ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра програмної інженерії програмного забезпечення та кібербезпеки

|  |  |
| --- | --- |
|  | Захищено на кафедрі інженерії програмного забезпечення та кібербезпеки  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис членів комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

КУРСОВА РОБОТА

з дисципліни

“ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ”

НА ТЕМУ:

**Розробка програмного забезпечення з генерації тексту за ключовим словом "Generatext" мовою програмування C#**

(назва теми)

Студента факультету

інформаційних технологій

4 групи 2 курсу

**Авєріної Наталії Ігорівни**

Науковий керівник:

асистент кафедри інженерії програмного

забезпечен

ня та кібербезпеки

**Хорольська Карина Вікторівна**

Київ 2023

**ЗМІСТ**

[**ВСТУП** 2](#_Toc131210681)

[**РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОБЛЕМАТИКА.** 4](#_Toc131210682)

[**1.1 ….** 4](#_Toc131210683)

[**РОЗДІЛ 2. ПОРІВНЯЛЬНИЙ АНАЛІЗ НАЯВНОГО ПЗ ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ.** 5](#_Toc131210684)

[**2.1 Аналіз проблеми і методи її вирішення** 5](#_Toc131210685)

[**РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ «Generatext».** 8](#_Toc131210686)

[**3.1 Налаштування віртуального середовища розробки** 8](#_Toc131210687)

[**3.2 Технології, використані для розробки програмного продукту** 10](#_Toc131210688)

[**Висновок до розділу 3.** 11](#_Toc131210689)

[**ВИСНОВОК** 11](#_Toc131210690)

[**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ** 11](#_Toc131210691)

Вступ – 1-3

Розділ 1 – 5-7

Розділ 2 – 8-10

Розділ 3 – 7

Висновок – 1

22-27

# **ВСТУП**

Моя курсова робота присвячена розробці програмного забезпечення, яке називається «Generatext» та написане мовою програмування С#. Головною функцією додатку є виконання генерації тексту за ключовим словом, яке буде введено користувачем, та обраною ним же книгою.

На сьогоднішній день досить багато є додатків та сайтів, які виконують роботу з генерації тексту за нейронною мережею. Але за моїм дослідженням (передивившись достатньо сайтів за запитом «генератор тексту» в Google та Play Маркеті) мало доволі корисних та дієвих саме для написання нових творів по книзі або звичайному тексту, не враховуючи ChatGPT від компанії OpenAI, що зовсім нещодавно з’явився на просторах Інтернету та дуже добре справляється з генерацією абсолютно нового тексту на будь-яку тему.

Метою дослідження даного курсового проєкту є розробка програмного додатку з генерації тексту за ключовим словом на основі самостійно завантаженого тексту (наприклад книги).

Актуальність обраної тематики дослідження полягає у необхідності отримання правдоподібного речення або тексту на основі вже існуючої теми (книги, оповідання тощо).

Предметною областю розробки ПЗ з генерації тексту за ключовим словом є обробка природної мови (Natural Language Processing, NLP) та машинне навчання (Machine Learning, ML). Обробка природної мови включає в себе методи та алгоритми аналізу, інтерпретації та генерації текстів на природній мові.

…..

Загальна концепція

1-2 абзаци

Завдання курсової

Метою

Об’єктом дослідження

Предметом дослідження

Під час виконання проекту буде створена

мовою програмування, інструменти….

# **РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОБЛЕМАТИКА**

## **1.1 ….**

Генератор речень на основі ключового слова та тексту може бути корисним інструментом у різних галузях, де потрібне автоматичне створення текстів на певну тему. Наприклад, такий генератор може використовуватися для створення текстів для сайтів, блогів, рекламних матеріалів, презентацій, рекламних текстів і т.д.

Переваги використання генератора тексту включають:

* Економія часу: замість витрачання великої кількості часу на написання текстів, можна використовувати генератор, який автоматично створює тексти на певну тему.
* Якість текстів: якщо генератор добре налаштований та використовує якісне джерело тексту, то створені тексти можуть бути високої якості та мати хорошу читабельність та структуру.
* Широкий спектр застосування: генератор промов може використовуватись у різних галузях, де потрібне автоматичне створення текстів на певну тему.

1.2

Згідно з даними отриманими на сайті: <https://trends.google.com> можна проаналізувати попит, динаміку популярності подібних програм та дослідити прогноз на майбутнє.

Дивлячись на інформацію, зображену на рис. 1, можна зробити висновок, що попит на генератори тексту зростає найбільше під час кінця та початку навчального року, вступних кампаній тощо.

