ЗМІСТ

С.

ВСТУП………………………………………………………………………..……4

1 ТЕХНІЧНІ ХАРАКТЕРИСТИКИ………………………………………….…..5

1.1 Постановка задачі……………………………………………………….….…5

1.1.1 Характеристика об’єкта автоматизації…………………………….……....5

1.1.2 Вимоги до програми………………………………………………………...5

1.2 Опис математичного методу рішення задачі………………………………..6

1.3 Визначення структури вхідних даних ………………………………………9

1.4 Визначення структури вихідних даних…………………………………….10

2 КОМП’ЮТЕРНА СИСТЕМА………………………………………………...12

2.1 Технічні характеристики комп’ютера та зовнішніх пристроїв ……….….12

2.2 Вибір програмних засобів та операційної системи……………………......12

3 ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ…………………….……..14

3.1 Інтерфейс програми…………………………………………………………14

3.2 Алгоритм рішення задачі…………………………………………………...22

4 ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ …………………………………...25

4.1 Розробка програми…………………………………………………………..25

4.2 Етапи налагодження………………………………………………………....25

4.3 Типи помилок………………………………………………………………..26

5 ОЧІКУВАНІ ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ…………………....27

5.1 Розрахунок вартості програмного продукту……………………………....27

5.2 Розрахунок економічної ефективності від впровадження програмного продукту………………………………………………………………………....36

5.3 Наукова організація праці на робочому місці програміста……………....42

6 ОХОРОНА ПРАЦІ…………………………………………………………....45

ВИСНОВКИ…………………………………………………..………………...57

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ………………………………………….. 58

Додаток А Схема алгоритму програми………………………………………. 59

Додаток Б Роздрук результатів роботи програми на

контрольному прикладі………………………………………………………. 60

Додаток В Значення коефіцієнта використання технологій

програмування та інструментів середовища розробки……………………. 64

Перелік інших частин:

1. Текст програми ДП.ХПФК.00452 –………………………………………. 65

2. Посібник програміста ДП.ХПФК.00452 – ………………………………..143

3. Посібник оператора ДП.ХПФК.00452– …………………………………...145

4. Специфікація ДП.ХПФК.00452 – …………………………………………147

# ВСТУП

Автоматизація є дуже важливою частиною в бізнесі, особливо в торговому. Також автоматизація процесів зменшить кількість помилок, спричинених людським фактором. Часто підприємці автоматизовують свій бізнес через збільшення даних та неможливість керування ними. У чому саме полягає автоматизація, це запровадження додатку в свій бізнес, до цього додатку прив’язується база даних, з якої будуть братись дані. Такий додаток буде застосовуватись в сфері торгівлі господарських товарів.

Область торгівлі продуктових товарів включає великий асортимент продукції. Саме тому автоматизація даної галузі є виправданою і доцільною. Тому що користувач може задовольнити більшість своїх потреб в одному магазині.

Метою дипломного проєкту є створення програмного продукту, який буде використовуватись касиром та адміністратором продуктового супермаркету.

Завдання дослідження − розробити адекватний меті дослідження інструментарій: додаток, робота якого буде спрямована на мінімізацію помилок і на пришвидшення роботи магазину та підвищення прибутковості з

максимальним скороченням працівників, що призведе до зменшення витрат на заробітні плати та підвищення прибутковості магазину.

Методи дослідження даного об’єкта. Моніторинг та аналіз інформаційних джерел об’єкта. Аналіз дає змогу розділяти предмети дослідження на складові частини. Моніторинг - метод постійного спостереження за станом окремих об'єктів, перебігом певних процесів в окремих системах. Даний метод є дуже продуктивним, він допомагає оцінити стан, у якому перебуває предметна область на даний час.

# ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

## Постановка задачі

### Характеристика об’єкта автоматизації

Продуктовий супермаркет - це комплексна торгівельна організація, яка спеціалізується на продажі різноманітних продуктів харчування, товарів для дому та особистої гігієни. Продуктові супермаркети мають розгалужену мережу підрозділів, які знаходяться в різних районах міста та регіонів.

Основні функції продуктового супермаркету включають продаж продуктів харчування, управління запасами та планування закупівель, керування продажами та клієнтською базою даних, ведення бухгалтерського обліку.

Для автоматизації діяльності продуктового супермаркету необхідне програмне забезпечення, яке дозволить ефективно управляти складським обліком, контролювати рух товарів, планувати закупівлі, керувати продажами та клієнтською базою даних, а також вести бухгалтерський облік. Важливим параметром є також маржинальність продуктів та обсяги продажів.

