав можливість для обрання товару. Дане вікно має такий дизайн:

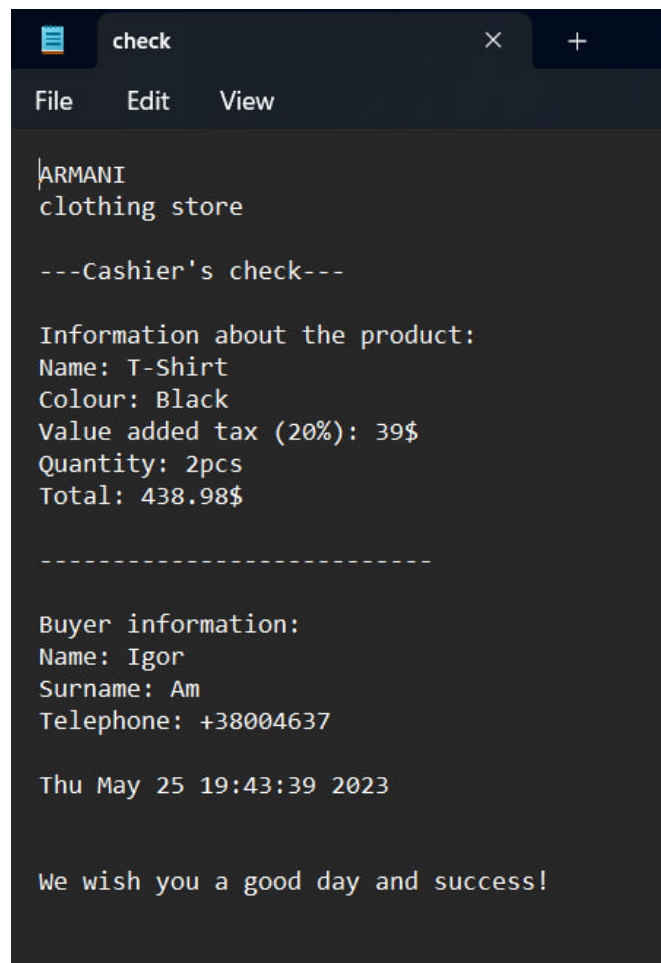


Мал 2.9 “Сторінка Buy”

Якщо кнопка “Buy” все ж таки була натиснута, з’являється четверте, останнє вікно “Final”.

Графічне вікно “Final”

Графічне вікно, в якому назва все говорить сама за себе. Після купівлі товару бренду “ARMANI”, магазин має подякувати покупцю за те, що він довіряє даному бренду, і готовий підтримувати його і далі. Мій магазин обліку не став виключенням. Перше, що бачить користувач у цьому вікні, слова подяки за покупку товару від цього бренду. Наступними словами йде інформація, про те, що чек інформацією про купівлю був надісланий йому txt-файлом. Файл містить приблизно таку інформацію:



```
check
File Edit View

|ARMANI
clothing store

---Cashier's check---

Information about the product:
Name: T-Shirt
Colour: Black
Value added tax (20%): 39$
Quantity: 2pcs
Total: 438.98$

-----

Buyer information:
Name: Igor
Surname: Am
Telephone: +38004637

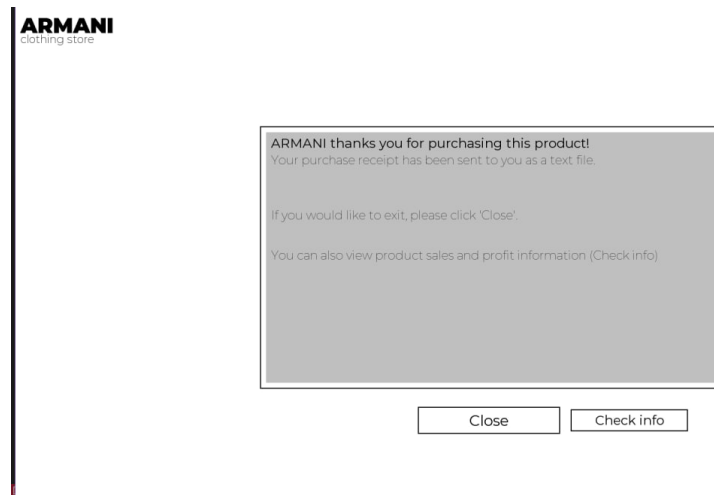
Thu May 25 19:43:39 2023

We wish you a good day and success!
```

Мал 2.10 “Фіксальний чек”

