МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

Курсова робота

з дисципліни «Програмування під платформу .NET»

Варіант 12

Студент Мальцев Є.Д.

Група ІН-81/2

Перевірила ст. викладач к.т.н. Москаленко А.С

СУМИ - 2020

**ЗМІСТ**

[Вступ 3](#_Toc43840813)

[1. Мета та постановка задачі 4](#_Toc43840814)

[1.1. Мета 4](#_Toc43840815)

[1.2. Постановка задачі 4](#_Toc43840816)

[2. Теоретична частина 6](#_Toc43840817)

[3. Практична частина 8](#_Toc43840818)

[3.1. Опис класів проекту 8](#_Toc43840819)

[3.2. Таблиці полів класів 9](#_Toc43840820)

[3.3. Методи класів 10](#_Toc43840821)

[4. Приклад роботи програми 11](#_Toc43840822)

[Висновки 15](#_Toc43840823)

[Література 16](#_Toc43840824)

[Додаток a 17](#_Toc43840825)

# ВСТУП

Дана курсова робота спрямована на покращення навичок програмування на мові C# та загального розвитку навичок роботи з програмним забезпеченням та кодом.

У процесі виконання цієї курсової буде розглянуто два завдання. Першим завданням є відповідь на теоретичне питання. Для даного варіанту курсової роботи буде розкрите питання про системи контролю версій.

Другим завданням курсової роботи буде створення додатку на мові програмування C#, що має реалізувати усі можливості вказані у практичному завданні та при необхідності або при бажанні виконавця буде розширений додатковим функціоналом.

Для демонстрації працездатності програми та її функціоналу буде наведений детальний опис усіх дій, що може зробити користувач у цьому додатку та їх результати. Усі дії будуть мати візуальне підтвердження за допомогою зображень головного вікна програми та діалогових вікон.

Опис усіх полів та методів реалізованих у додатку буде наведено у тексті курсової роботи з детальним поясненням для кожного поля та методу.

Також файли ресурсів, з яких береться інформація або в які вона заноситься будуть представлені у тексті курсової роботи.

# МЕТА ТА ПОСТАНОВКА ЗАДАЧІ

## **Мета**

Метою курсової роботи є створення програми, що буде забезпечувати простий інтерфейс для користувача та надавати йому можливість здійснювати покупки через дану програму, як через магазин. Окрім цього метою курсової роботи є також засвоєння теоретичних знань про UML діаграми.

## **Постановка задачі**

**Теоретична частина:** розповісти про основні поняття пов’язані системами контролю версій.

**Практична частина:** Створити програму, яка буде описувати класи **Магазин**, **Товар** та **Покупець**.

В класі **Магазин** повинна бути можливість:

* додати в кошик товар (інформація про корзину буде виводитись у форматі: зроблено замовлення у кількості N штук на суму Z грн.);
* видалити товар;
* зробити замовлення/підтвердити.

При підтвердженні замовлення розраховується сума знижки (наприклад на суму V знижка 5%, а на суму V2 – 7%).

Доставка товару буде оплачуватись з огляду на суму замовлення (розрахувати або фіксовану оплату, або в залежності від суми замовлення, або безкоштовна доставка при замовленні вище 1000 грн, наприклад).

Доставка в обмежений список міст здійснюється тільки певними службами доставки. При підтвердженні замовлення повідомити Покупця про це, або відразу вивести відповідну інформацію для ознайомлення. В день народження та за три дні після діє фіксована знижка на покупку – 10%.

Реалізувати в класі можливість пошуку товару по ключовим словам та можливість відсортувати товар по ціні, результат виводити на екран або у файл.

**Магазин** повинен мати назву, контактну інформацію та часи роботи. Якщо замовлення буде відбуватись поза робочий час – повідомити Покупця, що обробка відбудеться в робочі часи.

Клас **Покупець** повинен містити дані про покупця: ПІП, дата народження, місто проживання (опціонально), дату здійснення замовлення та час.

Покупець повинен мати можливість ознайомитись з контактною інформацією магазину та переглянути суму замовлення зі знижкою.

Клас **Товар** містить назву товару та ціну.

Розширити класи при необхідності потрібними методами, полями та властивостями.

За бажанням реалізацію можна виконати застосовуючи Windows form.

При відсутності бази даних дані можна зберігати у файлах.

1. **ТЕОРЕТИЧНА ЧАСТИНА**

Розробка складних інформаційних систем вимагає узгодженої роботи цілої групи програмістів. Проблеми організації взаємодії при розробці програмного продукту великою командою розробників зазвичай вирішується використанням системи керування версіями програмного коду (Version Control System, VCS) [1]. Така система дозволяє керувати поступовими змінами внесеними в електронні документи та відміняти цізміни у разі необхідності.

Система керування версіями  — програмний інструмент для керування версіями одиниці інформації: [вихідного коду програми](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), [скрипту](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82), веб-сторінки, [веб-сайту](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82), 3D-моделі, текстового документу тощо.

Система керування версіями — інструмент, який дозволяє одночасно, не заважаючи один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при [розробці програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) для відстеження, документування та контролю над поступовими змінами в електронних документах: у [сирцевому коді](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4) [застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігаються дані про автора зробленої зміни та її час.

Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Існують два основні типи систем керування версіями: з централізованим сховищем та розподіленим.

Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки.

[Гілки](https://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BB%D0%BA%D0%B0_(%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F%D0%BC%D0%B8)) зазвичай використовують для зберігання експериментальних, незавершених(alpha, beta) та повністю робочих версій проекту(final).

Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

Використання системи контролю версії є необхідним для роботи над великими проектами, над якими одночасно працює велика кількість розробників. Системи контролю версії надають ряд додаткових можливостей:

* Можливість створення різних варіантів одного документу;
* Документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
* Реалізує функцію [контролю доступу](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D1%83_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) користувачів до файлів. Є можливість його обмеження;
* Дозволяє створювати документацію проекту з поетапним записом змін в залежності від версії;
* Дозволяє давати пояснення до змін та документувати їх;

Існує цілий клас програм, які називаються системами контролю версій, і всі вони вирішують приблизно однакове завдання. Зараз існує три найбільш популярних рішення: Git, SVN і Mercurial. Переваги Git полягає в тому, що він найбільш популярний в середовищі розробників. Mercurial практично ні чим не відрізняється від Git, відмінності мінімальні, але він набагато менш поширений. SVN має цілий ряд недоліків у порівнянні з Git.

1. **ПРАКТИЧНА ЧАСТИНА**

Для виконання практичного завдання нам знадобиться створити додаток, який виконує усі вимоги поставленого завдання. Для цього нам знадобиться створити певну кількість класів, що разом утворять наш додаток. Для цього наведемо загальну діаграму проекту та діаграми класів, які в неї входять. Дані діаграми зображені за рисунках 1-3.

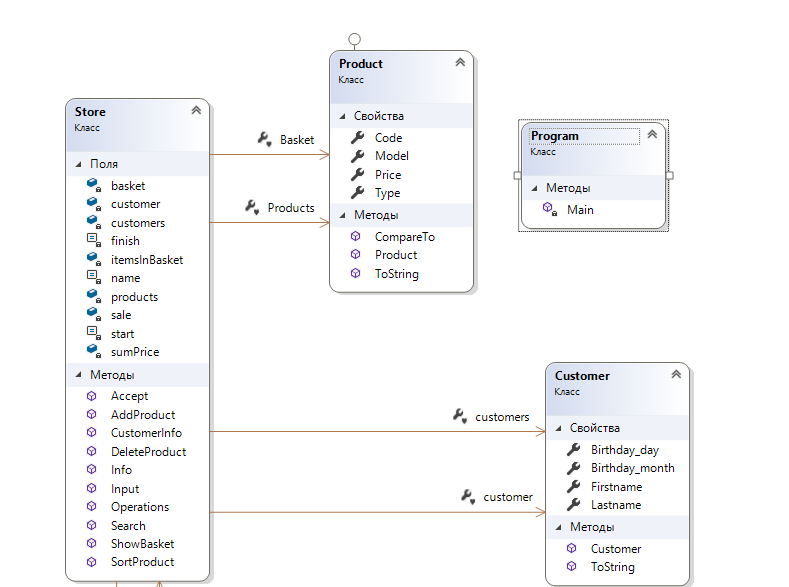


Рисунок 1 – Діаграма класів проекту.

* 1. **Опис класів проекту**

Для кращого розуміння взаємодії елементів програми необхідно описати класи, які в неї входять. Тож опишемо їх.

Клас “**Program**” призначений для зберігання шляхів для файлів ресурсів даної програми та статичного об’єкту класу “Store” через який і буде виконуватися основна кількість операцій. Інформація про поля класу наведена у таблиці 4.

Клас “**Customer**” призначений для зберігання інформації про одного користувача. Інформація про поля класу наведена у таблиці 1.

Клас “**Product**” призначений для зберігання інформації про одиницю продукту. Інформація про поля класу наведена у таблиці 2

Клас “**Store**” призначений для зберігання списку усіх наявних продуктів у магазині, та користувачів. 3дійснює всі необхідні операції пов’язані з роботою магагазину.

* 1. **Таблиці полів класів**

Зараз опишемо поля усіх класів наявних у програмі. Для зручності кожен клас має свою таблицю.

Таблиця 1 –Таблиця полів класу Customer.

| Назва змінної | Тип | Призначення |
| --- | --- | --- |
| Firstname | String | Зберігає ім’я користувача. |
| Lastname | String | Зберігає прізвище користувача. |
| Birthday\_day | int | Зберігає день народження користувача. |
| Birthday\_month | int | Зберігає місяць народження користувача. |

Таблиця 2 – Таблиця полів класу Product.

| Назва змінної | Тип | Призначення |
| --- | --- | --- |
| Code | int | Зберігає ідентифікаційний номер продукта. |
| Type | String | Зберігає тип продукта. |
| Model | String | Зберігає назву продукта. |
| Price | float | Зберігає ціну продукта. |

Таблиця 3 – Таблиця полів класу Store.

| Назва змінної | Тип | Призначення |
| --- | --- | --- |
| name | const string | Назва магазину |
| sumPrice | double | Загальна ціна продуктів корзини |
| itemsInBasket | int | Кількість товарів у корзині |
| Start, finish | int | Години роботи магазину |
| sale | int | Змінні для оцінки дисконтів |
| products | List<Product> | Каталог продуктів |
| basket | List<Product> | Корзина товарів |
| customers | List<Customer> | Перелік користувачів |
| customer | Customer | Актуальный користувач |

* 1. Методи класів

Опишемо методи, які були створені у програмі. Їх перелік наведемо у таблиці 4.

