МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
автоматизованих систем обробки інформації та управління

КУРСОВА РОБОТА

з «Основи програмування -2. Основи об'єктно-орієнтованого програмування»

на тему:«Робот-редактор»

Студента I курсу, групи ІП-81
Спеціальності 121 «Інженерія програмного забезпечення»
Петруняка Дмитра Васильовича

Керівник Старший викладач кафедри АСОІУ
Головченко М.М.

Кількість балів: _____
Національна оцінка: _____

Члени комісії

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2019 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-81 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Петруняка Дмитра Васильовича

(прізвище, ім'я, по батькові)

1. Тема роботи Робот-редактор

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи Технічне завдання (додаток А)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
Постановка задачі, аналіз предметної області, опис архітектури програмної системи, опис програмного забезпечення, результати його тестування, інструкція користувачу

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи		
2.	Підготовка ТЗ		
3.	Пошук та вивчення літератури з питань курсової роботи		
4.	Розробка сценарію роботи програми		
6.	Узгодження сценарію роботи програми з керівником		
5.	Розробка (вибір) алгоритму рішення задачі		
6.	Узгодження алгоритму з керівником		
7.	Узгодження з керівником інтерфейсу користувача		
8.	Розробка програмного забезпечення		
9.	Налагодження розрахункової частини програми		
10.	Розробка та налагодження інтерфейсної частини програми		
11.	Узгодження з керівником набору тестів для контрольного прикладу		
12.	Тестування програми		
13.	Підготовка пояснювальної записки		
14.	Здача курсової роботи на перевірку		
15.	Захист курсової роботи		

Студент _____

(підпис)

Керівник _____ Головченко М.М.

(підпис) (прізвище, ім'я, по батькові)

"___" _____20__ р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 45 сторінок, 13 рисунків, 3 таблиці, 5 посилань.

Об’єкт дослідження: задача обробки текстової інформації

Мета роботи: дослідження методів обробки тексту, а саме видалення певних рядків, заміна певних рядків на інший текст, додання рядка в певну позицію, заміна тексту та створення програмного забезпечення для реалізації цих методів.

Дана курсова робота включає в себе: опис методів, застосування методу до конкретного завдання, код програми вирішення перерахованих вище методів на мові програмування С++, а також описання детального процесу розв’язання кожного з них.

РЕДАГУВАННЯ ТЕКСТУ, ВИДАЛЕННЯ РЯДКІВ, ЗАМІНА РЯДКІВ, ДОДАВАННЯ РЯДКА, ЗАМІНА ТЕКСТУ.

ЗМІСТ

ВСТУП

Кожен користувач комп'ютера зустрічається з необхідністю підготовки, редагування, тієї чи іншої текстової інформації. При колишніх методах роботи на друкарських машинках виникала велика кількість проблем, з обробки текстової інформації, так як при роздруківці тексту змінювався зовнішній вигляд з'являлися друкарські помилки, спотворення, потрібен додатковий час для набору тексту повторно.

При використанні персонального комп'ютера для підготовки документів текст редагованого документа виводиться на екран, і користувач може в діалоговому режимі вносити в нього свої зміни. Також у користувача з'являються великі можливості з редагування тексту.

Ефективність застосування комп'ютерів для підготовки текстів призвела до створення безлічі прикладних програм для обробки документів. Такі програми поділяють на текстові редактори та текстові процесори.

Текстові редактори - це найпростіші програми (типу Блокнота). Вони дозволяють вводити, редагувати та друкувати текст. Текстові процесори (типу Microsoft Word) мають більш розвинені функції - зокрема, перевірка орфографії, складне форматування, використання елементів автоматизації, таких як автоматична нумерація абзаців, рядків, сторінок, таблиць, малюнків, автоматичну збірку змістів, списків ілюстрацій та ін. Все це можна автоматизувати за допомогою макрокоманд.

В цій курсовій роботі мною було розглянуто та проаналізовано найбільш вживані функції програм для роботи з текстом та на основі них створено власний текстовий редактор з такими основними функціями: видалення та заміна рядка, вставлення рядка та заміна тексту. Окрім цього розроблено графічний інтерфейс, який спрощує користування програмою.

1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде редагувати текст, введений користувачем в одному з файлів. Інший файл – ‘‘рецепт’’, в який можна ввести такі команди:

- видалення та заміна рядка;
- вставлення рядка;
- заміна тексту.