Наступне, що побачить користувач, є дві кнопки. Перша кнопка Close, яка, як не складно здогадатись, закриває дане вікно, і завершує виконання програми, і це можна вважати кінцем графічного інтерфейсу для даного застосунку. Друга ж кнопка “Check info” друкує чек для менеджера, в

якому міститься така інформація, як: Інформація про продаж товарів у даний день, в якому містяться інші купівлі, крім нашої, також там є підрахування повного прибутку товару за ці покупки, і оновлена кількість товарів на складі після продажу. Також, слід зауважити, що після натискання Check info, вікно не закривається, і користувач все одно має натиснути Close. Фінальний вигляд вікна такий:



Мал 2.11 “Сторінка Final”

Підсумовуючи цей невеличкий розділ, можна сказати, що програмний застосунок вийшов доволі непоганим і якісним, з цікавим та непримітивним дизайном та функціоналом. І звісно, ця програма не ідеальна, вона має деякі недоліки, але в цілому я вважаю цей застосунок працюючим, а головне простим у реалізації.

Висновок:

В даному розділі були описані покрово та детально всі складові коду мого застосунку, кожному класу, методу, функції, було дано визначення та опис,

за яким вже можна чітко зрозуміти, що виконує програма. Також, для більш детального опису класів розроблені UML-діаграми з демонструванням наслідування класів, та всіх наявних методів в ньому.

Покроково описаний графічний інтерфейс застосунку, всі можливості, що надає даний інтерфейс, всі графічні вікна, для більш легкого орієнтування в цих вікнах, та взагалі в всьому графічному інтерфейсу SFML

РОЗДІЛ 3

«Тестування та дослідження отриманих результатів».

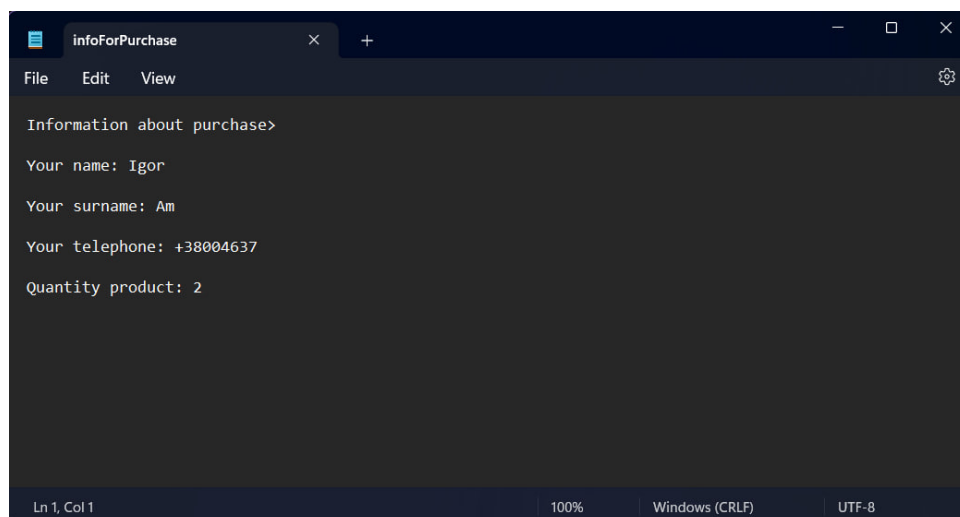
Дослідження програмної системи з точки зору її можливих переваг та недоліків, та перспектив.

Мій програмний застосунок обліку речей ARMANI, містить в собі деякі переваги перед аналогами. Першим, що можна виділити, це простота реалізації моєї програми. Якщо порівнювати інші програми для обліку речей, їх реалізація є дуже великою, складною, і тяжко зрозумілою. Так, звісно це пов'язано з більшою кількістю функцій, обчислень, обсягу клієнтів і товарообігом, але не дивлячись на все це, моя програма має всі базові найголовніші функції для магазину даного типу, і через це реалізація вийшла набагато простішою і зрозумілішою, тож я вважаю це найголовнішою перевагою перед аналогами. Другою перевагою я можу виділити час, що витрачає програма на обробку запитів користувача. Це означає, що, наприклад, при купівлі товару програма друкує чек за лічені мікросекунди, і покупець не повинен чекати, поки чек буде надісланий або роздрукований. Також, коли покупець вводить дані про себе, на зчитування цих даних програма витрачає просто нікчемно малий відрізок часу. Так, звісно, друк чеку та інформація про покупця є чисто імпровізованою частиною моєї програми у відмінності від інших застосунків, але якщо вдасться перенести швидкість роботи програми в реальну програму магазину, це буде суттєво відрізняти мою програму від всіх інших, і це буде головною перевагою мого магазину.