Таблиця 4 – Перелік усіх методів.

| **Назва** | **Призначення** |
| --- | --- |
| AddProduct | Додати продукт в корзину |
| DeleteProduct | Видалити продукт з корзини |
| CustomerInfo | Отримати інформацію про користувача |
| Search | Пошук продукту за назвою |
| ShowBasket | Отрмати продукти з корзини |
| SortProduct | Сортувати продукти за ціною |
| Operations | Запуск відображення списку команд та виконання списку команд |
| Accept | Підтвердити заказ |
| Input | Отримати товари з файлу |

# Приклад Роботи програми

Для початку роботи з програмою нам заповнити файл каталогу продуктів та файл з інформацією про користувача.

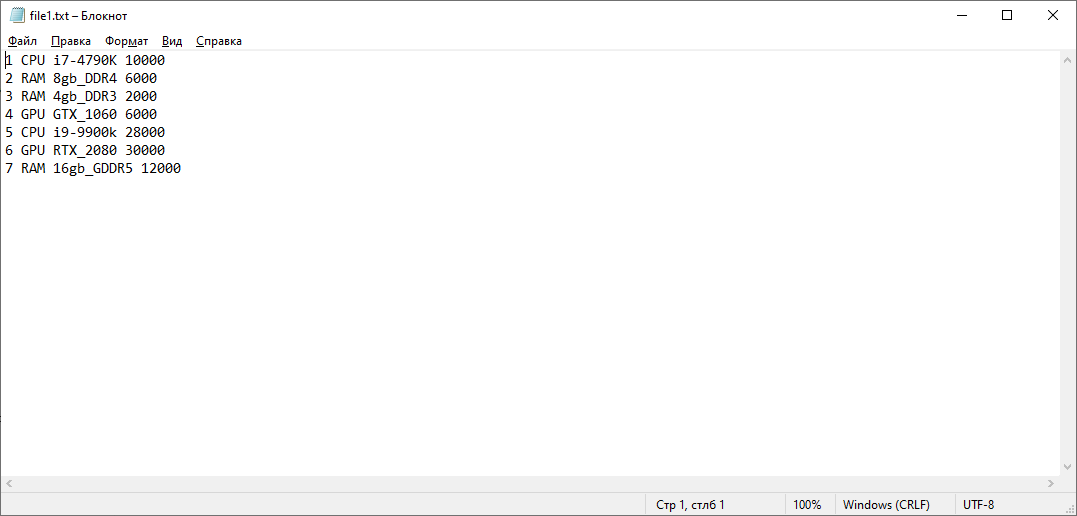


Рисунок 7 ̶ Каталог продуктів.

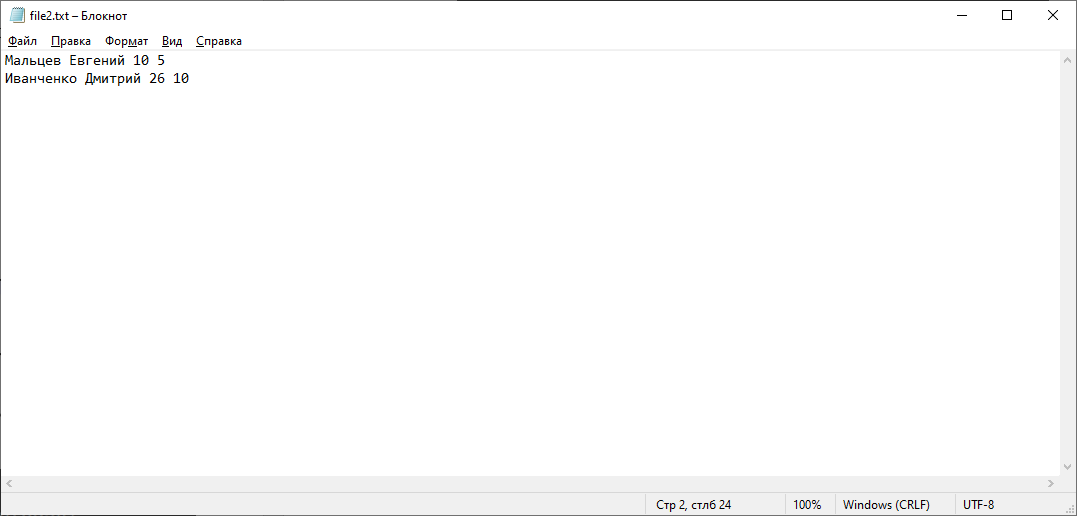


Рисунок 8 ̶ Дані користувачів

Після запуску додатку ми отримаємо перелік користувачів та маємо можливість обрати свій обліковий запис.

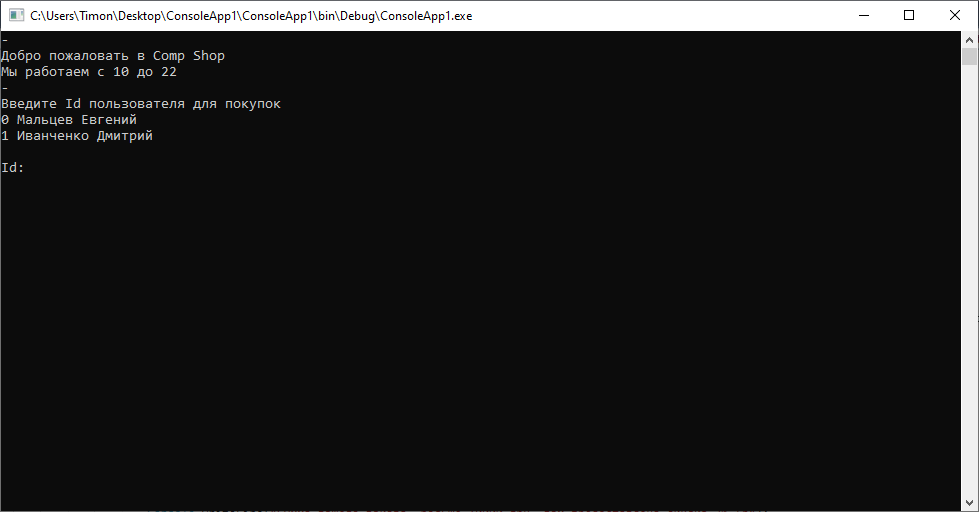


Рисунок 9 ̶ Перелік користувачів

Наступним кроком програма виведе список доступних для покупки товарів.