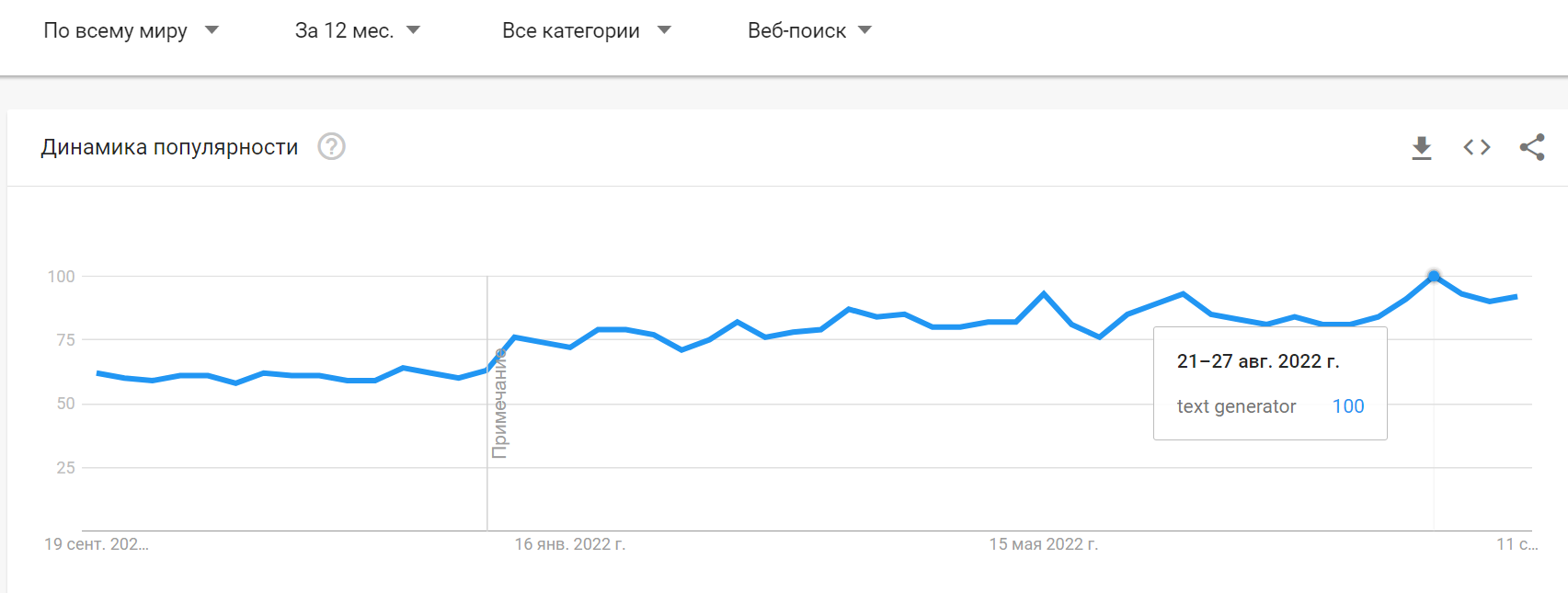


Рис. 1

На рис. 2 зображено динаміку запитів останні 5 років по всьому світові. Можна побачити, що, порівняно з 2018 роком, попит на подібні програми та сайти стає стабільно вищим, тобто можна сказати, що прогноз на майбутнє має додатні показники.

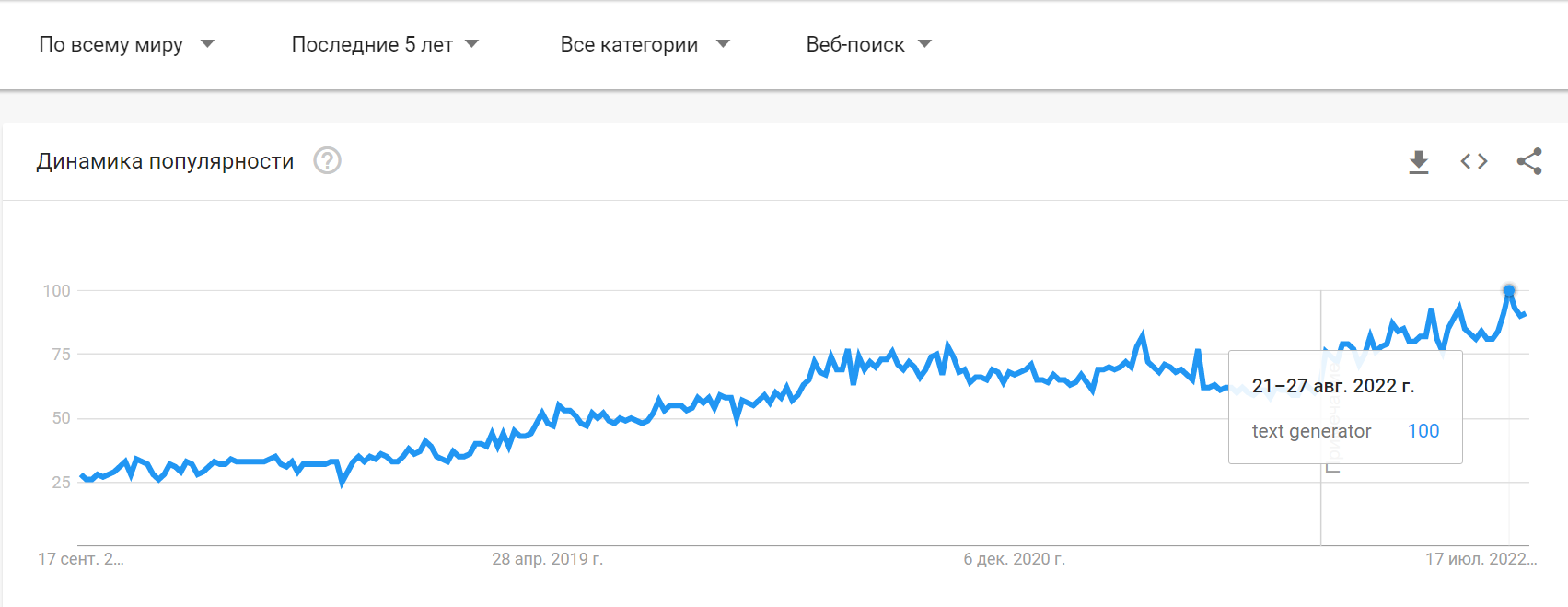


Рис. 2

Аналогічні програми, як показано на рис. 3, популярні майже по всьому світові. Отже можна зробити висновок, що розробка подібних програм, які будуть полегшувати рутинне життя користувачів на сьогоднішній день є доволі вигідною справою.

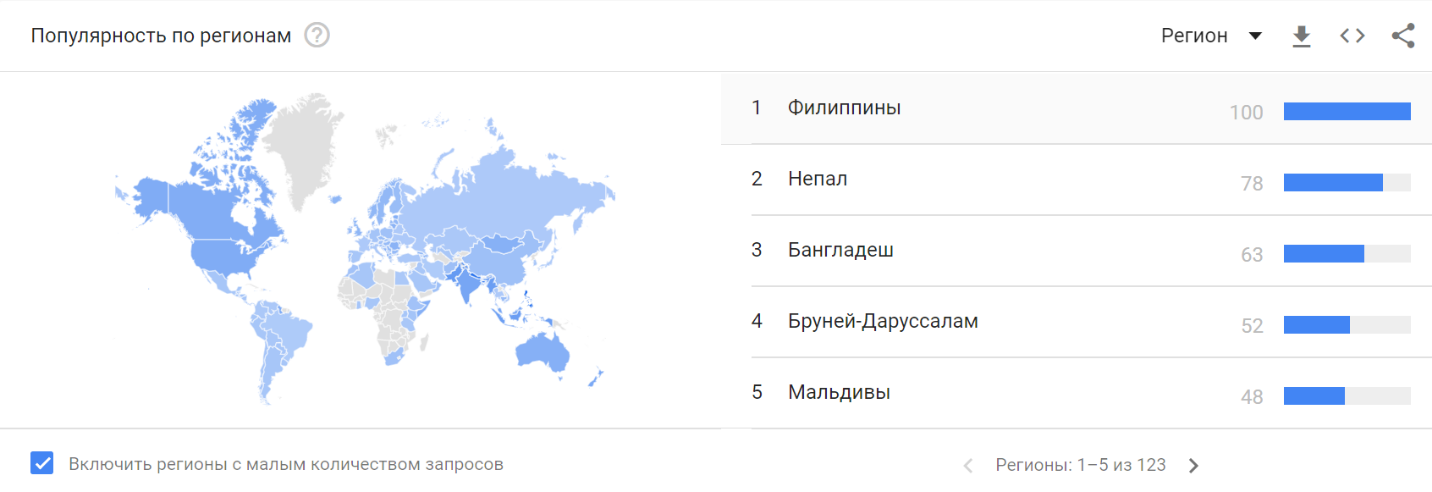


Рис. 3

Відмінністю від аналогічних програм буде те, що в комп’ютера для створення речень буде основа, завдяки якій текст не зійде із заданої теми.