Але, звісно, кожна “ідеальна” програма або застосунок і мають свої недоліки які можна буде виправити у майбутньому. Мій магазин не став виключенням, тож він також їх має. Почати можна з першого недоліку. Суть цього недоліку полягає у тому, що користувач не може одразу

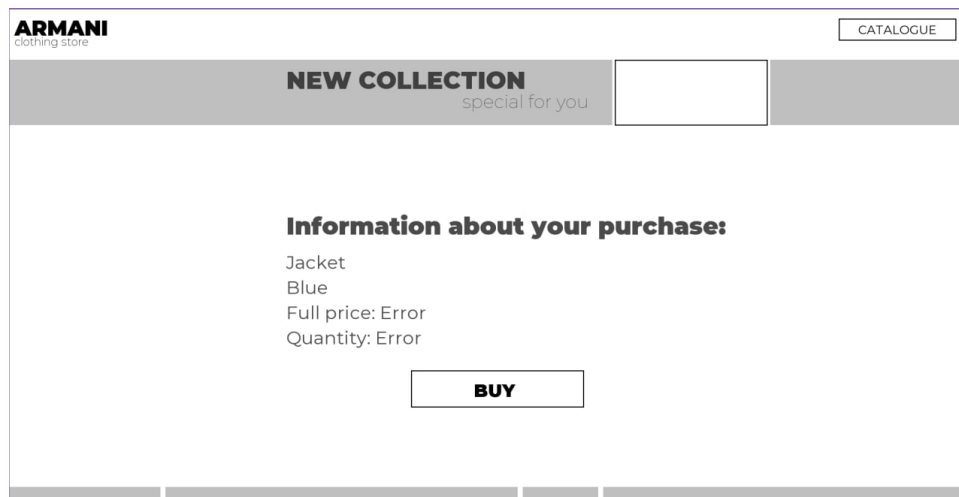
придбати два або більше товарів різного типу, це просто не було покладено в основу коду. Пов'язано це з тим, що реалізація цієї можливості зробила би код застосунку набагато складнішим і більшим, та і взагалі створення такої функції є дуже складним та енерговитратним процесом. Тому, жертвуючи можливістю придбати одразу декілька різних товарів, було прийнято рішення не додавати даний метод до мого застосунку. Але, не дивлячись на це, користувач може придбати якусь кількість товару, але цей товар буде одного й того ж типу.

Другим недоліком є те, що програму доволі легко вивести з ладу, ввівши невірні дані до текстового файлу. Наприклад, якщо введені дані будуть такі, як на Мал 3.1



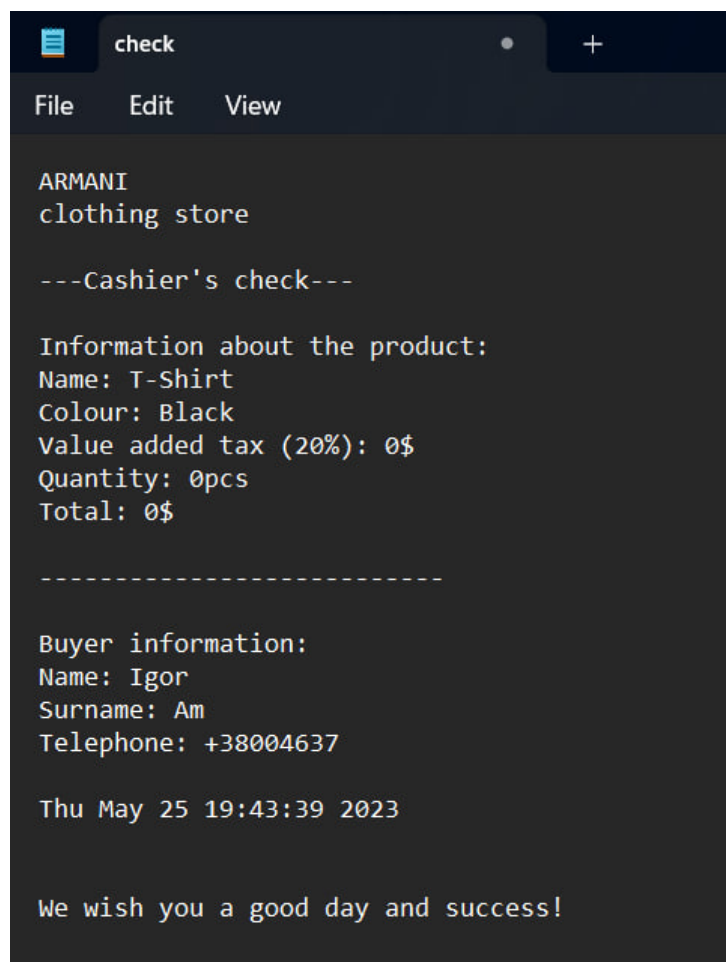
Мал 3.1 “Правильний ввід даних користувача”