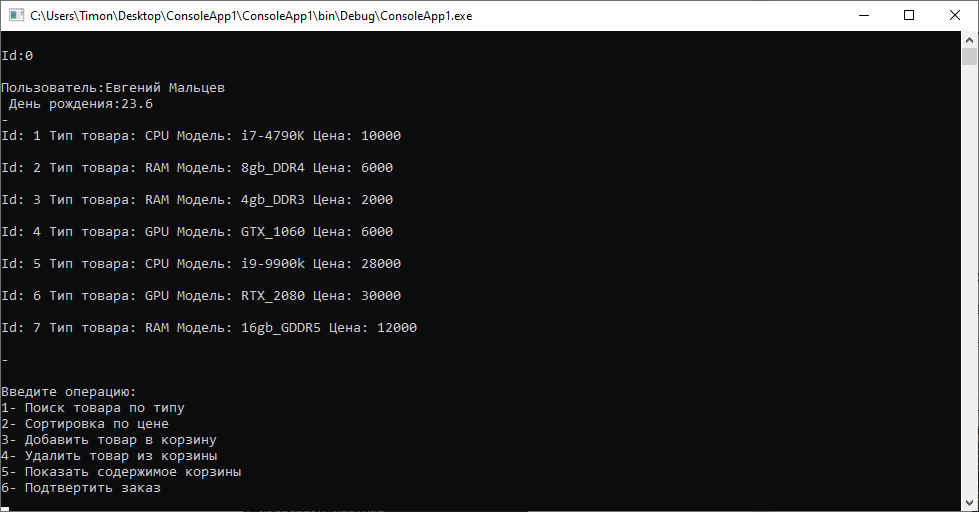


Рисунок 10 ̶ Каталог товарів

Наступним етапом буде виведено список можливих для користувача операцій.

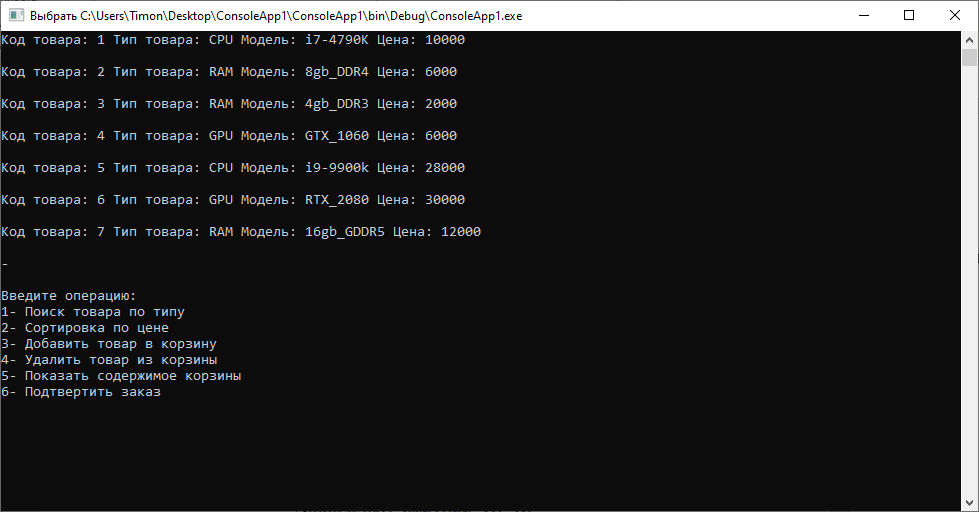
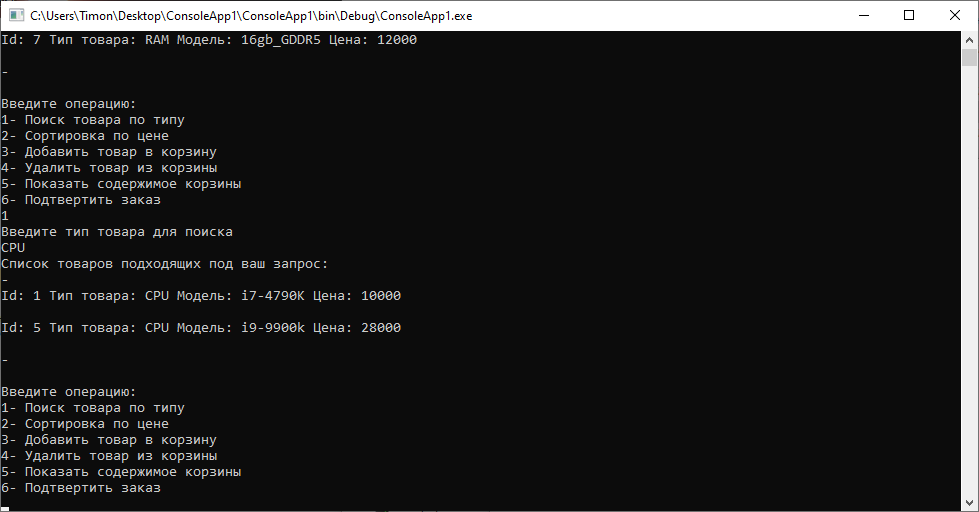


Рисунок 11 ̶ Список доступних операцій

Програма підтримує операції пошуку та сортування продуктів по назві та ціні відповідно, додавання та видалення продуктів з корзини та підтвердження заказу користувачу. Розглянемо кожну операцію окремо.