Для забезпечення ефективної автоматизації продуктового супермаркету, програмне забезпечення повинно бути інтегровано з програмами обліку та управління запасами, які вже використовуються в компанії. Це дозволить створити єдину інформаційну систему, яка забезпечить повну інтеграцію всіх підрозділів та ефективний контроль за управлінням компанією.

### 1.1.2 Вимоги до програми

Призначення програмного забезпечення та програмного продукту: розробка системи автоматизації управління продажами в продуктовому супермаркеті.

Перелік задач, які будуть вирішуватися в результаті використання розробленого програмного продукту:

* Облік товарів на складі;
* Керування закупівлями товарів;
* Ведення обліку продажів;
* Взаємодія з клієнтами та програма лояльності;
* Генерація звітів та аналіз даних.

Структура програмного продукту має наступний вигляд:

* База даних товарів та клієнтів;
* Модуль замовлення товарів;
* Модуль продажів та створення звітів;
* Модуль програми лояльності;
* Модуль аналізу даних.

Опис функціонування програмного продукту та його частин:

* Модуль замовлення товарів дозволяє закуповувати товари та зберігати інформацію про склад;
* Модуль продажів веде облік продажів та може генерувати звіти по продажам;
* Модуль програми лояльності дозволяє взаємодіяти з клієнтами та створювати пропозиції для їх мотивації до покупок;
* Модуль аналізу даних аналізує дані та генерує звіти для управління та оптимізації бізнес-процесів.

Користувачами програмного продукту можуть бути касир, що використовує програму на робочому комп’ютері за касою під час взаємодії з клієнтами. Також програма може використовуватись адміністратором супермаркету, який може виконувати всі вище згадані задачі.

## Опис математичного методу рішення задачі

При розробці поставленої задачі було використано метод об’єктно-орієнтованого програмування.

Об’єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину взаємодіючих об'єктів. У ній використано декілька технологій від попередніх парадигм, включаючи успадкування, модульність, поліморфізм та інкапсуляцію. Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-тих роках, одночасно з посиленням [дискусій](http://ua-referat.com/%D0%94%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%8B) про кризу програмного забезпечення. Незважаючи на те, що ця [парадигма](http://ua-referat.com/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0) з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих.

Об'єктно-орієнтовані програми можна розглядати як множину взаємодіючих об'єктів, на відміну від традиційних поглядів, коли програма розглядається як набір підпрограм, або як перелік інструкцій комп'ютеру. [Відповідно](http://ua-referat.com/%D0%92%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C) до парадигми ООП, кожен об'єкт здатен отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожний об'єкт можна розглядати як незалежний [автомат](http://ua-referat.com/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82) з окремим призначенням або відповідальністю.

Об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

* все є об'єктами;
* всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
* кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів;
* кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів;
* у класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії;
* класи організовані у єдину деревовидну структуру з загальним корінням, яка називається ієрархією [успадкування](http://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29). Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Об’єктно-орієнтоване програмування засноване на трьох принципах:

* інкапсуляції;
* наслідуванні;
* поліморфізмі.

Спостережуване в об'єктах об'єднання даних і операцій в одне ціле було позначене [терміном](http://ua-referat.com/%D0%A2%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B8) інкапсуляція. Застосування інкапсуляції зробило об'єкти схожими на маленькі програмні модулі і забезпечило приховування їх внутрішньої структури. Для об'єктів з'явилося [поняття](http://ua-referat.com/%D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D1%82%D1%8F) інтерфейсу, що значно підвищило їх надійність і цілісність.

Наслідування означає, що при створенні нового класу, немає необхідності в переписуванні заново всіх полів, методів і властивостей. Ви оголошуєте, що новий клас є нащадком (або дочірнім класом) наявного класу, так званим предком (або батьківським класом), і додаєте до нього нові поля, методи та властивості. Іншими словами, додається те, що потрібне для переходу від загального до приватного. [Процес](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81) створення нових класів на основі інших класів називається наслідуванням. Нові класи мають як успадковані ознаки, так і нові.

Поліморфізм означає, що в похідних класах можна змінювати роботу методів, що вже існують в базовому класі. При цьому весь програмний код, що керує об'єктами батьківського класу, придатний для управління об'єктами дочірнього класу без будь-якої модифікації.

Також у програмі використано ієрархію класів для створення динамічного графічного інтерфейсу та взаємодії даних.

## Визначення структури вхідних даних

Вхідними даними називають такі змінні (параметри, дані), які надаються виконавцеві до початку його роботи над алгоритмом.

Застосування алгоритму до неприпустимих(або неперевірених) вхідних даних може призводити до того, що алгоритм ніколи не зупиниться, або потрапить в тупиковий стан (зависання), з якого не зможе продовжитися виконання процесу.