1.1 Об’єктно-орієнтоване програмування…..,

1.2 Об’єктно-орієнтоване програмування…..,

Висновки до Розділу 1

5-7 листків

# **РОЗДІЛ 2. ПОРІВНЯЛЬНИЙ АНАЛІЗ НАЯВНОГО ПЗ ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ**

**2.1 Аналіз проблеми і методи її вирішення**

Проблема автоматичної генерації тексту за ключовим словом є актуальною в сучасному світі, де необхідно швидко створювати якісний та унікальний контент для веб-сайтів, соціальних мереж, медіа-платформ тощо. Також автоматизація генерації тексту забезпечує можливість заощадження часу та зусиль людей, які займаються написанням контенту.

Однією з основних проблем є необхідність створення різноманітного та цікавого контенту, що був би зрозумілий та корисний для різних аудиторій. Також важливим фактором є підтримання стилю та тону тексту, що може бути складним завданням для автоматичної генерації тексту.

Одним з основних завдань при розробці ПЗ є вибір методу генерації тексту. Існують різні підходи, такі як статистичні моделі, нейронні мережі, правила та шаблони. Кожен з цих методів має свої переваги та недоліки, і вибір методу залежить від конкретної задачі та вимог до результатів.

Для вирішення цієї проблеми було обрано використання N-граммних моделей. Вони дозволяють створювати текст на основі статистичного аналізу корпусу текстів. Ідея полягає у тому, що на основі вхідного тексту формується словник N-грамм, а потім генерується текст, використовуючи цей словник.

2.2 Вибір середовища та інструментів розробки

Мова програмування є необхідним інструментом у написанні будь-якої програми. Аналогічно до людського спілкування – не знаючи мови складно пояснити щось так щоб тебе зрозуміли. Так само і в програмуванні – комп’ютер не зрозуміє яку задачу треба виконати і як, якщо написати код незрозумілою йому мовою, наприклад українською.

Мова програмування – це формальний мовний засіб, який використовується для написання програмного коду, який згодом буде виконуватись комп'ютером або іншою обчислювальною машиною.

Суворіше визначення: мова програмування — це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії, які виконує виконавець (комп'ютер) під її управлінням.[1]

Історія мов програмування починається з 19 століття, коли було розроблено перші "машини на картах", які використовували спеціальні отвори в перфокартах для виконання обчислювальних операцій. У 20-му столітті з'явилися перші електронні комп'ютери, для яких були розроблені високорівневі і низькорівневі мови програмування.

Перша високорівнева мова програмування – FORTRAN, була розроблена 1950-х роках для наукових обчислень. Пізніше з'явилися інші мови високого рівня, такі як COBOL, BASIC та C, які використовуються і до сьогодні.

У 1980-х роках були розроблені об'єктно-орієнтовані мови програмування, такі як C++, які дозволяють програмістам створювати складні програми з більшим рівнем абстракції та повторного використання коду.

З появою Інтернету та розширенням індустрії програмного забезпечення з’явилися нові мови програмування, такі як JavaScript, Python та Ruby. Ці мови програмування використовуються для веб-розробки, машинного навчання, аналізу даних та багатьох інших задач.

Сьогодні існує безліч мов програмування, кожна з яких має свої особливості та призначення. Вибір мови програмування залежить від задачі, яку потрібно вирішити, та від особистих уподобань програміста.

Саме для реалізації системи генерації тексту за ключовим словом необхідні спеціальні інструменти, що дозволяють створювати та редагувати словник N-грамм. Один з найбільш ефективних інструментів для створення та редагування текстових даних є мова програмування Python, зокрема бібліотека NLTK, яка надає можливості для обробки природних мов та створення N-грамм.

Але для розробки такої системи було вибрано середовище програмування Visual Studio та мову програмування C#. Visual Studio було обрано з огляду на те, що це середовище програмування дозволяє швидко створювати різноманітні застосунки та інтерфейси, має велику кількість вбудованих інструментів для автоматизації рутинних задач та відлагодження коду. Мова програмування C# була обрана з огляду на її об'єктно-орієнтовану структуру та високу продуктивність. Ця мова програмування розроблена компанією Microsoft та має широку підтримку від спільноти розробників та бібліотек.

Додатково було використано такі інструменти:

* .NET Framework - це платформа для розробки та виконання програмного забезпечення, яка надає зручний інтерфейс для програмування мовами, такими як C#.
* LINQ (англ. Language Integrated Query - запити, інтегровані в мову) – компонент Microsoft .NET Framework, який додає нативні можливості виконання запитів даних до мов, що входять у .NET[2].
* Git - це розподілена система контролю версій, яка дозволяє зберігати та відстежувати зміни в коді проекту. Git надає можливість повернутися до будь-якої попередньої версії коду, яка була збережена, що дозволяє відновити роботу в разі помилки чи неправильної зміни коду.

Якщо програма розробляється декількома розробниками або навіть компанією, то Git дозволяє розробникам працювати з одним і тим же кодом, зберігаючи кожну зміну як версію. Кожен розробник може працювати зі своєю власною копією коду та об’єднувати зміни з іншими розробниками, що робить роботу над проектом більш організованою та ефективною.

А для того, щоб мати доступ до свого проєкту в будь-якому місці та з будь-якого комп’ютера, або передати код на розгляд або правки іншим розробникам, використовувався хмарний сервіс GitHub для зберігання та управління проектами програмного забезпечення. Для зручного керування версіями та виконання комітів використовувався графічний клієнт GitHub Desktop.

Вибір Visual Studio та мови програмування C# дозволив мені швидко та ефективно створити систему генерації тексту за ключовим словом з високою продуктивністю та широкими можливостями для налагодження та тестування. Користування .NET Framework, LINQ та Git також сприяло покращенню продуктивності та якості розробки проекту.