Вхідні дані два файли:

- перший призначений для тексту, що потрібно відредагувати
- другий – ‘‘рецепт’’ (тобто дії які потрібно виконати над текстом).

Вихідні дані:

Файл, що містить відредагований текст.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Під час аналізу предметної галузі були розглянуті існуючі проблеми та розроблена концептуальна модель.

Предметна область: обробка тексту.

Текст – це один із засобів представлення інформації, який включає в себе можливість читати та змінювати його;визначати ключові слова; зберігати достовірну інформацію довгий час.

Обробка тексту - вся сукупність операцій (введення, форматування, редагування, зберігання та знищення), що здійснюються за допомогою технічних і програмних засобів.

Редагування тексту це набір операцій які ми можемо виконати над ним. Наприклад: видалення, замінна, часткова заміна та додавання елементів тексту(літер, слів, речень, рядків).Зміна зовнішнього вигляду тексту(шрифту, розміру, додаткових позначень)

Незважаючи на широкі можливості використання комп'ютерів для обробки найрізноманітнішої тексту, найпопулярнішими як і раніше залишаються програми, призначені для роботи з текстом.

Отож, в даній предметній області:

- Об’єкт – текст
- Суб’єкт – людина, яка намагається відредагувати текст;

Для початку дізнаємся що кожний з термінів означає.

Введення тексту – перетворення тексту з будь-якого виду в письмовий, з подальшими можливостями використання всіх переваг цього виду.

Форматування - це процес видалення помилок, доповнення та зміни тексту для покращення його сприйняття.

Зберігання тексту – процес, під час його інформація яку користувач обробляє, переноситься з оперативної пам’яті на жосткий диск комп’ютера.

Видалення – операція під час якої здійснюється прибирання зайвих елементів тексту.

Заміна – це операція, під час якої відбувається не тільки видалення зайвих елементів тексту, але і на їх місце вставляються нові елементи.

Шрифт - графічний малюнок накреслень літер і знаків, які складають єдину стилістичну та композиційну систему, набір символів визначеного розміру і малюнка.

Рядок - в тексті, це кілька слів, літер або інших знаків, написаних чи надрукованих в одну лінію.

Беручи до уваги все вищезгадане, можемо зрозуміти, що основні функціональні вимоги до розроблюваної системи це:

- Можливість видаляти рядки від MM до NN.
- Можливість замінити рядки з MM до NN на текст TEXT.
- Вставляти TEXT після рядка MM.
- Замінити в рядках від MM до NN текст TEXT на TEXT.
- Скасувати дію попередньої команди
- Робота з графічним інтерфейсом

На рисунку 2.1 зображено діаграму прецедентів.

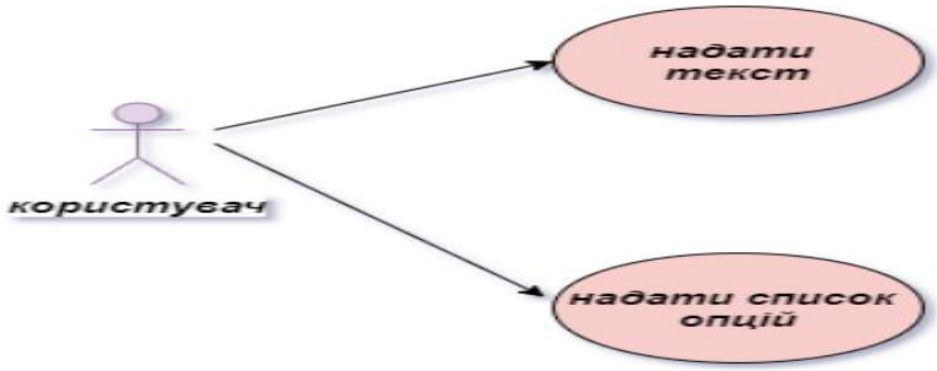

Рисунок 2.1 – Діаграма прецедентів

Сценарій поведінки системи:

- Користувач надає текст, який потрібно обробити;
- Користувач надає список операцій

Для опису загального алгоритму [1] представимо змінні які ми будемо використовувати, у вигляді таблиці 2.1