Для пошуку товарів по типу достатньо ввести ключове слово, або його частину, які містяться в необхідному типу:

  
Рисунок 12 ̶ Пошук продуктів

При натисканні на кнопку 2, продукти будуть відсортовані відповідно по обраному критерію(зростання/спадання).

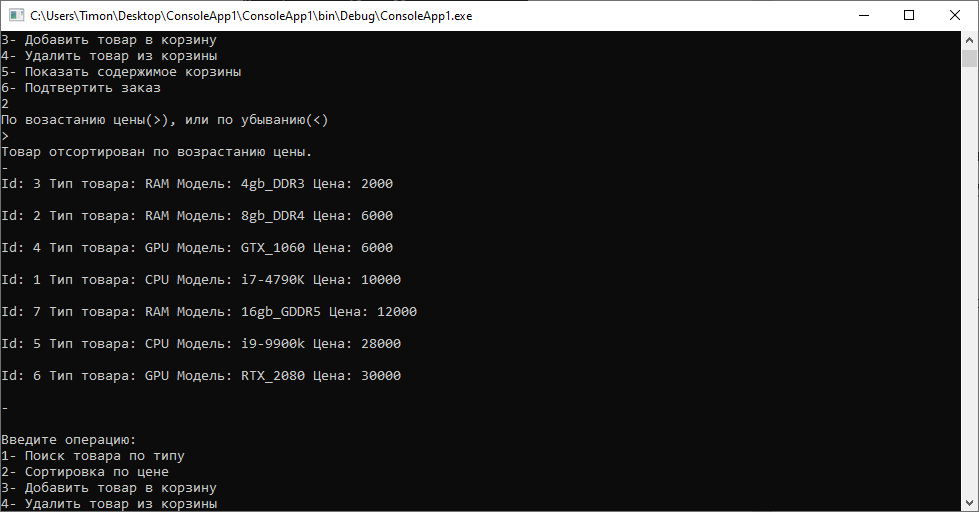


Рисунок 13 ̶ Сортування по зростанні ціни

Для того щоб додати продукт в корзину необхідно вибрати третю операцію, а потім ввести код необхідного продукту.

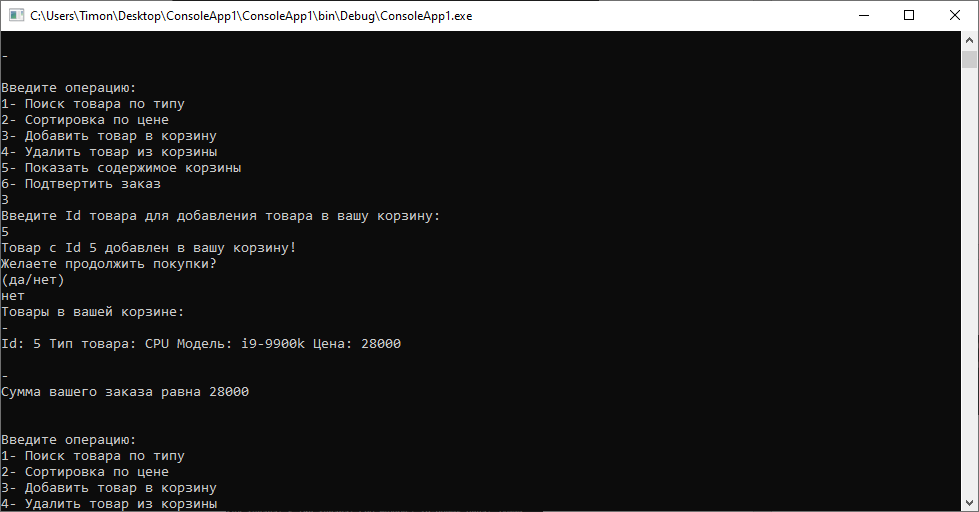


Рисунок 14 ̶ Додаваня продукту

Для того, щоб видалити продукт необхідно провести аналогічні маніпуляції, обравши відповідну операцію.

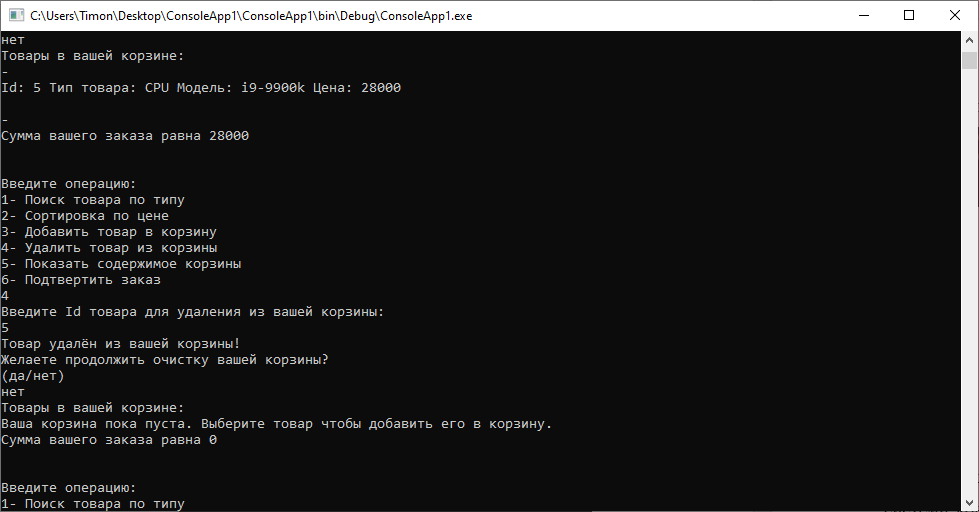


Рисунок 15 ̶ Видалення продукту з корзини

Перегляд продуктів в корзині виглядає наступним чином.