Таблиця 1.1 – Вхідна інформація

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва | Форма подання | Термін і частота використання |
| 1 | Товар | Масив оперативної інформації | Не обмежено |
| 2 | Працівники | Масив нормативно-довідкової інформації | Не обмежено |
| 3 | Клієнт | Масив нормативно-довідкової інформації | Не обмежено |
| 4 | Товарна накладна | Масив нормативно-довідкової інформації | Не обмежено |

Вхідні дані зазвичай поділяють на два класи: довідкову та оперативну інформацію. Довідкові дані змінюються рідше, ніж оперативні, які можуть змінюватися навіть в режимі реального часу.

Таблиця 1.2 – Структура вхідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Pr\_Id | Integer | 0..2147483647 | Код товару |
| 2. | Pr\_BarCode | String | 0..20 | Штрих-код |
| 3. | Pr\_Description | String | 0..200 | Опис товару |
| 4. | Pr\_Brand | String | 0..30 | Виробник |
| 5. | Pr\_Title | String | 0..200 | Назва товару |
| 6. | Pr\_Category | String | 0..30 | Категорія |
| 7. | Pr\_Price | Decimal | 0..2147483647 | Ціна |
| 8. | Pr\_PriceOpt | Decimal | 0..2147483647 | Оптова ціна |
| 9. | Pr\_Qty | Integer | 0..2147483647 | Кількість |
| 10. | Emp\_Id | Integer | 0..2147483647 | Код працівника |
| 11. | Emp\_Name | String | 0..50 | Ім’я працівника |
| 12. | Emp\_Postion | String | 0..30 | Посада |
| 13. | Emp\_Salary | Decimal | 0..100000 | Зарплата |
| 14. | Sti\_Id | Integer | 0..2147483647 | Код замовлення |
| 15. | Sti\_Date | DateTime |  | Дата замовл. |
| 16. | Sti\_StockInBy | String | 0..30 | Замовник |
| Продовження таблиці 1.2 | | | | |
| 17. | Sti\_Status | String | 0..20 | Статус замовлення |
| 18. | Sti\_Supplier | String | 0..50 | Постачальник |
| 19. | Sti\_IsConfirmed | Bolean | true/false | Підтвердження замовлення |
| 20. | Cl\_Id | Integer | 0..2147483647 | Код клієнта |
| 21. | Cl\_Name | String | 0..30 | Ім’я клієнта |

## Визначення структури вихідних даних

Вихідною інформацією задачі для користувача будуть звіти та форми, які будуть відображатися на під час роботи з програмою як таблиці та форми. Також вихідна інформація може бути роздрукована . Вихідні дані – це дані, які виконавець видає як результат здійсненої роботи. Вихідними даними у програмному додатку є інформація про товари, чеки, клієнтів та працівників. В таблиці 1.3 показані вихідні дані, які використовуються в дипломному проєкті. В таблиці 1.4 показана структура вихідніх даних, які використовуються в даному дипломному проєкті.

Таблиця 1.3 - Вихідна інформація

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Форма | Звіт |
| 1 | Товари | + | + |
| 2 | Чек |  | + |
| 3 | Залишки на складі | + | + |
| Продовження таблиці 1.3 | | | |
| 4 | Клієнти |  | + |
| 5 | Транзакції покупок | + | + |
| 7 | Працівники | + | + |

Таблиця 1.4 – Структура вихідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Pr\_Id | Integer | 0..2147483647 | Код товару |
| 2. | Pr\_BarCode | String | 0..20 | Штрих-код |
| 3. | Pr\_Description | String | 0..200 | Опис товару |
| 4. | Pr\_Brand | String | 0..30 | Виробник |
| 5. | Pr\_Title | String | 0..200 | Назва товару |
| 6. | Pr\_Category | String | 0..30 | Категорія |
| 7. | Pr\_Price | Decimal | 0..2147483647 | Ціна |
| 8. | Pr\_Image | Binnary |  | Зображення товару |
| 8. | Pr\_PriceOpt | Decimal | 0..2147483647 | Оптова ціна |
| 9. | Pr\_Qty | Integer | 0..2147483647 | Кількість |
| 10. | Emp\_Id | Integer | 0..2147483647 | Код працівника |
| 11. | Emp\_Name | String | 0..50 | Ім’я працівника |
| 12. | Emp\_Postion | String | 0..30 | Посада |
| Продовження таблиці 1.4 | | | | |
| 13. | Emp\_Salary | Decimal | 0..100000 | Зарплата |
| 14. | Sti\_Id | Integer | 0..2147483647 | Код замовлення |
| 15. | Sti\_Date | DateTime |  | Дата замовлення |
| 16. | Sti\_StockInBy | String | 0..30 | Замовник |
| 17. | Sti\_Status | String | 0..20 | Статус замовлення |
| 18. | Sti\_Supplier | String | 0..50 | Постачальник |
| 19. | Sti\_IsConfirmed | Bolean | true/false | Підтвердження замовлення |
| 20. | Cl\_Id | Integer | 0..2147483647 | Код клієнта |
| 21. | Cl\_Name | String | 0..30 | Ім’я клієнта |