Таблиця 2.1 – Основні змінні та їх призначення

Змінна	Призначення
Text	Текст, який потрібно відредагувати
que	Список команд, які потрібно виконати
pathText	Шлях до файлу, який потрібно відредагувати
pathRecipe	Шлях до файлу-рецепту, в якому зберігається список команд що потрібно виконати

а) Загальний алгоритм

1) ПОЧАТОК;
2) Зчитування шляху до файлу у змінну pathText, що потрібно відредагувати;
3) Зчитування шляху до файлу-рецепту у змінну pathRecipe;
4) ЯКЩО шлях до файлу порожній, ТО вивести помилку і перейти до пункту 11;
5) ЯКЩО шлях до файлу-рецепт поро порожній, ТО вивести помилку і перейти до пункту 11;
6) Зчитати файл, що потрібно відредагувати:
   I. ПОКИ не дістанемся кінця файла, будемо зчитувати по рядку і записувати в Text;
7) Зчитати файл-рецепт:
   I. ПОКИ не дістанемся кінця файла, будемо зчитувати по рядку і записувати в que;
8) Цикл проходу всіх елементів que(a_i − поточний елемент): ЯКЩО a_i дорівнює “undo”, ТО видаляєм попередній елемент;
9) ПОКИ que не порожнє:
   I. Дістаєм перший елемент:
   II. ЯКЩО він дорівнює “delete” викликаєм б;
   III. ЯКЩО він дорівнює “change” викликаєм в;
   IV. ЯКЩО він дорівнює “insert” викликаєм г;
   V. ЯКЩО він дорівнює “replace” викликаєм д;
10) Викликати вікно про успішне завершення програми;
11) КІНЕЦЬ;

б) Алгоритм методу delete:

1) З рядка que[i] отримати значення з якого рядка (MM) і до якого рядка(NN) видаляти;
2) ЯКЩО не вдало прочитати MM або NN, ТО вивести вікно про неправильно введені дані;
3) ЯКЩО MM< 1, ТО MM = 1;
4) ЯКЩО NN>size(Text), ТО NN=size(Text);
5) Видалити рядки з MM до NN (vector.erase());
6) КІНЕЦЬ;

в) Алгоритм методу change:

1) З рядка que[i] отримати значення з якого рядка (MM) і до якого рядка(NN) заміняти, та TEXT на який заміняти;
2) ЯКЩО не вдало прочитати MM або NN, ТО вивести вікно про неправильно введені дані;
3) ЯКЩО MM< 1, ТО MM = 1;
4) ЯКЩО NN>size(Text), ТО NN=size(Text);
5) ЯКЩО (NN-MM) > size(TEXT):
   I. Викликати алгоритм delete (пункт б)) для рядків від (MM + size(TEXT)-1) до (NN -1);
   II. Цикл проходу по всіх елементах TEXT: Text[i+NN] = TEXT[i];
6) ІНАКШЕ:
   I. Цикл проходу від MM до NN: Text[i] = TEXT[i];
   II. Цикл проходу від NN до size(TEXT): вставлення TEXT[k] після k;
7) КІНЕЦЬ;

г) Алгоритм методу insert:

1) З рядка que[i] отримати значення після якого рядка (MM) вставляти текст, та TEXT на який потрібно вставити;
2) ЯКЩО не вдало прочитати MM, ТО вивести вікно про неправильно введені дані;

3) ЯКЩО MM< 1, ТО MM = 1;
4) ЯКЩО MM>size(Text), ТО MM=size(Text);
5) Цикл проходу від i=0 до size(TEXT): вставити TEXT[i] після (MM+k), k++;

д) Алгоритм методу replace:

1) З рядка que[i] отримати значення від якого рядка (MM) і до якого (NN) заміняти текст, TEXT1 який потрібно замінити та TEXT2, на який потрібно замінити;
2) ЯКЩО не вдало прочитати MM або NN ТО вивести вікно про неправильно введені дані;
3) ЯКЩО MM< 1, ТО MM = 1;
4) ЯКЩО MM>size(Text), ТО MM=size(Text);
5) ЯКЩО size(TEXT1) == 1 то:
    I. Цикл проходу від i=MM-1 до NN:
        i. ПОКИ в рядках від MM до NN буде TEXT1:
            1. Виконати заміну в Text[i] на TEXT[0];
            2. ЯКЩО size(TEXT2 > 1):
                01. Видаляєм рядки від i до size(Text)
                02. Цикл проходу від j=1 до size(TEXT2): після j вставляєм TEXT2[j]
6) ІНАКШЕ:
    I. Перевірити чи існує в Text підрядок TEXT1,ЯКЩО не існує, то перейти до пункт 7, ІНАКШЕ:
        i. ЯКЩО size(TEXT2) > size(TEXT1):
            1. Цикл проходу від k=0 до k< size(TEXT1):
                01. ЯКЩО size(Text[k]) < size(TEXT2), ТО ітеруємся від j = k до size(Text[i]): Text[j+k][j] = TEXT2[k][j];
                02. ІНАКШЕ, ітеруємся від j =0 до size(TEXT2): Text[j+k][j] = TEXT2[k][j];