Висновки до Розділу 2

8-10 листів

# **РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ «Generatext»**

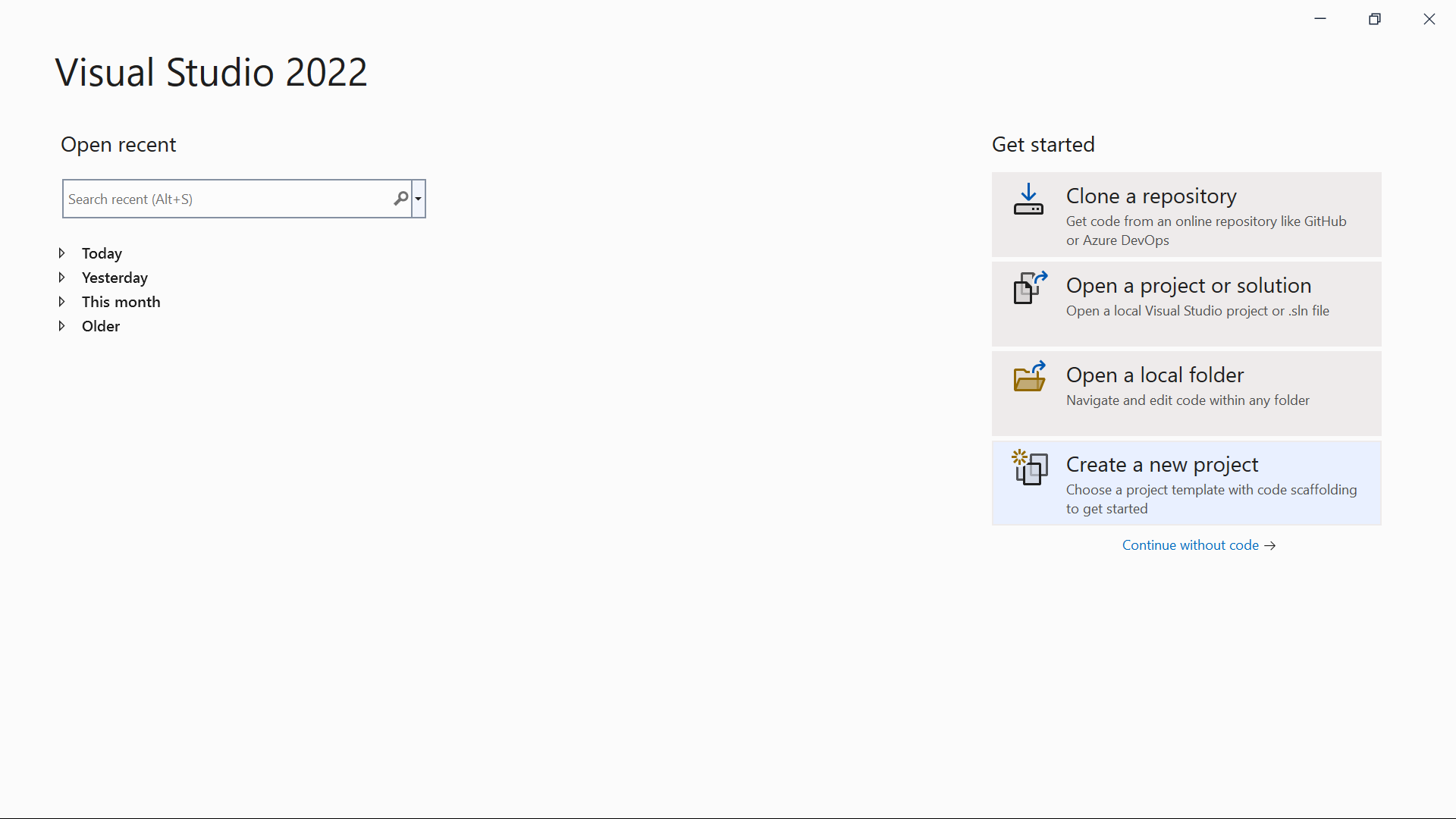
## **3.1 Налаштування віртуального середовища розробки**

Перед тим як почати роботу над проєктом потрібно налаштувати простір, в якому буде створюватись наш програмний додаток. Працювати над своїм проєктом я буду в середовищі розробки, яке створила компанія Microsoft в 1997 році, Microsoft Visual Studio 2022.

Visual Studio - це інтегроване середовище розробки (IDE), яке використовується для розробки програмного забезпечення на платформах Microsoft, таких як Windows, .NET Framework, та ін. Це середовище надає повний спектр інструментів для розробки програмного забезпечення на різних мовах програмування, таких як C, C++, VB.NET, C#, F#, JavaScript та Python. Функціональність VS охоплює всі етапи розробки програм, включаючи написання та редагування коду, управління версіями, проєктування графічного інтерфейсу, збірку, відлагоджування, компіляцію, збірку та тестування. Також Visual Studio має можливості автодоповнення коду (IntelliSense), вирівнювання, підсвічування синтаксису, а також відлагодження програми крок за кроком, щоб виявляти та виправляти помилки. Функціонал Visual Studio можна доповнити за допомогою підключення необхідних розширень.

Для того, щоб почати розробку нашого програмного продукту потрібно виконати такі кроки:

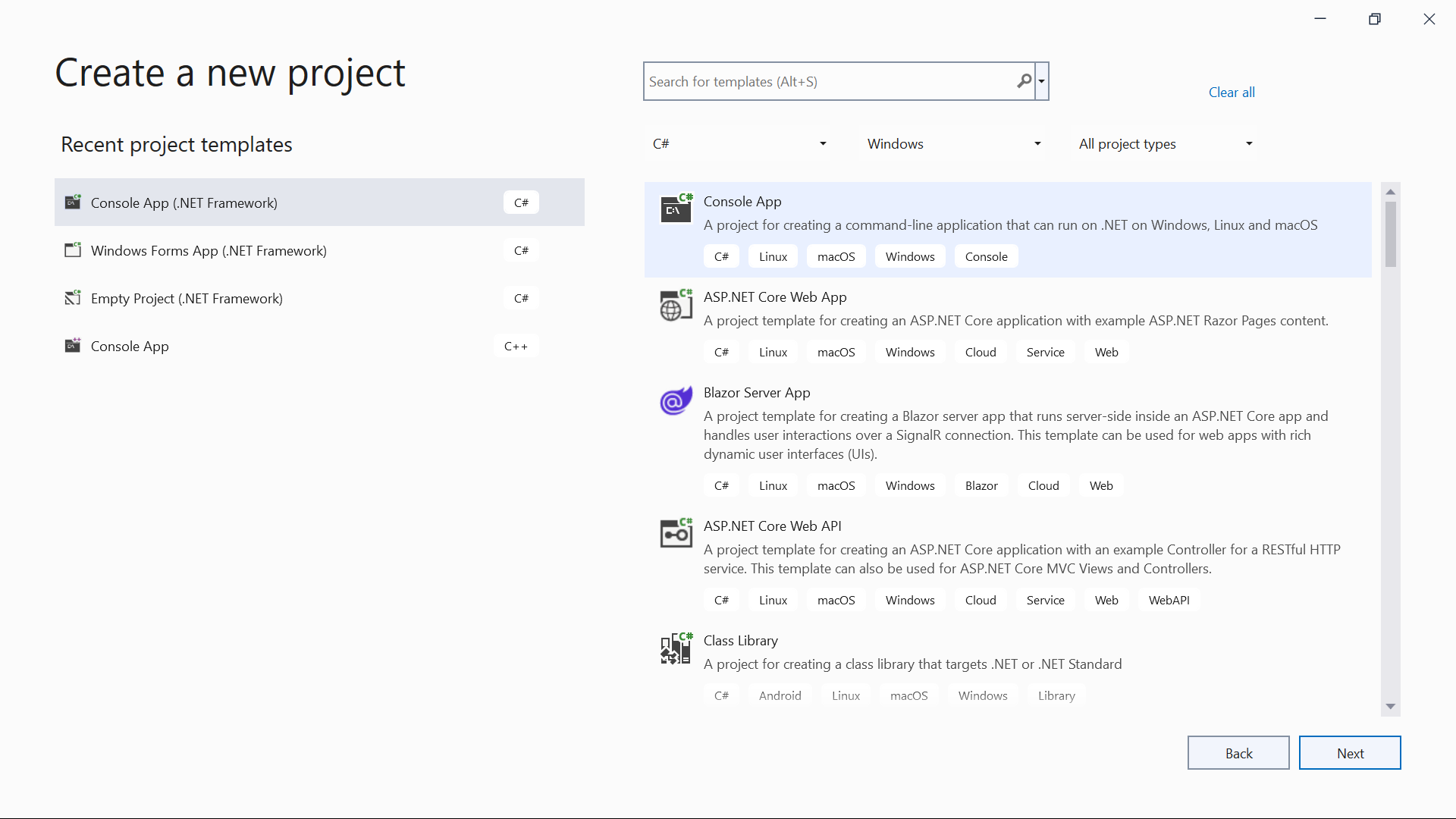
1. Відкрити середовище розробки Microsoft Visual Studio.
2. У вікні, що відкрилося, натиснути «Create a new project» (рис. 3.1).



**Рис. 3.1 Створення нового проєкту**

*Джерело: побудовано автором (знімок з екрану)*

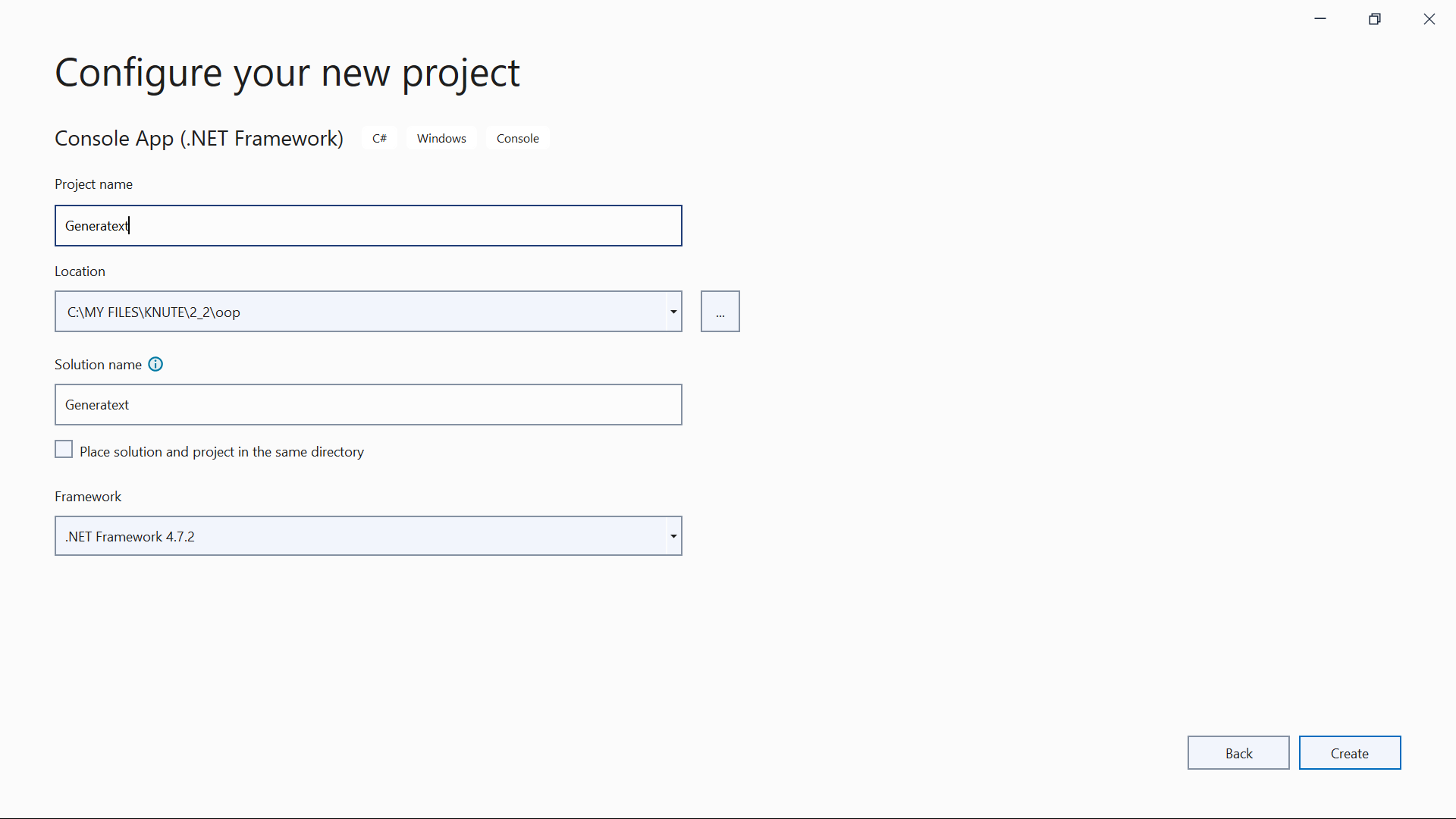
1. Обрати мову програмування C#, після цього знайти необхідний шаблон. В нашому випадку «Console App (.NET Framework)» (рис 3.2).



**Рис. 3.2 Вибір шаблону серед запропонованих Visual Studio**

*Джерело: побудовано автором (знімок з екрану)*

1. Обрати коректне ім’я проєкту, його розташування, ім’я рішення (встановлюється автоматично при розміщенні рішення та проєкту в одній директорії) та версію фреймворка, з якою ми будемо працювати (рис. 3.3).