то програма продовжить своє функціонування, і успішно “продасть товар” користувачу. Але, якщо ввести невірні дані у ці поля, програма буде давати збій, і якщо неправильне написання ім'я, прізвища, або номеру телефона дасть просто некоректне відображення в чеку, то неправильна ініціалізація кількості товару може призвести до такого:



Мал 3.2 “Вікно Buy, після некоректного вводу інфо”

Як видно, програма не змогла зчитати коректно кількість товару, і поля Full price і Quantity не були ініціалізовані. Але при цьому користувач може і надалі натиснути кнопку Buy, і надрукований чек матиме такий вигляд:



Мал 3.3 “Чек, після некоректного вводу інфо”

Але ці поля ініціалізовані нулем. Тобто, нічого не заплативши не отримуєш нічого, все цілком справедливо і правильно.

І третім недоліком цієї програми я вважаю неможливість прямої логістичної взаємодії між моїм магазином та реальним складом з речами ARMANI. Тобто, я не маю об’єктивної можливості дізнатися про поставки товарів зі складів, ні саму інформацію про даний товар, так як доступу до цих даних я не маю. І в обхід цьому недоліку, була створена імпровізована накладна товару, де я використовував приблизні дані про товари, їх вартість, опис, колір, тощо. По тій же самій причині, я не можу реалізувати поповнення складу речами, бо я не маю інформації про розташування складів ARMANI, і не можу розрахувати час, вартість доставок товару до складу. Натомість я реалізував поновлення кількостей товару на складі після кожного перезапуску мого застосунку. Тим самим, ми можемо думати, що після кожного перезапуску програми, кількість речей на складі поповнюється завдяки логистиці.

І, описавши всі ці недоліки, тепер можна описати перспективи по розвитку мого програмного застосунку, бо кожний застосунок, кожна програма потребує постійного поліпшення і доведення її до “ідеалу”.

Першим перспективним напрямком можна безумовно вважати доопрацювання всіх наявних недоліків на даному етапі розробки програми. В цей список можна внести і розширення кількості товарів, наявних на складі, можливість придбання товарів різних типів, і використання реальної логістики для поповнення складів магазину, доставка товарів по запропонованому користувачем адресу, і багато чого іншого.

Другою перспективою свого магазину я бачу пряму співпрацю з брендом ARMANI. Така співпраця дасть дуже багато позитивних наслідків для магазину. Це й ліцензійне використання запатентованих ARMANI назв, логотипів, облич, пряма співпраця з швейними фабриками, яка дозволить легше покривати дефіцити товарів на складах, продаж особливих лімітованих серій товарів, а можливо і взагалі спільні колаборації з даним брендом. І найголовнішим плюсом - довіра покупців. Людина буде повністю упевнена в оригінальності та надійності даного магазину, бо як ми знаємо по собі, ми звикли довіряти лише перевіреним джерелам, в яких впевнені, ось і мій магазин не стане виключенням.

Тож, описавши всі ці переваги, недоліки, та перспективи, можна підбити підсумок, і сказати що моїй програмі ще є куди рости і розвиватись, багато чого було вже зроблено для роботи застосунку, і ще більше потрібно зробити, бо кожна програма, як і людина, прагне стати кращою версією самої себе. Але, навіть вже на даному етапі я задоволений своєю програмою, і вважаю що вона має місце у світі програмних застосунків магазинів.

Висновок:

В даному розділі детально описав всі переваги мого застосунку перед іншими програмами, також навів тести, виконання яких дозволяє пересвідчитись у правильності роботи програми. При цьому, описав недоліки, які також наявні у програмі, описав набори відповідних тестових даних та очікувані результати, деякі з яких виводили програму з ладу. І третім описав можливі перспективні напрямки розвитку мого магазину, які можна і потрібно реалізувати в майбутньому.