2. Цикл проходу від y = 0 до size(TEXT2): вставляєм після (k+y) рядок TEXT2[k+y];

ii. ІНАКШЕ видаляєм рядки від (k) до (k + (size(TEXT1) – size(TEXT2)));

7) КІНЕЦЬ.

3 ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

Для представлення архітектури програмного забезпечення, що буде вирішувати поставлену задачу, я в даній курсовій роботі вирішив скористатися UML-діаграмами [2].

На рисунку 3.1 представлено діаграму класів.

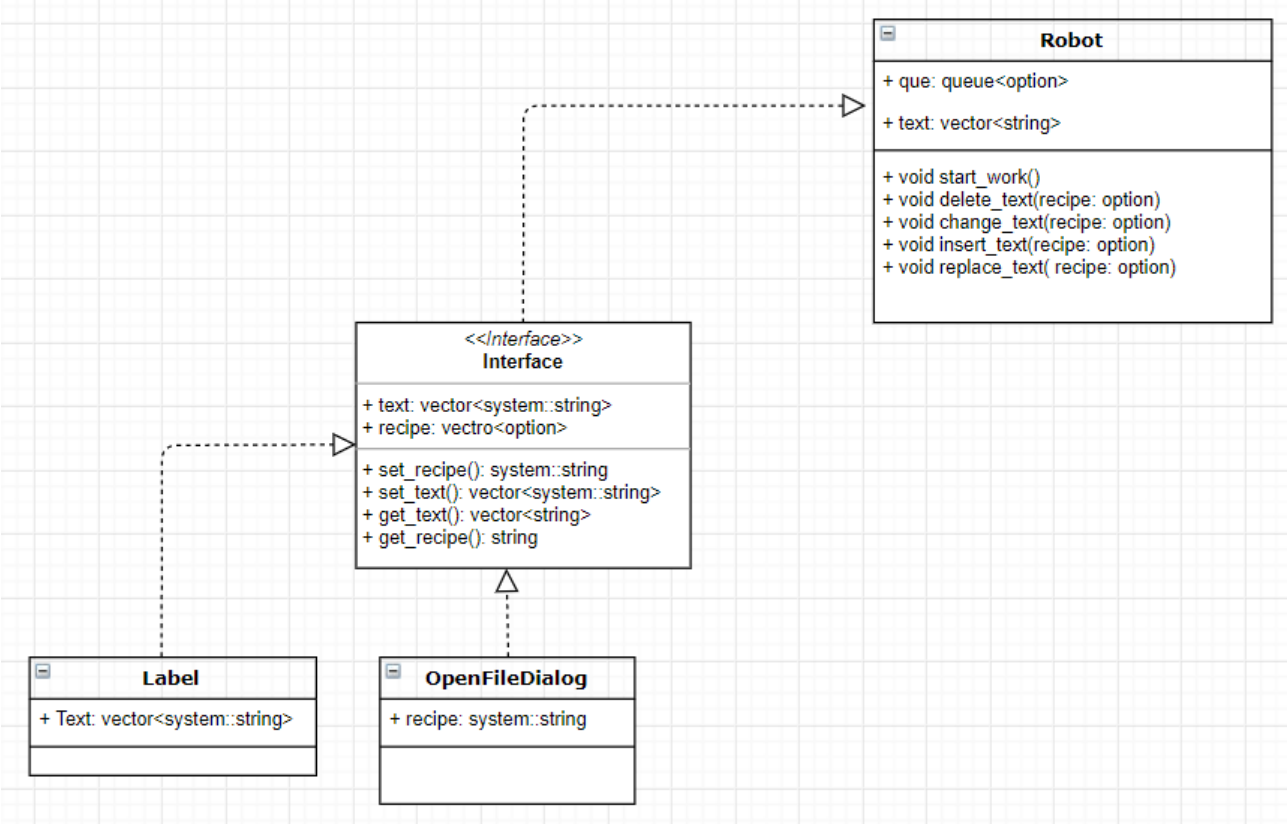

Рисунок 3.1- Діаграма класів.