**Рис. 3.3 Налаштування проєкту**

*Джерело: побудовано автором (знімок з екрану)*

## **3.2 Технології, використані для розробки програмного продукту**

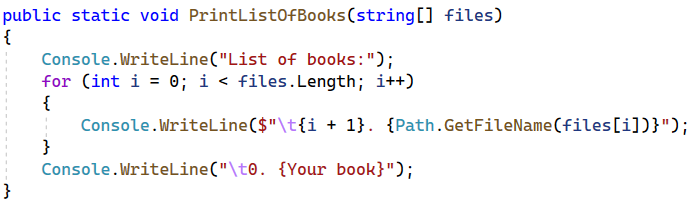
Даний програмний додаток, як вже було сказано, розроблено мовою програмування C# та побудовано із застосуванням шаблону «Console App (.NET Framework)». Шаблони надають користувачам деякий базовий код та структуру, які спрощують початок написання власного коду.

Мій програмний продукт містить 5 класів, серед яких:

1. **BookSelector.cs**

Даний клас містить всі методи, що потрібні під час вибору користувачем книги та введення ним даних, необхідних для генерації тексту. Всі методи оголошені з модифікатором public, бо використовуються в інших класах та викликають інші методи всередині своєї реалізації.

Метод **PrintListOfBooks(string[] files)** друкує в консоль список книг, а якщо точніше – список файлів з вказаної директорії. Користувач з цього списку має вибрати номер файлу (книги, завдяки якій буде відбуватися генерація) або цифру ‘0’ – користувацький файл (рис. 3.4). Після чого метод **BookChoice(string[] files*)*** вже безпосередньо отримує від користувача обраний файл і передає текст з нього в змінну text (рис. 3.5). Далі методи **GetUserBeginningWord()** (рис. 3.6) та **GetUserWordsCount()** (рис. 3.7) обробляють введені користувачем значення ключового (першого) слова та кількості слів у реченні відповідно, після чого вертають їх.



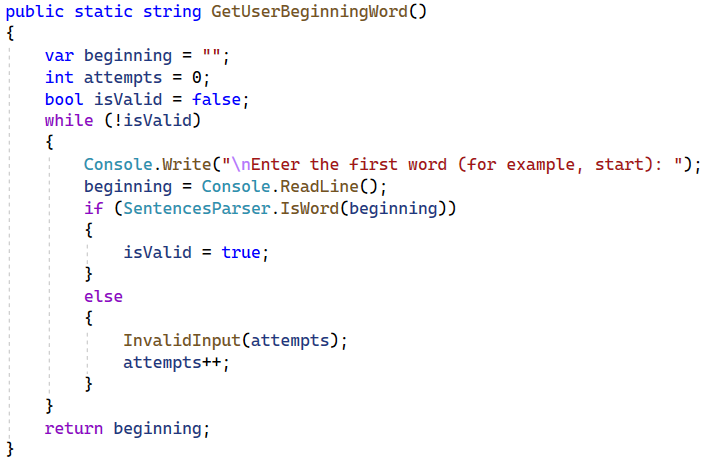
**Рис. 3.4 Виведення списку файлів у консоль**

*Джерело: побудовано автором (знімок з екрану)*



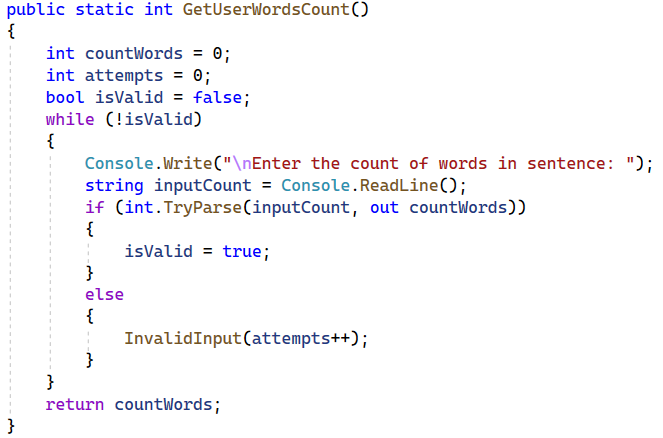
**Рис. 3.5 Обробка вибору файлу користувачем**

*Джерело: побудовано автором (знімок з екрану)*



**Рис. 3.6 Обробка введеного ключового слова**

*Джерело: побудовано автором (знімок з екрану)*

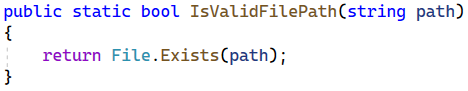


**Рис. 3.7 Обробка введеного значення кількості слів**

*Джерело: побудовано автором (знімок з екрану)*

Щоб уникнути помилок виконання програми, якщо користувач введе недопустиме значення (наприклад літеру замість цифри), в кожному методі, що приймає значення від користувача, виконуються перевірки.

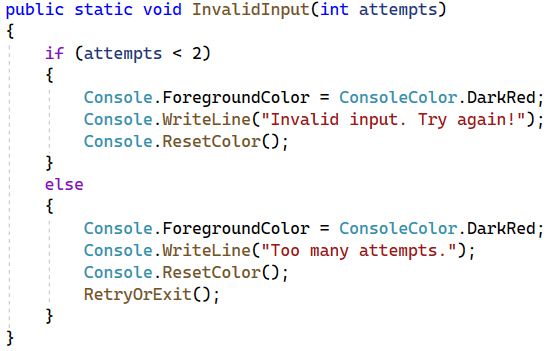
Наприклад для того, щоб перевірити чи правильно було введено шлях до користувацього файлу або чи існує він взагалі, використовується метод **IsValidFilePath(string path)** (рис. 3.8).



**Рис. 3.8 Перевірка існування файлу за введеним шляхом**

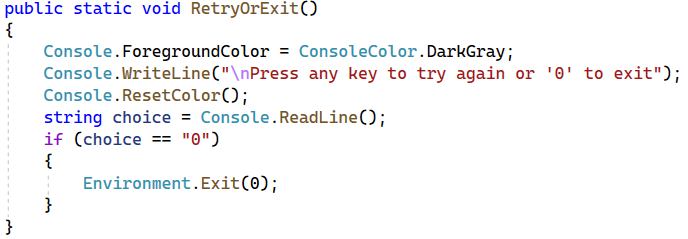
*Джерело: побудовано автором (знімок з екрану)*

Для того, щоб повідомляти користувача, що той ввів недопустиме значення, виконується метод **InvalidInput(int attempts)** (рис. 3.9). Після третьої спроби (лічильником є змінна *attempts*) програма пропонує користувачу вибір: продовжити чи завершити виконання програми (за допомогою метода **RetryOrExit()**) (рис. 3.10).