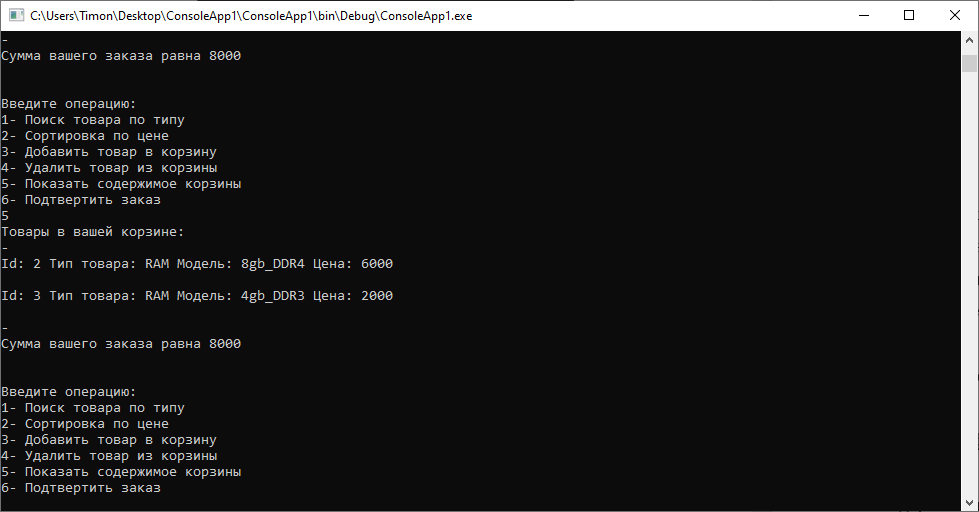


Рисунок 16 ̶ Перегляд корзини

Для того щоб підвердити заказ необхідно обрати операцію 6. У випадку якщо вона буду обрана в неробочі для магазину години, то користувач побаче відповіду повідомлення.

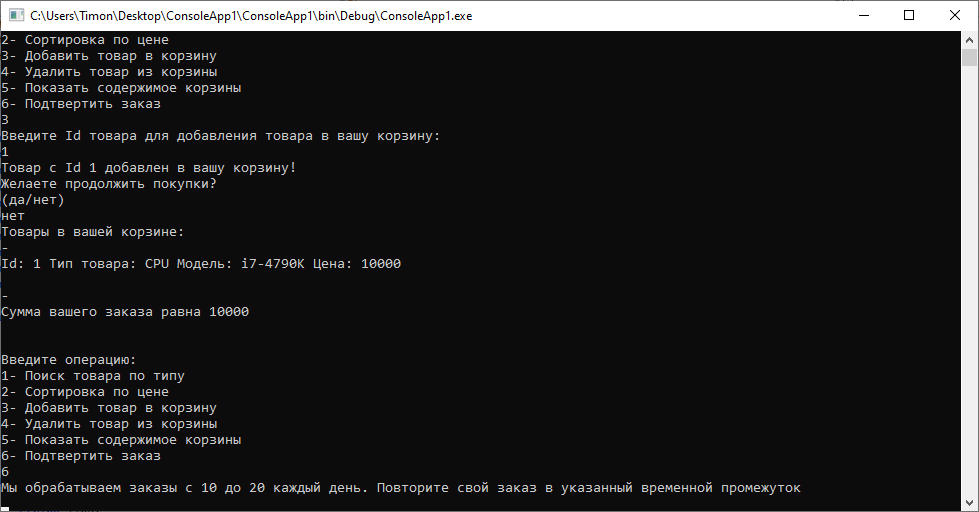


Рисунок 17 ̶ Підвердження заказу в неробочий для магазину час

В іншому випадку буде розраховано процес буде закінчено успішно. А також надані знижки до дня народження, або замовлення на суму більшу за задане значення.