# КОМП’ЮТЕРНА СИСТЕМА

## Технічні характеристики комп’ютера та зовнішніх пристроїв

Для роботи розробленої прикладки використовувався персональний комп’ютер (ПК) з наступними характеристиками:

* Процесор AMD Ryzen 7 4800 H 2.9 GHz;
* оперативна пам’ять 16 ГБ DDR4 3200 MHz;
* відеокарта NVIDIA GTX 1650 Ti;
* монітор 1920x1080 60Hz;
* операційна система Windows 11 Pro;

Програма може функціонувати без втрати всіх функціональних можливостей на комп’ютері з такими технічними характеристиками:

- процесор з частотою 1ггц;

- RAM 128 MB ;

- відеокарта Intel HD Graphics;

- операційна система Windows, Mac, Linux, Chrome OS.

## Вибір програмних засобів та операційної системи

Для розробки веб-сайту було обрано такі засоби реалізації:

* мова програмування С# на платформі .NET;
* СУБД MSSQL Server;
* Microsoft RDLC дизайнер звітів.

C# є об'єктно-орієнтованою мовою програмування, що дозволяє розробникам створювати програми, які можуть бути легко розширені та підтримувані з часом. Мова підтримує багато функцій, які дозволяють розробникам писати якісний та ефективний код. C# використовується для створення різноманітних програм, включаючи веб-додатки, мобільні додатки, ігри та програми для роботи з базами даних. Мова є основою для платформи .NET, що дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах, таких як Windows, Linux та macOS.

Переваги мови програмування C#:

1. Об'єктно-орієнтований підхід: C# є повноцінною об'єктно-орієнтованою мовою програмування, що дозволяє розробникам створювати більш прості, ефективні та складні програми.
2. Безпека: C# має вбудований механізм безпеки, який дозволяє розробникам створювати більш безпечні програми, знижуючи ризик вразливостей та атак зловмисників.
3. Платформно-незалежний код: С# розроблена для роботи на платформі .NET, що дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах.
4. Бібліотеки та фреймворки: C# має велику кількість сторонніх бібліотек та фреймворків, що дозволяє розробникам створювати програми швидко та ефективно.

Недоліки мови програмування C#:

1. Залежність від платформи: C# була розроблена для платформи .NET, тому програми, написані на C#, можуть працювати тільки на операційній системі Windows або на платформі .NET.
2. Швидкість виконання: Порівняно з мовами програмування, такими як C та C++, виконання програм на C# може бути менш ефективним та повільним.
3. Низький рівень контролю над пам'яттю: Хоча C# має вбудований збірник сміття, що допомагає управляти пам'яттю, розробникам може бути складніше контролювати пам'ять.

MS SQL є однією з найпоширеніших реляційних баз даних, розроблених корпорацією Microsoft. Ця система управління базами даних дозволяє ефективно зберігати та організовувати великі об'єми даних. MS SQL забезпечує високу надійність, безпеку та продуктивність завдяки своїм функціям, таким як механізми резервного копіювання, відновлення даних та захист від вторгнень. Ця система має широкий спектр можливостей, включаючи розширені можливості звітування, інструменти аналізу даних та можливість створювати складні запити для отримання потрібної інформації.

Переваги MS SQL:

1. Надійність та стабільність: MS SQL має високий рівень надійності та стабільності завдяки своїм механізмам резервного копіювання та відновлення даних.
2. Висока продуктивність: MS SQL забезпечує високу продуктивність завдяки своїм оптимізаційним можливостям та підтримці паралельної обробки даних.
3. Розширені можливості звітування: MS SQL має вбудовані інструменти для створення різноманітних звітів та аналізу даних.
4. Інтеграція з продуктами Microsoft: MS SQL інтегрується з іншими продуктами Microsoft, що дозволяє ефективно працювати з даними та ділитися ними між різними системами.
5. Масштабованість: MS SQL може легко масштабуватися залежно від потреб користувачів та організацій.