ВИСНОВОК ДО КУРСОВОЇ РОБОТИ

Метою даної курсової роботи особисто для мене було створення програмного застосунку магазину обліку речей ARMANI; опис всіх використаних методів, програм, різних технологій програмування; дослідження застосунку на предмет похибок, несправностей, переваг та недоліків.

В першому розділі я чітко для себе поставив задачу та мету: який вигляд матиме мій застосунок, які дії він може робити, за яким алгоритмом буде працювати, та як буде виглядати його графічний інтерфейс. Також, в даному розділі описані всі методи та технології, які я використовував для створення програми, детально провів аналіз предметної області застосунку, порівняв мій застосунок з існуючими аналогами.

Другий розділ я повністю присвятив детальному та покроковому опису моєї програми, був описан кожен клас, метод, функція, їх призначення. Теж саме було зроблено для графічної частини, описав кожен елемент, з якого складається графіка. Також, для кожного класу створив окремі UML - діаграми, на яких позначив методи об'єктно-орієнтованого програмування, а саме наслідування класів. Створив головну блок - схему, на якій показав алгоритм, за яким працює мій застосунок. Також, окремої уваги приділив покроковому поясненню алгоритма, де описав, за що відповідає кожна кнопка, кожне вікно мого інтерфейсу.

В третьому розділі моя програма була описана зі сторони її можливих переваг перед іншими аналогами, недоліків, які виділяють програму на фоні інших, та поміркував на рахунок можливих перспективних напрямів розвитку даного застосунку.

Підсумовуючи дану курсову роботу, я абсолютно задоволений результатом своєї роботи, та фінальним результатом мого застосунку.

СПИСОК ДЖЕРЕЛ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ У КУРСОВІЙ РОБОТІ

Електронні книги:

1. Packt, [SFML Game Development], 9781849696845, 2013, 296 pages

Електронні ресурси:

2. [Retail Pro - сайт обліку товарів].

URL: <https://www.retailpro.com/include/eng/Solutions/Chain-Store-POS/>

3. [Lightspeed - веб-програмне забезпечення].

URL: <https://www.lightspeedhq.com/pc/retail/promotion-retail/>

4. [Vend - хмарне програмне забезпечення].

URL: <https://www.vendhq.com/multi-store-pos>

5. [Zoho Inventory - програмне забезпечення для управління запасами].

URL: <https://www.zoho.com/commerce/?src=top-header&ireft=inventory>

6. [Як вести облік у магазині?]

URL: <https://onix-soft.com.ua/info-centr/kak-vesti-uchet-v-magazine/>

7. [Що таке зворотна логістика і чому вона важлива?]

URL: <https://ub1.com.ua/shho-take-zvorotna-logistyka-i-chomu-vona-vazhlyva/>

8. [Посібник з графічного застосунку SFML].

URL: <https://www.sfml-dev.org/tutorials/2.5/>

ДОДАТКИ

Додаток “А”

Головна блок - схема алгоритму



Додаток “Б”

UML - діаграми класів

InfoProduct
<ul style="list-style-type: none">- string name- string colour- double price- int count
<ul style="list-style-type: none">+ InfoProduct()+ storages(istream&, InfoProduct&): void+ read(istream&, InfoProduct&): void+ in_file(istream&, int): void+ init(int): InfoProduct+ sold(istream&, int, int): void+ get_count(): int+ get_name(): string+ get_colour(): string+ get_price(): double+ delete_mat(InfoProduct&): void

storage
<ul style="list-style-type: none">- int quantity
<ul style="list-style-type: none">+ full(ostream&, InfoProduct&, int): void+ result(InfoProduct&, InfoProduct&, int, int, int): void+ Printed(ostream&, InfoProduct&, InfoProduct&, int, int, int): void

InfoClient
<ul style="list-style-type: none">- string names- string surnames- string numberPh- int kolvo
<ul style="list-style-type: none">+ InfoClient()+ in_file2(): void+ Inform(istream&): void+ read2(istream&): void+ get_names(): string+ get_surnames(): string+ get_numberPh(): string+ get_kolvo(): int

ARMANI
<ul style="list-style-type: none">+ ARMANI()

Order
<ul style="list-style-type: none">+ Order()+ out_file(InfoProduct&, InfoClient&, int, int): void+ Print(ostream&, InfoProduct&, InfoClient&, int, int): void

TextObject
- Font m_font
+ TextObject(const string&, const string&, unsigned int, const Color&, const Vector2f&)