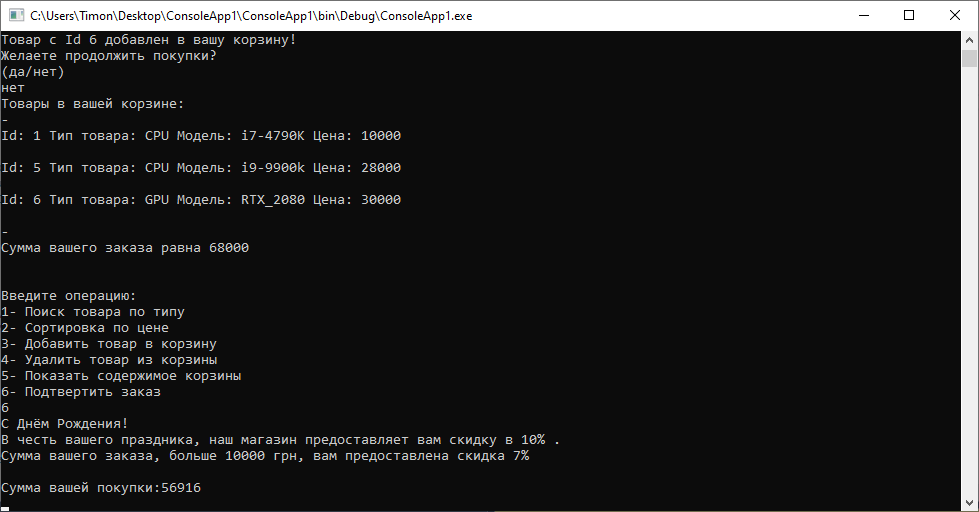


Рисунок 18 ̶ Підвердження заказу

# ВИСНОВКИ

У ході даної курсової роботи були засвоєні теоретичні знання про системи контролю версій, а також згідно поставленої практичної задачі був створений додаток на мові програмування С# у середовищі Visual Studio. Цей додаток реалізовує функціонал інтернет-магазину за допомогою якого користувач може легко переглядати список наявних товарів, додавати їх у корзину або видаляти товар з корзини.

Текст програми та скріншоти таблиць даних наведені у тексті програми. Основний функціонал даної програми зображено у вигляді опису дій, які необхідно виконати, щоб отримати певний результат. Усі етапи дій підкріпляються відповідними до них скріншотами.

# ЛІТЕРАТУРА

1. Стив Макконнелл. Совершенный код = Code complete. СПб.: Питер, 2005.
2. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System
3. Pro Git book, written by Scott Chacon and Ben Straub системы контроля версий

# ДОДАТОК A

У цьому додатку наведений повний код програми розподілений по класам, також повний код програми можна знайти за посиланням на “GitHub” наведеним у тексті курсової роботи.

Клас “Product”:

using System;

namespace ConsoleApp1

{

class Product:IComparable

{

public int Code { get; private set; }

public string Type { get; private set; }

public string Model { get; private set; }

public float Price { get; private set; }

public Product (int code, string type, string model, float price)

{

Code = code;

Type = type;

Model = model;

Price = price;

}

public override string ToString()

{

string str = String.Empty;

str += ("Id: " + Code + " ");

str += ("Тип товара: " + Type + " ");

str += ("Модель: " + Model + " ");

str += ("Цена: " + Price + " ");

return str;

}

public int CompareTo(object obj)

{

Product a = (Product)obj;

if (this.Price == a.Price)

return 0;

else if (this.Price > a.Price)

return 1;

else

return -1;

}

}

}

Клас “Customer”:

using System;

namespace ConsoleApp1

{

class Customer

{

public string Firstname { get; private set; }

public string Lastname { get; private set; }

public int Birthday\_day { get; private set; }

public int Birthday\_month { get; private set; }

public Customer(string firstname, string lastname, int birthday\_day, int birthday\_month)

{

Firstname = firstname ?? throw new ArgumentNullException(nameof(firstname));

Lastname = lastname ?? throw new ArgumentNullException(nameof(lastname));

Birthday\_day = birthday\_day;

Birthday\_month = birthday\_month;

}

public override string ToString()

{

return ("\nПользователь:" + Lastname + " " + Firstname + "\n День рождения:" + Birthday\_day + "." + Birthday\_month);

}

}

}

Клас “Store”:

using System;

using System.Collections.Generic;

using System.IO;

namespace ConsoleApp1

{

class Store

{

private const string name = "Comp Shop";

private const int start = 10;

private const int finish = 22;

private double sumPrice = 0;

private int itemsInBasket;

private int sale = 10000;

private List<Product> products = new List<Product>();

private List<Product> basket = new List<Product>();

private List<Customer> customers = new List<Customer>();

private Customer customer;

public void Info()

{

String str = String.Empty;

str += "-\n";

str += ("Добро пожаловать в " + name + "\n");

str += ("Мы работаем с " + start + " до " + finish + "\n");

str += "-";

Console.WriteLine(str);

}

public void Input(string file)

{

string line;

using (StreamReader MyFile = new StreamReader(file))

{

while ((line = MyFile.ReadLine()) != null)

{

string[] data = line.Split(' ');

Product NewProduct = new Product(int.Parse(data[0]), data[1], data[2], float.Parse(data[3]));

products.Add(NewProduct);

}

}

if (products.Count == 0)

{

Console.WriteLine("Товаров нет в наличии");

Environment.Exit(1);

}

else

{

Console.WriteLine("-");

for (int i = 0; i < products.Count; i++)

{

Console.WriteLine(products[i] + "\n");

}

Console.WriteLine("-");

}

}

public void AddProduct()

{

string continueBuying;

while (true)

{

Console.WriteLine("Введите Id товара для добавления товара в вашу корзину:");

int code\_1 = int.Parse(Console.ReadLine());

int t = 0;

for (int i = 0; i < products.Count; i++)

{

if (products[i].Code == code\_1)

{

basket.Add(products[i]);

t++;

sumPrice += products[i].Price;

Console.WriteLine("Товар с Id {0} добавлен в вашу корзину!", products[i].Code);

}

}

if (t == 0)

{

Console.WriteLine("Товара с таким Id не существует");

}

itemsInBasket = basket.Count;

Console.WriteLine("Желаете продолжить покупки?");

Console.WriteLine("(да/нет)");

continueBuying = Console.ReadLine();

if (String.Equals(continueBuying, "да"))

continue;

else

{

if (String.Equals(continueBuying, "нет"))

{

ShowBasket();

break;

}

else

{

Console.WriteLine("Данная операция не поддерживается. Выберите одну из предложенных выше.");

break;

}

}

}

}

public void ShowBasket()