Недоліки MS SQL:

1. Високі витрати: MS SQL є комерційним продуктом, що означає високі витрати на ліцензування та підтримку.
2. Складність: MS SQL може бути складним у використанні для новачків, оскільки має велику кількість функцій та можливостей.
3. Обмежена підтримка інших платформ: MS SQL більш спрямований на платформу Windows та менш підтримує інші операційні системи.
4. Невелика спільнота користувачів: порівняно з деякими іншими реляційними базами даних, MS SQL має меншу спільноту користувачів, що може ускладнити пошук допомоги та розв'язання проблем.

Microsoft RDLC (Report Definition Language Client-Side) Designer є інструментом для створення звітів у середовищі Visual Studio. Він дозволяє створювати різноманітні звіти, використовуючи різні джерела даних, такі як бази даних, веб-сервіси, Excel-файли тощо. RDLC Designer забезпечує зручний інтерфейс для налаштування форматування звітів, додавання тексту, зображень, графіків, таблиць, графіків та інших елементів. Після створення звіту, його можна зберегти в форматі RDLC або скомпілювати в DLL-файл, щоб використовувати його в програмному коді. RDLC Designer є частиною Microsoft SQL Server Reporting Services (SSRS) та може бути використаний для створення звітів у багатьох типах проектів, включаючи ASP.NET, WinForms, WPF, Silverlight тощо.

# КОМП’ЮТЕРНА СИСТЕМА

## Інтерфейс програми

Цінність будь-якої програми визначається не тільки її здатністю безпомилково і якісно вирішувати поставлену задачу, а й тим, наскільки легко і зручно цією програмою можна користуватися. Навіть найпростіша, і в той же час, функціональна програма виявиться незатребуваною, якщо користувач (тобто кінцевий споживач програми) при роботі з нею швидко втомлюється, дратується, відчуває себе некомфортно. Під призначеним для користувача інтерфейсом (англ. User interface) будемо розуміти своєрідний комунікаційний канал (сполучна ланка), за допомогою якого користувач взаємодіє з комп'ютером.

З точки зору користувача саме інтерфейс є кінцевим продуктом. Інтерфейс являє собою сукупність використовуваних в програмі засобів для введення даних, способів відображення інформації на екрані дисплея, елементів для управління різними етапами рішення задачі. Основне призначення (головна мета) призначеного для користувача інтерфейсу - забезпечити короткий і зручний шлях до отримання результату.

Інтерфейс користувача комп’ютерного додатку включає:

* Ясність;
* Виразність;
* Відповідність;
* Послідовність;
* Продуктивність;

Невдалі інтерфейси сильно уповільнюють роботу користувачів, сприяють швидкої стомлюваності, великому числу помилок і, як наслідок, відбивають у користувачів подальше бажання працювати з програмою. У зв'язку з цим розробнику програми слід враховувати відомі рекомендації провідних фахівців в області створення призначеного для користувача інтерфейсу.

Саме тому було розроблено максимально комфортний і зручний для користування інтерфейс (див. рис. 4.1 та рис. 4.2)