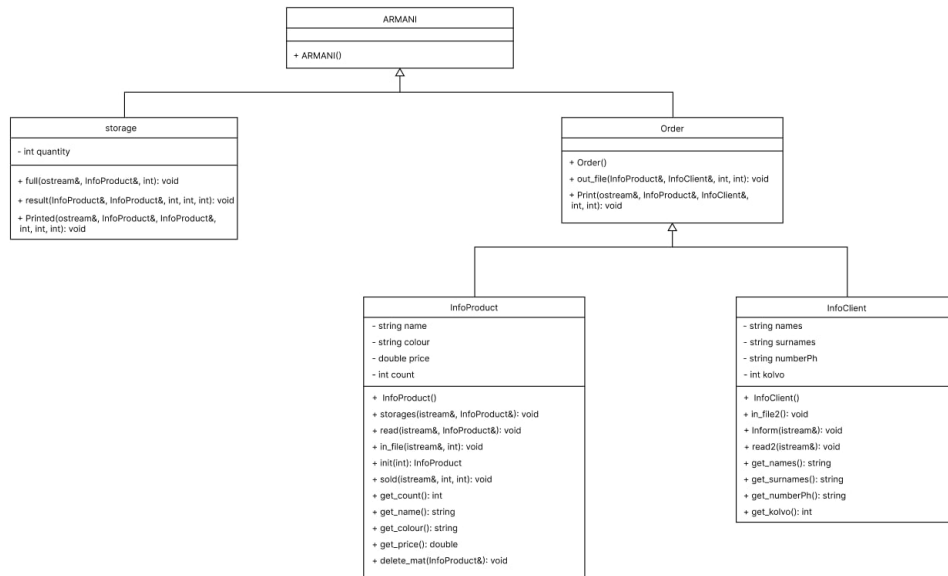
Button
- RectangleShape rect
+ Button(Vector2f, Vector2f, Color, float, Color)
+ isClicked(RenderWindow&): bool
+ isMouseOver(RenderWindow&): bool
+ draw(RenderWindow&): void
+ setFillColor(Color): void
+ setPosition(Vector2f): void
+ getPosition(): Vector2f

SpriteObject
+ Sprite sprite
+ SpriteObject(Vector2f, Texture&, Vector2f, Color)
+ getSprite(): Sprite&

Rectangle
- RectangleShape rect
+ Rectangle(Vector2f, Vector2f, Color)
+ draw(RenderWindow&): void
+ setOutlineColor(Color): void
+ setOutlineThickness(float): void

Додаток “В”

Опис ієрархічних залежностей



Додаток “Г”

Опис класів, їх методів і функцій

№ п/п	Назва класу	Назва методу	Призначення методу	Вхідні параметри	Вихідні параметри	Заголовний файл
	InfoProduct	storages	Зчитування даних з файлу	istream& in, InfoProduct & infos	void	program.h
	InfoProduct	read	Обробка даних, що зчитались з файлу.	istream& in, InfoProduct & info	void	program.h
	InfoProduct	in_file	Надання конкретних даних для зчитування, запис даних з файлу до масиву	InfoProduct *& i, int size	void	program.h
	InfoProduct	init	Створення масиву для запису даних	int size	Масив даних	program.h
	InfoProduct	sold	Оновлення даних про кількість товару на складі після продажу	InfoProduct *& i, int number, int kolvo	void	program.h
	InfoProduct	get_count	Повертає кількість конкретного товару	-	Кількість конкретного товару	program.h
	InfoProduct	get_name	Повертає назву конкретного товару	-	Назва конкретного товару	program.h

	InfoProduct	get_colour	Повертає колір конкретного товару	-	Колір конкретного товару	program.h
	InfoProduct	get_price	Повертає ціну конкретного товару	-	Ціну конкретного товару	program.h
	InfoProduct	delete_mat	Видаляє масив	InfoProduct *& i	void	program.h
	storage	result	Надання конкретних даних для запису	InfoProduct *& i, InfoProduct & d, int number, int kolvo, int size	void	program.h
	storage	Printed	Запис даних до файлу	ostream& out, InfoProduct *& i, InfoProduct & d, int number, int kolvo, int size	void	program.h
	InfoClient	in_file2	Надання конкретних даних щодо зчитування з файлу	-	void	program.h
	InfoClient	Inform	Зчитування з файлу	istream& in	void	program.h
	InfoClient	read2	Обробка даних після зчитування	istream& in	void	program.h