{

Console.WriteLine("Товары в вашей корзине:");

if (basket.Count != 0)

{

Console.WriteLine("-");

for (int i = 0; i < basket.Count; i++)

{

Console.WriteLine(basket[i] + "\n");

}

Console.WriteLine("-");

}

else

Console.WriteLine("Ваша корзина пока пуста. Выберите товар чтобы добавить его в корзину.");

Console.WriteLine("Сумма вашего заказа равна {0}", sumPrice);

Console.WriteLine("");

}

public void DeleteProduct()

{

string continueCleaning;

while (true)

{

int t = 0;

Console.WriteLine("Введите Id товара для удаления из вашей корзины:");

int code\_1 = int.Parse(Console.ReadLine());

if (basket.Count != 0)

{

for (int i = 0; i < basket.Count; i++)

{

if (basket[i].Code == code\_1)

{

sumPrice -= basket[i].Price;

basket.Remove(basket[i]);

itemsInBasket--;

Console.WriteLine("Товар удалён из вашей корзины!");

t++;

}

}

if (t == 0)

{

Console.WriteLine("В вашей корзине нет товара с таким Id");

break;

}

Console.WriteLine("Желаете продолжить очистку вашей корзины?");

Console.WriteLine("(да/нет)");

continueCleaning = Console.ReadLine();

if (String.Equals(continueCleaning, "да"))

continue;

else

{

if (String.Equals(continueCleaning, "нет"))

{

ShowBasket();

break;

}

else

{

Console.WriteLine("Данная операция не поддерживается");

break;

}

}

}

else

{

Console.WriteLine("Ваша корзина пуста, для начала добавьте товар, чтобы удалить его");

break;

}

}

}

public void SortProduct()

{

Console.WriteLine("По возастанию цены(>), или по убыванию(<)");

string tSort = Console.ReadLine();

switch (tSort)

{

case (">"):

Console.WriteLine("Товар отсортирован по возрастанию цены.");

Console.WriteLine("-");

products.Sort();

for (int i = 0; i < products.Count; i++)

{

Console.WriteLine(products[i] + "\n");

}

Console.WriteLine("-");

break;

case ("<"):

Console.WriteLine("Товар отсортирован по убыванию цены.");

Console.WriteLine("-");

products.Sort();

products.Reverse();

for (int i = 0; i < products.Count; i++)

{

Console.WriteLine(products[i] + "\n");

}

Console.WriteLine("-");

break;

default:

Console.WriteLine("Недопустимая операция.");

break;

}

}

//поиск по критерию на выбор

public void Search()

{

int t = 0;

Console.WriteLine("Введите тип товара для поиска");

string need\_type = Console.ReadLine();

Console.WriteLine("Список товаров подходящих под ваш запрос:");

Console.WriteLine("-");

for (int i = 0; i < products.Count; i++)

{

if (String.Equals(products[i].Type, need\_type))

{

Console.WriteLine(products[i] + "\n");

t++;

}

}

Console.WriteLine("-");

if (t == 0)

{

Console.WriteLine("В нашем магазине нет товаров такого типа.");

}

}

public void Accept()

{

if (basket.Count == 0)

{

Console.WriteLine("Ваша корзина пуста. Для подтверждения заказа, выберите хотя бы один товар.");

}

else

{

if (DateTime.Now.Hour < start || DateTime.Now.Hour > finish)

{

Console.WriteLine("Мы обрабатываем заказы с {0} до {1} каждый день. Повторите свой заказ в указанный временной промежуток", start, finish);

}

else

{

if (customer.Birthday\_day == DateTime.Now.Day && customer.Birthday\_month == DateTime.Now.Month)

{

Console.WriteLine("С Днём Рождения!\nВ честь вашего праздника, наш магазин предоставляет вам скидку в 10% .");

sumPrice \*= 0.9;

}

if (sumPrice >= sale)

{

sumPrice \*= 0.93;

Console.WriteLine("Сумма вашего заказа, больше 10000 грн, вам предоставлена скидка 7% \n");

}

Console.WriteLine("Сумма вашей покупки:{0}", sumPrice);

}

}

}

public void CustomerInfo(string file)

{

Console.WriteLine("Введите Id пользователя для покупок");

using (StreamReader MyFile = new StreamReader(file))

{

string line;

while ((line = MyFile.ReadLine()) != null)

{

string[] data = line.Split(' ');

customers.Add(new Customer(data[0], data[1], int.Parse(data[2]), int.Parse(data[3])));

}

}

for (int i = 0; i < customers.Count; i++)

{

Console.WriteLine(i + " " + customers[i].Firstname + " " + customers[i].Lastname);

}

Console.Write("\nId:");

customer = customers[int.Parse(Console.ReadLine())];

Console.WriteLine(customer);

}

public void Operations()

{

Store menu = new Store();

Console.WriteLine();

Console.WriteLine("Введите операцию:\n" +

"1- Поиск товара по типу \n2- Сортировка по цене\n" +

"3- Добавить товар в корзину\n4- Удалить товар из корзины\n" +

"5- Показать содержимое корзины\n6- Подтвертить заказ");

string operation = Console.ReadLine();

switch (operation)

{

case ("1"):

Search();

Operations();

break;

case ("2"):

SortProduct();

Operations();

break;

case ("3"):

AddProduct();

Operations();

break;

case ("4"):

DeleteProduct();

Operations();

break;

case ("5"):

ShowBasket();

Operations();

break;

case ("6"):

Accept();

break;

default:

Console.WriteLine("Операция не существует");

break;

}

}

}

}