Рисунок 4.1

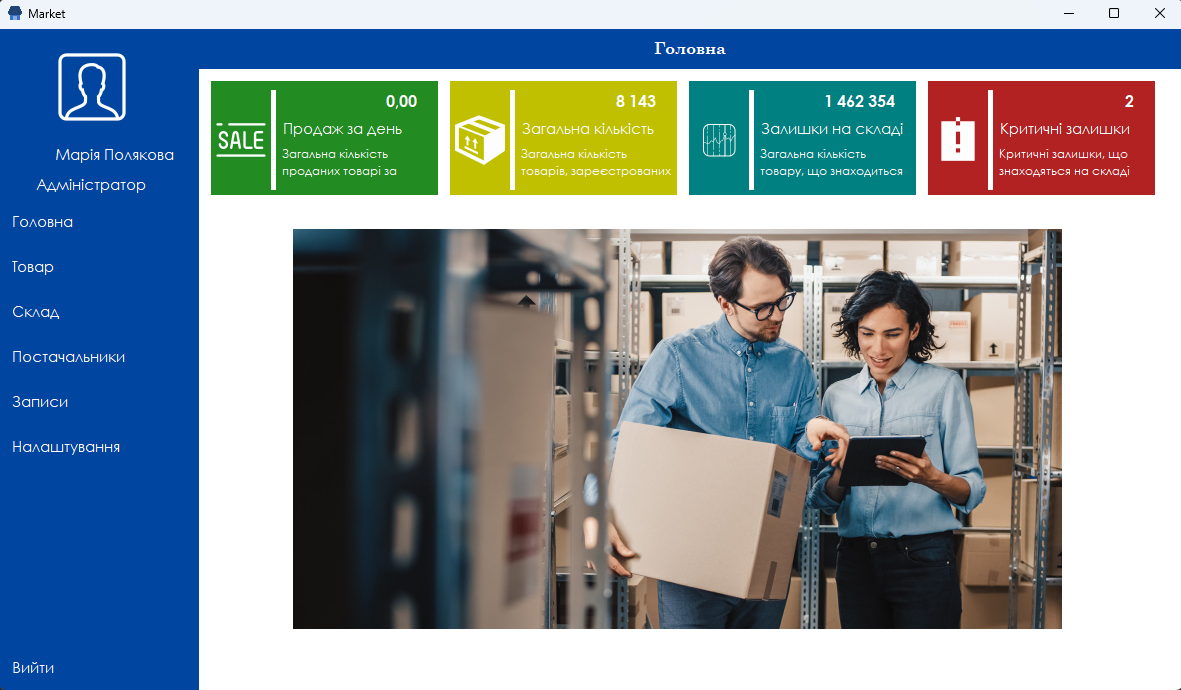
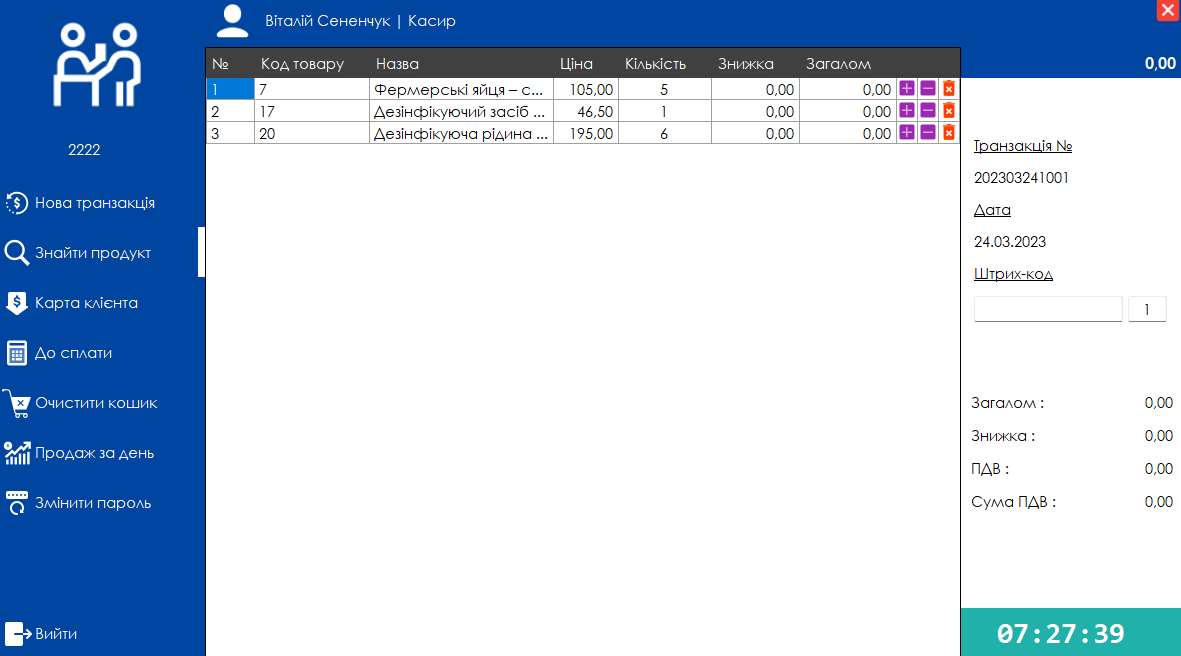


Рисунок 4.2



3.2 Алгоритм рішення задачі

Програмний додаток складається з багатьох екранних форм, кожна за яких виконує певну функцію та має унікальну логіку роботи.

При запуску програми спочатку відкривається форма де користувачу потрібно ввести логін та пароль. Після чого на основі ролі користувача відкриється або головна форма для адміністратора або головна форма для касира. Кожна з яких має свій унікальний функціонал та інтерфейс.

private void btnLogin\_Click(object sender, EventArgs e) – перевірка користувача та відкриття форми відповідно до його ролі;

private void btnProductList\_Click(object sender, EventArgs e) – відкриття форми зі списком продуктів;

private void dgvProduct\_CellContentClick(object sender, DataGridViewCellEventArgs e) – відкриття форми із інформацією про товар;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми реєстрації продукту;

private void btnCategory\_Click(object sender, EventArgs e) – відкриття форми зі списком категорій;

private void dgvCategory\_CellContentClick(object sender, DataGridViewCellEventArgs e) – видалення категорії;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми для реєстрації нової категорії;

private void btnStockEntry\_Click(object sender, EventArgs e) – відкриття форми зі списком закупок;