	InfoClient	get_names	Повертає ім'я користувача	-	Ім'я користувача	program.h
	InfoClient	get_surnames	Повертає прізвище користувача	-	Прізвище користувача	program.h
	InfoClient	get_numberPh	Повертає номер мобільного телефону користувача	-	Номер телефону користувача	program.h
	InfoClient	get_kolvo	Повертає кількість товару, яку хоче придбати користувач	-	Кількість товару, яку хоче придбати користувач	program.h
	Order	out_file	Надання конкретних даних щодо запису до файлу	InfoProduct *& i, InfoClient& lol, int number, int kolvo	void	program.h
	Order	Print	Запис даних до файлу	ostream& out, InfoProduct *& i, InfoClient& lol, int number, int kolvo	void	program.h
	Button	isClicked	Зчитування кліку мишею на границі кнопки, або в них	RenderWindow& window	Повертає true, якщо клік мишкою відбувся, false, якщо ні	graphic.h
	Button	isMouseOver	Зчитування положення курсору, коли	RenderWindow& window	Повертає true, якщо курсор	graphic.h

			він потрапляє на кнопку		знаходиться в границі кнопки, false, якщо ні	
	Button	draw	Вивід кнопки у вікно	RenderWindow & window	void	graphic.h
	Button	setFillColor	Зміна кольору кнопки	Color color	void	graphic.h
	Rectangle	draw	Вивід прямокутника у вікно	RenderWindow & window	void	graphic.h
	Rectangle	setOutlineColor	Надання кольору обрамленню	Color outlineColor	void	graphic.h
	Rectangle	setOutlineThickness	Надання прямокутнику обрамлення	float thickness	void	graphic.h

Додаток “Д”

Опис функцій

	-	Home_Page	Створення першого графічного окна, та вивід всіх графічних об'єктів в це вікно.	ARMANI&a, InfoProduct *& i, int size	void	graphic.h
	-	Catalogue	Друге графічне вікно з вибором товару, та з виводом всіх графічних об'єктів в це вікно.	ARMANI&a, InfoProduct *& i, int size	void	graphic.h
	-	Buy	Третє графічне вікно, з підтвердженням купівлі товару користувачем.	ARMANI&a, InfoProduct *& i, int number, int size	void	graphic.h
	-	Final	Четверте графічне вікно з подякою користувачу за купівлю, та з деякою іншою інформацією	ARMANI&a, InfoProduct *& i, int number, int size	void	graphic.h
	-	string_count	Перетворення кількості (тип int) в тип string.	InfoProduct *& i, int index	Повертає модифікований текст	graphic.h

	-	string_price	Перетворення ціни (тип double) в тип string.	InfoProduct *& i, int index	Повертає модифікований текст	graphic.h
	-	string_fullprice	Перетворення повної ціни (тип double) в тип string.	InfoClient&a, InfoProduct *& i, int index	Повертає модифікований текст	graphic.h
	-	string_kolvo	Перетворення кількості (тип int) в string.	InfoClient&s	Повертає модифікований текст	graphic.h

Додаток “Е”

Код програми

```
class InfoProduct
{
private:
    string name;
    string colour;
    double price;
    int count;
public:

    InfoProduct() {};

    void storages(istream& in, InfoProduct& infos)
    {
        string str;

        in >> str;

        in >> str >> str >> infos.name;

        in >> str >> str >> infos.colour;

        in >> str >> str >> infos.price;

        in >> str >> str >> infos.count;

    }

    void read(istream& in, InfoProduct& info)
    {
```

```

        storages(in, info);
    }

void in_file(InfoProduct*& i, int size)
{
    const char* filein = "invoice.txt";
    ifstream fin(filein);

    if (!fin)
    {
        cout << "Ой, возникла ошибка открытия файла...";
        return;
    }

    for (int j = 0; j < size; j++)
    {
        read(fin, i[j]);
    }
}

InfoProduct* init(int size)
{
    return new InfoProduct[size];
}

void sold(InfoProduct*& i, int number, int kolvo)
{
    i[number].count = i[number].count - kolvo;
}

int get_count()
{

```

```

        return count;
    }

    string get_name()
    {
        return name;
    }

    string get_colour()
    {
        return colour;
    }

    double get_price()
    {
        return price;
    }

    void delete_mat(InfoProduct*& i)
    {
        delete[] i;
        i = nullptr;
    }
};

class storage
{
public:
    storage() {}

    void result(InfoProduct*& i, InfoProduct& d, int number, int kolvo, int size)
    {
        const char* fileout = "result_for_day.txt";