**Рис. 3.9 Виведення в консоль повідомлень про помилку**

*Джерело: побудовано автором (знімок з екрану)*

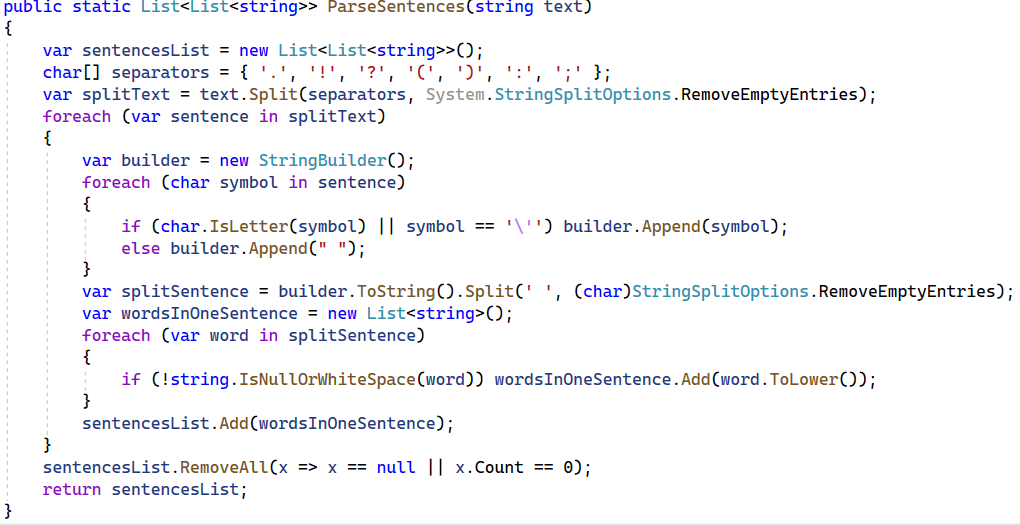


**Рис. 3.10 Вибір щодо продовження виконання програми**

*Джерело: побудовано автором (знімок з екрану)*

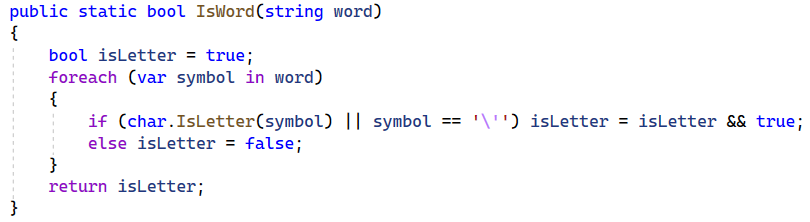
1. **SentencesParser.cs**

Цей клас виконує розбиття тексту, на основі якого буде відбуватися генерація, на речення та слова. Для того, щоб розділити текст на речення, був створений метод **ParseSentences(string text)**, який приймає текст типу *string* з файлу та вертає *List<List<string>>()* (рис. 3.11)*.* В ньому використовується вбудований метод **string.Split** та створений власноруч метод **IsWord(string word)** типу *bool* (рис. 3.12), який перевіряє, чи є символи в реченні буквами. Після перевірки символів використовується рядковий клас ***StringBuilder***, визначений у просторі імен *System.Text*.



**Рис. 3.11 Виконання розбиття тексту**

*Джерело: побудовано автором (знімок з екрану)*



**Рис. 3.12 Перевірка чи є частина розбитого тексту словом**

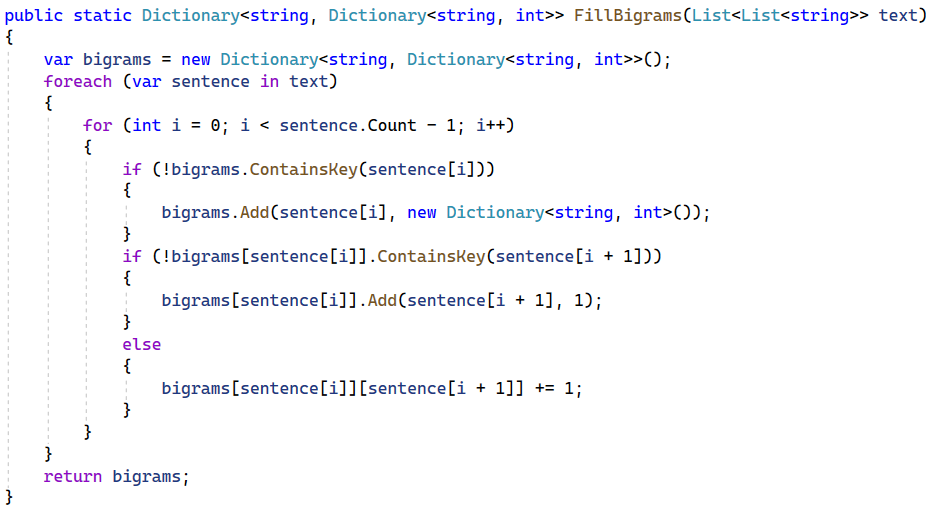
*Джерело: побудовано автором (знімок з екрану)*

1. **FrequencyAnalysis.cs**

В даному класі створюється словник найчастіших продовжень біграм та триграм.

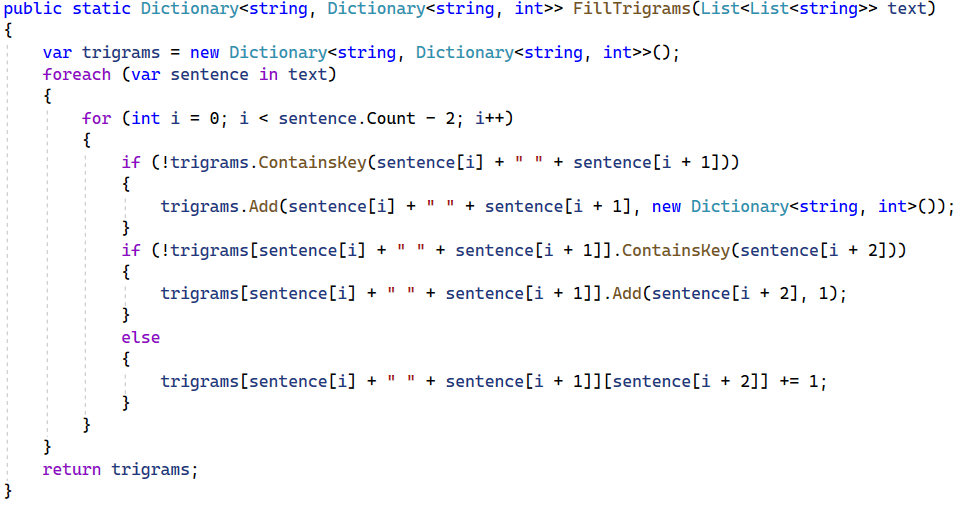
N-грама - це N сусідніх слів в одному реченні. 2-грами називають біграмами, 3-грами – триграмами. Наприклад, з тексту: *"She stood up. Then she left."* можна виділити такі біграми *"she stood"*, *"stood up"*, *"then she"* та *"she left"*, але не *"up then"*. І дві триграми *"she stood up"* та *"then she left"*, але не *"stood up then"*.