Поглянувши на діаграму ми можемо зрозуміти, що основний клас – це клас Robot з полями que та text, в яких відповідно зберігаються список команд, що потрібно виконати та текст, який потрібно відредагувати. Дані для цих полів беруться з методів класу Interface set_recipe для que та set_text для text.

Поле que має тип queue<option> це означає що ми використовуємо клас, реалізований в бібліотеці STL, який працює надає можливість використовувати функціонал черги, як структури даних за принципом FIFO (перший зайшов – перший вийшов).

Поле text має тип vector<string> це означає що ми використовуємо клас, реалізований в якості динамічного масиву, змінного розміру. В ньому елементи зберігаються неперервно, отже це дає можливість звертатися до них не тільки під час ітерації а й за допомогою зміщень, що додаються до вказівників на елементи.

Клас Interface – це клас в якому зібрані поля та методи для взаємодії користувача з системою.

Як бачимо клас Interface отримує дані з класу Label і OpenFileDialog. Ці класи отримують уже інформацію напряму від користувача.

Клас option створений тільки для зручного зберігання інформації, та не несе в собі ніяких взаємодій між класами.

Варто зазначити, що оскільки клас Label та OpenFileDialog це класи створені за допомогою Windows Forms, то вони мають багато полів і методів. В даній діаграмі класів я зазначив тільки декілька, зараз економії ресурсів, оскільки інші поля або впливають тільки на графічний вигляд або не використовуються мною в даній курсовій роботі.

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Структура програми зображена на рисунку 4.1

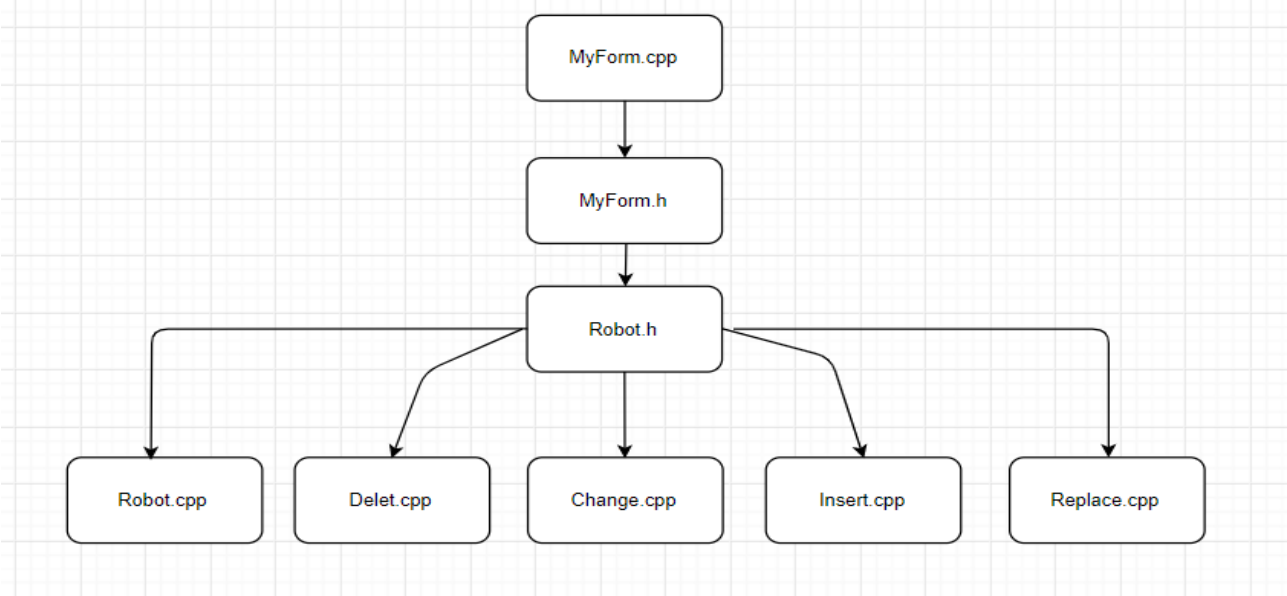

Рисунок 4.1 – Структура програми

В ході виконання поставленого завдання було створено наступні модулі та бібліотеки [3]:

a) MyForm.h – реалізує інтерфейс користувача
b) Robot.h – реалізує клас Robot, і його методи:
    1) Видалення вказаних рядків
    2) Заміна вказаних рядків на TEXT (під TEXT розуміється текст введений користувачем)
    3) Вставка рядка після вказаного номера
    4) Заміна елементів тексту на TEXT

Опис класів та їх методів представлені у таблиці 4.1

Таблиця 4.1 – Опис класів та їх методів

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	Robot	void start_work	Ініціалізація списку команд	Шляхи до вхідних файлів	Черга заповнена командами	Robot.h
2	Robot	vector<string> Open_Text	Читання тексту з файлу	Шлях до тексту	Vector<string> заповнений текстом	Robot.h

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
3	Robot	queue<option> Open_Recipe	Читання списку команд з файлу	Шлях до файлу-рецепту	Vector<string> заповнений командами	Robot.h
4	Robot	void delete_text	Видалення рядків	Текст та рядок-рецепт	Відредагований текст	Robot.h
5	Robot	void parce_for_delete	Обробка рядка-рецепта	Рядок-рецепт	Номери з якого і до якого рядка потрібно провести видалення	Robot.h
6	Robot	void insert_text	Вставлення рядка	Текст та рядок-рецепт	Відредагований текст	Robot.h
7	Robot	void parce_for_insert	Обробка рядка-рецепта	Рядок-рецепт	Номер після якого потрібно вставити, TEXT, який потрібно вставити	Robot.h
8	Robot	void change_text	Заміна рядків	Текст та рядок-рецепт	Відредагований текст	Robot.h
9	Robot	void parce_for_change	Обробка рядка-рецепта	Рядок-рецепт	Номери з якого і до якого рядка провести заміну та TEXT- на який потрібно поміняти	Robot.h
10	Robot	void replace_text	Заміна елементів рядка	Текст та рядок-рецепт	Відредагований текст	Robot.h
11	Robot	void parce_for_replace	Обробка рядка-рецепт	Рядок-рецепт	Номери з якого і до якого провести заміну, TEXT1, який міняти, TEXT2 на який міняти	Robot.h

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних даних	Заголовочний файл
12	string	stoi	Перетворення рядка в число	числа в рядку	число	<string>
13	string	stirng.size	Довжина рядка	-	кількість елементів рядку	<string>
14	string	getline	Читати рядки з файлу	текствовий файл	Рядок типу string	<string>
15	fstream	ofstream	Потоковий запис у файл	шлях для запису файлу	-	<fstream>
16	fstream	ifstream	Потокове зчитування файлу	шлях для читання файлу	-	<fstream>
17	vector	push_back	Додання елемента у кінець вектора	елемент, який потрібно додати	-	<vector>
18	vector	erase	Видалення елемента з вектора	Діапазон в якому потрібно видалити елементи	-	<vector>
19	vector	insert	Вставлення елемента в середину вектора	Номер після якого потрібно вставити елемент	-	<vector>
20	sstream	stringstream	Перетворення string в потік	Рядок типу string	потік, який містить ті ж дані, що і string	<sstream>
21	queue	queue	Структура даних типу FIFO	-	-	<queue>

5 ТЕСТУВАННЯ ПРОГРАМИ

Усі можливі випадки виникнення помилок у програмі залежать від вхідних даних, тобто тої інформації, що отримується від користувача. Тому тестування програми полягає у виявленні правильності та коректності обробки програмою різних вхідних даних.

На створене програмне забезпечення користувач має значний вплив [4], адже він вводить як текст, який повинен оброблятися, так і команди які повинні виконуватися, причому все вручну, що збільшує ймовірність неправильно введених даних.

Для вирішення відповідних помилкових ситуацій слід вдосконалити алгоритми роботи програми та обробити всі виключні ситуації.

Далі слід упевнитись, що усі методи програми коректо працюють на усіх наборах вхідних значень.