```

```

ofstream fout(fileout);

if (!fout)
{
    cout << "Ой, возникла ошибка открытия файла...";
    return;
}

Printed(fout, i, d, number, kolvo, size);

fout.close();
}

void Printed(ostream& out, InfoProduct*& i, InfoProduct& d, int number, int kolvo, int size)
{
    out << "Final sales receipt in the ARMANI store: " << endl << endl;

    out << "-----" << endl << endl;

    time_t currentTime = time(nullptr);
    tm localTime;
    localtime_s(&localTime, &currentTime);
    int year = localTime.tm_year + 1900;
    int month = localTime.tm_mon + 1;
    int day = localTime.tm_mday;

    out << "Report for: " << day << "." << month << "." << year << endl << endl;

    out << "Product sold today:" << endl;

    out << i[number].get_name() << " in quantity " << kolvo << "pcs" << " for the total price " <<
    i[number].get_price() * kolvo << "$" << endl;

    srand(time(0));

```

```

int al = rand() % 6 + 3;
double fullmoney = 0;
for (int index = 0; index < al; index++)
{
    number = rand() % 6;
    kolvo = rand() % 5 + 1;

    if (i[number].get_count() < kolvo)
    {
        continue;
    }

    out << i[number].get_name() << " in quantity " << kolvo << "pcs" << " for the total price " <<
i[number].get_price() * kolvo << "$" << endl;
    fullmoney += i[number].get_price() * kolvo;
    d.sold(i, number, kolvo);
}

out << endl;
out << "Total profit of the store for today: " << fullmoney << "$" << endl << endl;

out << "-----" << endl << endl;

out << "Items in stock : " << endl;
for (int index = 0; index < size; index++)
{
    out << i[index].get_name() << " - " << i[index].get_count() << "pcs " << endl;
}

out << endl;
out << "-----" << endl << endl;

out << "ARMANI" << endl;

```

```

        out << "clothing store" << endl;

    }
};

class InfoClient
{
    string names;
    string surnames;
    string numberPh;
    int kolvo;
public:
    InfoClient() {}

    void in_file2()
    {
        const char* filein = "infoForPurchase.txt";
        ifstream fin(filein);

        if (!fin)
        {
            cout << "Ой, возникла ошибка открытия файла...";
            return;
        }

        read2(fin);

    }

    void Inform(istream& in)
    {
        string str;

```

```
in >> str >> str >> str;
```

```
in >> str >> str >> names;
```

```
in >> str >> str >> surnames;
```

```
in >> str >> str >> numberPh;
```

```
in >> str >> str >> kolvo;
```

```
}
```

```
void read2(istream& in)
```

```
{
```

```
    Inform(in);
```

```
}
```

```
string get_names()
```

```
{
```

```
    return names;
```

```
}
```

```
string get_surnames()
```

```
{
```

```
    return surnames;
```

```
}
```

```
string get_numberPh()
```

```
{
```

```
    return numberPh;
```

```
}
```

```

int get_kolvo()
{
    return kolvo;
}

};

class Order : public InfoProduct, public InfoClient
{
public:
    Order() {}

    void out_file(InfoProduct*& i, InfoClient& lol, int number, int kolvo)
    {
        const char* fileout = "check.txt";
        ofstream fout(fileout);

        if (!fout)
        {
            cout << "Ой, возникла ошибка открытия файла...";
            return;
        }

        Print(fout, i, lol, number, kolvo);

        fout.close();
    }

    void Print(ostream& out, InfoProduct*& i, InfoClient& lol, int number, int kolvo)
    {
        out << "ARMANI" << endl;
        out << "clothing store" << endl << endl;
    }
}

```



```

out << "---Cashier's check---" << endl << endl;

out << "Information about the product: " << endl;
out << "Name: " << i[number].get_name() << endl;
out << "Colour: " << i[number].get_colour() << endl;
int VAT = i[number].get_price() * 0.2;
out << "Value added tax (20%): " << VAT << "$" << endl;
out << "Quantity: " << kolvo << "pcs " << endl;
out << "Total: " << i[number].get_price() * kolvo + VAT << "$" << endl << endl;

out << "-----" << endl << endl;

out << "Buyer information:" << endl;
out << "Name: " << lol.get_names() << endl;
out << "Surname: " << lol.get_surnames() << endl;
out << "Telephone: " << lol.get_numberPh() << endl;

auto now = chrono::system_clock::now();
time_t t = chrono::system_clock::to_time_t(now);
char str[26];
ctime_s(str, sizeof str, &t);
out << endl;
out << str << endl;

out << endl;
out << "We wish you a good day and success!" << endl;
}

};

```