private void dgvStockIn\_CellDoubleClick(object sender, DataGridViewCellEventArgs) – відкриття форми з інформацією про закупівлю;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми для створення нової закупки;

private void btnStockAdjustment\_Click(object sender, EventArgs e) – відкриття форми для списання продукції;

private void btnSupplier\_Click(object sender, EventArgs e) – відкриття форми зі списком постачальників;

private void btnPosRecord\_Click(object sender, EventArgs e) – відкриття форми для перегляду звітів;

private void btnUser\_Click(object sender, EventArgs e) – відкриття форми налаштування користувачів;

private void btnStore\_Click(object sender, EventArgs e) – відкриття форми налаштування магазину;

private void btnNTran\_Click(object sender, EventArgs e) – оновлення форми длял створення нової транзакції;

private void btnSearch\_Click(object sender, EventArgs e) – відкриття форми зі списком продуктів для їх додавання в кошик;

private void btnDiscount\_Click(object sender, EventArgs e) – відкриття форми для додавання картки клієнта;

private void btnSettle\_Click(object sender, EventArgs e) – відкриття форми для підтвердження суми та видачі решти;  
private void btnClear\_Click(object sender, EventArgs e) – оновлення форми з метою видалення всіх продуктів з кошика;

private void btnPass\_Click(object sender, EventArgs e) – відкриття форми для зміни паролю для поточного користувача;

private void btnLogout\_Click(object sender, EventArgs e) – вихід з програми;

# Програмування та тестування

## Розробка програми

Розроблена програма призначена лише для операційної системи Windows, тому програма несумісна з іншими операційними системами (наприклад: Linux, Debian, MacOS тощо).

Після запуску програми, користувач повинен буде ввести логін та пароль, після чого відкриється форма призначена для адміністратора або для касира в залежності від ролі користувача.

Для переходу на інші форми, користувач повинен обрати потрібний пункт меню.

На головній формі адміністратора в меню доступні наступні пункти:

* Головна;
* Товар:
* Список товарів;
* Категорії;
* Виробник;
* Склад:
* Список замовлень;
* Списання
* Постачальники;
* Записи;
* Налаштування:
* Користувач;
* Магазин;

Обравши пункт “Товар”, в меню з’являється 3 підпункти: Список товарів, Категорії, Виробник. Натиснувши на Список товарів, користувачеві відображається форма зі списком товарів та інформацією про нього при натискані певної кнопки. Натиснувши на Категорії, користувачеві відображається форма зі списком категорій. Натиснувши на Виробник, користувачеві відображається форма зі списком виробників товарів.

Обравши пункт “Склад”, в меню з’являється 2 підпункти: Список замовлень та Списання. Натиснувши на Список замовлень, користувачеві відображається форма зі списком замовлень. На цій формі також можна створити нове замовлення або редагувати існуючі. Натиснувши на Списання, користувачеві відображається форма для списання товарів.

Обравши пункт Постачальники, користувачеві відображається форма зі списком постачальників, на якій також можна зареєструвати нового постачальника.

Обравши пункт Записи, користувачеві відображається форма з усіма звітами та інструментами аналізу.

Обравши пункт “Налаштування”, в меню з’являється 2 підпункти: Користувач та Магазин. Натиснувши на пункт Користувач, користувачеві відображається форма для налаштування всіх користувачів зареєестрованих в системі. Натиснувши на пункт Магазин, користувачеві відображається форма для налаштування інформації про магазин.

На головній формі адміністратора в меню доступні наступні пункти:

* Нова транзакція;
* Знайти продукт;
* Карта клієнта;
* До сплати;
* Очисти кошик;
* Змінити пароль;

Натиснувши на пункт Нова транзакція на формі очищається список товарів, що були додані до кошика та створюється нова транзакція.

Обравши пункт Знайти продукт, користувачеві відображається форма зі списком продуктів, на якій він повинен обрати товар та підтвердити кількість, яка буде додана до кошика.

Обравши пункт Карта клієнта , користувачеві відображається форма для введення коду карти клієнта та реєстрації нової в випадку її відсутності. Карта клієнта надасть 5-ти відсоткову знижку на кожен продукт.

Обравши пункт До сплати, на екрані відобразиться форма, в якій потрібно ввести суму, яку надав клієнт, потім на формі підрахується решта. Після натиснення кнопки Підтвердити транзакція закриється та видрукується чек.

Обравши пункт Очистити кошик, на формі очиститься таблиця зі списком товарів в кошику, при цьому транзакція залишиться тією ж самою.

Обравши пункт Змінити пароль, касиру відобразиться форма на якій він може змінити пароль для свого профілю.

## Етапи налагодження

Налагодження - це процес виявлення причин виникнення помилки та її подальшого виправлення (на відміну від тестування, що є процесом виявлення самого факту існування помилки).