Отже, результатом використання цього класу є словник ***result***, ключами якого є всі можливі початки біграм і триграм, а значеннями їх найчастіші продовження. Якщо є кілька продовжень з однаковою частотою, використовуються ті, що лексикографічно менше. Для лексикографічного порівняння був використаний вбудований у .NET спосіб порівняння ***string.CompareOrdinal.***



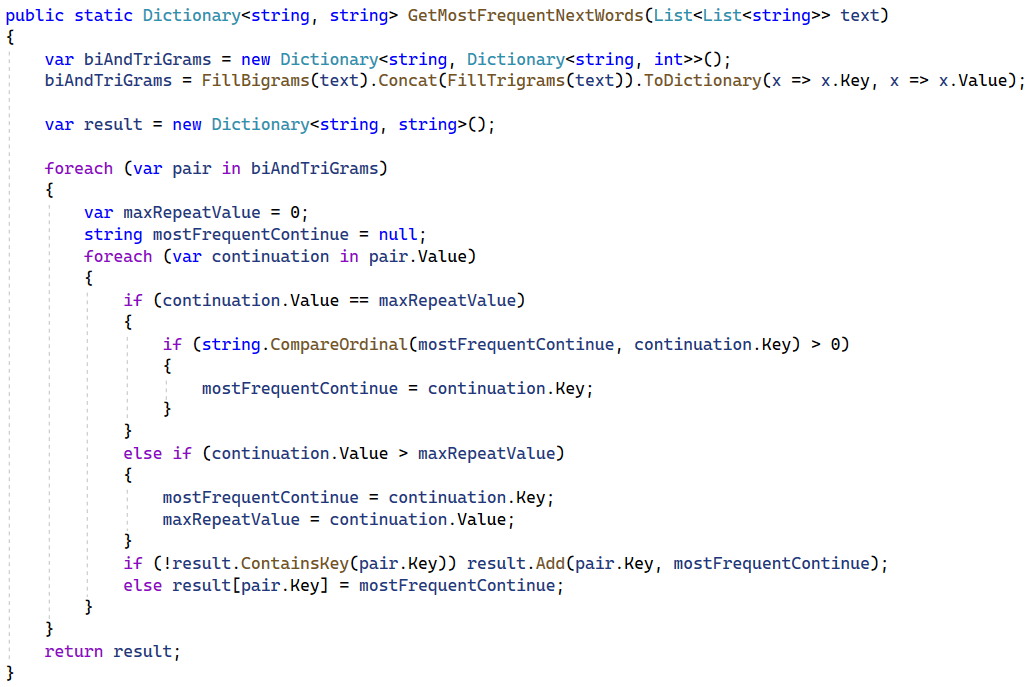
**Рис. 3.13 Створення словнику біграм**

*Джерело: побудовано автором (знімок з екрану)*



**Рис. 3.14 Створення словнику триграм**

*Джерело: побудовано автором (знімок з екрану)*



**Рис. 3.15 Отримання найбільш частого продовження**

*Джерело: побудовано автором (знімок з екрану)*

1. **TextGenerator.cs**

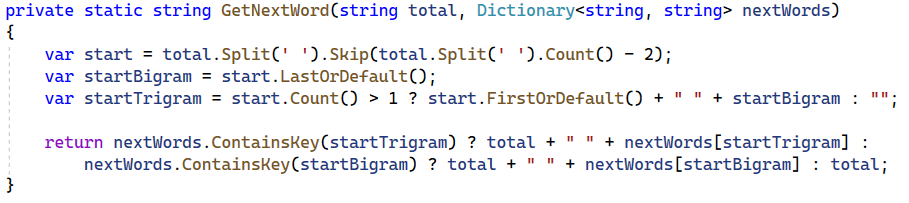
На вхід алгоритму передається словник **NextWords**, отриманий у класі **FrequencyAnalysis.cs**, одне або кілька перших слів фрази **phraseBeginning** і **wordsCount** — кількість слів, які потрібно дописати до початкової фрази.

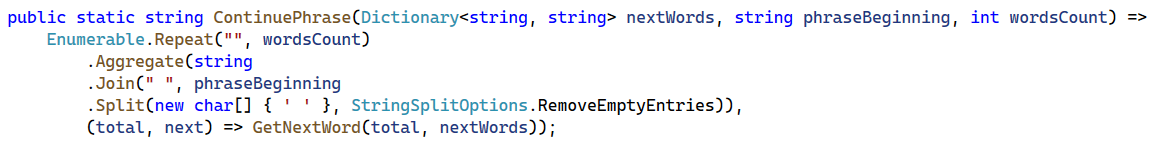
Інакше, якщо у словнику є ключ, що складається з одного останнього слова фрази, то продовжувати потрібно словом, що зберігається у словнику цього ключа.

Інакше потрібно достроково закінчити генерування фрази і повернути згенерований на даний момент результат.

Метод GetNextWord розділяє рядок total на слова, використовуючи пробіл як роздільник, і відбирає останнє слово, яке є початком наступного слова. Якщо у total міститься більше одного слова, метод відбирає останні два слова, які є початком наступного слова.

Далі метод перевіряє, чи містить словник nextWords наступні слова після останнього слова total. Якщо так, метод додає наступне слово в рядок total і повертає його. Якщо словник не містить наступних слів після останнього слова total, метод перевіряє, чи є наступне слово, що може йти після передостаннього слова. Якщо так, метод додає наступне слово в рядок total і повертає його. Якщо і це не можливо, метод повертає рядок total без змін.





1. **Program.cs**

Цей клас містить метод **Main** (рис. 3.18), що є точкою входу до програми. В ньому виконуються початкові налаштування та викликаються інші методи, які виконують основну роботу додатку.



## **Висновок до розділу 3. Чи можна висновком блок-схему?**

Для того, щоб наглядно показати структуру коду та зрозуміти логіку роботи програми, було побудовано блок-схему.

[БЛОК-СХЕМА]

# **ВИСНОВОК**

………………………………..

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. <https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%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>
2. <https://www.wiki-data.uk-ua.nina.az/LINQ.html>