Процес Налагодження включає в себе:

* Вироблення системних вимог;
* Вироблення вимог до програмного забезпечення;
* Аналіз та проектування;
* Реалізацію - процес написання коду;
* Продуктизацію – процес вияснення відношення програмного забезпечення вимогам і доведення до них;
* Експлуатацію – процес використання програми.

Під час створення програмного продукту в середовищі Visual Studio використовувались інструменти для відлагодження програмного коду. Один з них - це відлагоджувальник, який дозволяє розробникам встановлювати точки зупинки в програмному коді та відстежувати значення змінних та виконання програмного коду по крокам. Відлагоджувальник також дозволяє розробникам виконувати розгалуження в програмному коді та виконувати відлагодження в певних умовах. Інші інструменти для відлагодження в Visual Studio включають в себе вікна «Вихідні теки» та «Вихідний вивід», які дозволяють розробникам переглядати вихідний код та відслідковувати виконання програмного коду. Вікна «Локальні» та «Глобальні» змінні дозволяють розробникам переглядати значення змінних під час виконання програмного коду.

## Типи помилок

Під час розробки програмного продукту можуть виникати різні типи помилок, які можуть бути класифіковані в кілька категорій. Нижче перераховано деякі з найпоширеніших типів помилок:

* + Синтаксичні помилки: ці помилки виникають під час компіляції програмного коду та пов'язані з невірним використанням синтаксичних елементів мови програмування, таких як знаки пунктуації, ключові слова, типи даних тощо.
  + Логічні помилки: ці помилки пов'язані з невірним виконанням програмного коду під час його виконання. Наприклад, програма може вивести неправильні результати через неправильні умови вибору, помилки у логіці обчислень або неправильний алгоритм.
  + Помилки виконання: ці помилки виникають під час виконання програми та пов'язані з недостатнім обсягом ресурсів, таких як пам'ять або процесорний час, що призводить до аварійного завершення програми.
  + Помилки взаємодії: ці помилки пов'язані з невірною взаємодією програмного продукту з іншими програмами, операційним середовищем або залежними компонентами. Наприклад, програма може взаємодіяти з базою даних, але через помилки підключення вона не може здійснювати запити.
  + Помилки конфігурації: ці помилки пов'язані з неправильним налаштуванням програмного продукту або залежних компонентів, що можуть призводити до неправильної роботи програми або до аварійного завершення.
  + Помилки безпеки: ці помилки пов'язані з недостатньою безпекою програмного продукту, що може призвести до несанкціонованого доступу до системи, крадіжки даних або вірусного атаки.

Кожен помилок має свої особливості та специфічні методи їх виявлення та виправлення. Для виявлення синтаксичних та логічних помилок використовують методи відлагодження та тестування. Для виявлення помилок виконання та взаємодії можуть використовуватись моніторинг, аналіз логів та діагностика проблем. Помилки конфігурації можуть бути виявлені за допомогою перевірки налаштувань системи та залежних компонентів. Для виявлення помилок безпеки можуть використовуватись методи тестування на проникнення та аудит безпеки програмного продукту.

Важливо виявляти та виправляти помилки якомога раніше в процесі розробки, оскільки чим пізніше помилка буде виявлена, тим важче її буде виправити. Тому важливо забезпечувати високу якість коду та правильну його організацію, а також використовувати ефективні методи виявлення та виправлення помилок.

Також під час розробки програмного продукту активно використовувався механізм try-catch. Це механізм обробки помилок, який дозволяє контролювати помилки в програмі та забезпечує їх відловлювання та обробку, щоб уникнути краху програми. Конструкція try включає блок коду, який потенційно може породити виключення або помилку, і блок catch, який оброблює ці виключення та надає інструкції щодо їх обробки. Наприклад, у випадку, коли ми виконуємо ділення на нуль, програма породить помилку DivisionByZeroException. Ми можемо використати конструкцію try-catch для перехоплення цієї помилки та виведення користувачеві повідомлення про помилку замість краху програми.

В ході розробки програмного забезпечення виникали деякі помилки різного типу. Приклад синтаксичної помилки в курсовому проекті (див. рис. 4.1)

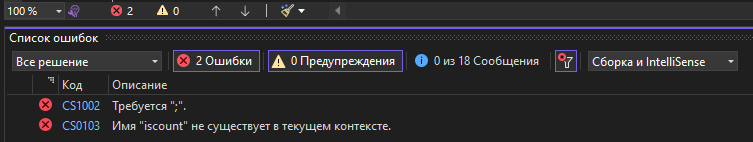


Рисунок 4.1 – Синтаксична помилка