Приклади тестування

а) Введення некоректних даних:

Створена програма працює таким чином, що при введені завеликого чи замалого номера рядка, програма автоматично переходить на максимальне й мінімально можливе значення, відповідно. В цьому можна переконатися за допомогою рисунку 5.1 – показаний вхідний текст, рисунку 5.2 – показана команда, яку потрібно виконати, рисунку 5.3 – результат виконання.

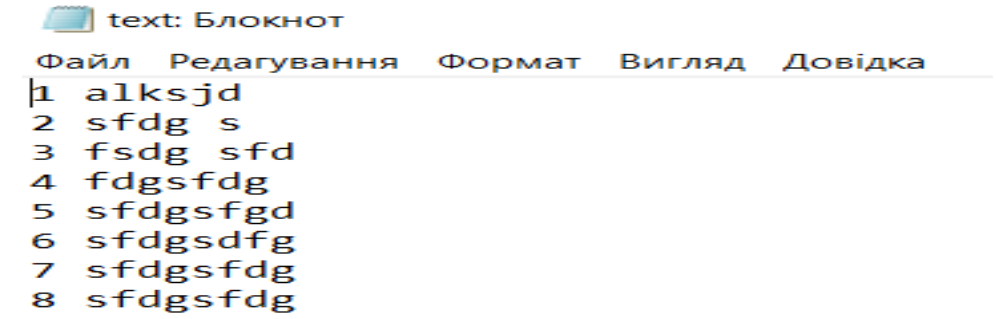

Рисунок 5.1 – Вхідний текст

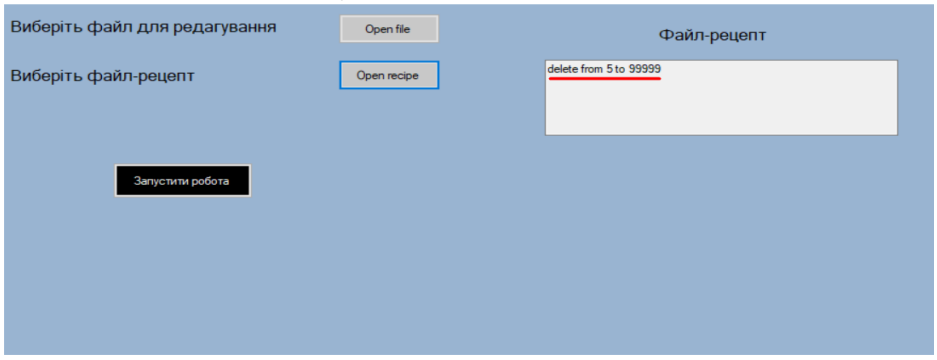

Рисунок 5.2 – Вхідна команда

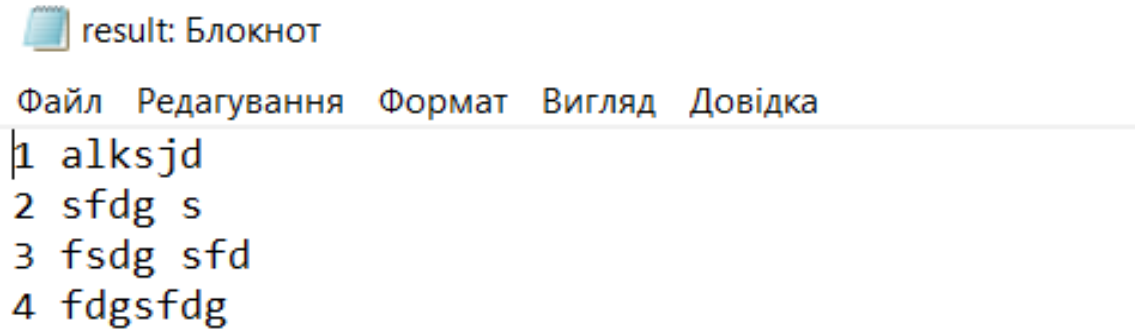

Рисунок 5.3 – Результат виконання команди

б) Тестування програми при невказаних шляхах до файлів:

Якщо під час роботи з програмою користувач не вибере шляхи до файлів, які вказані як вхідні данні, то появиться вікно помилку, і користувачу надасться змога вибрати шлях ще раз, і так до тих пір поки користувач не вибере шлях або не завершить програму. Це можна побачити на рисунку 5.4.

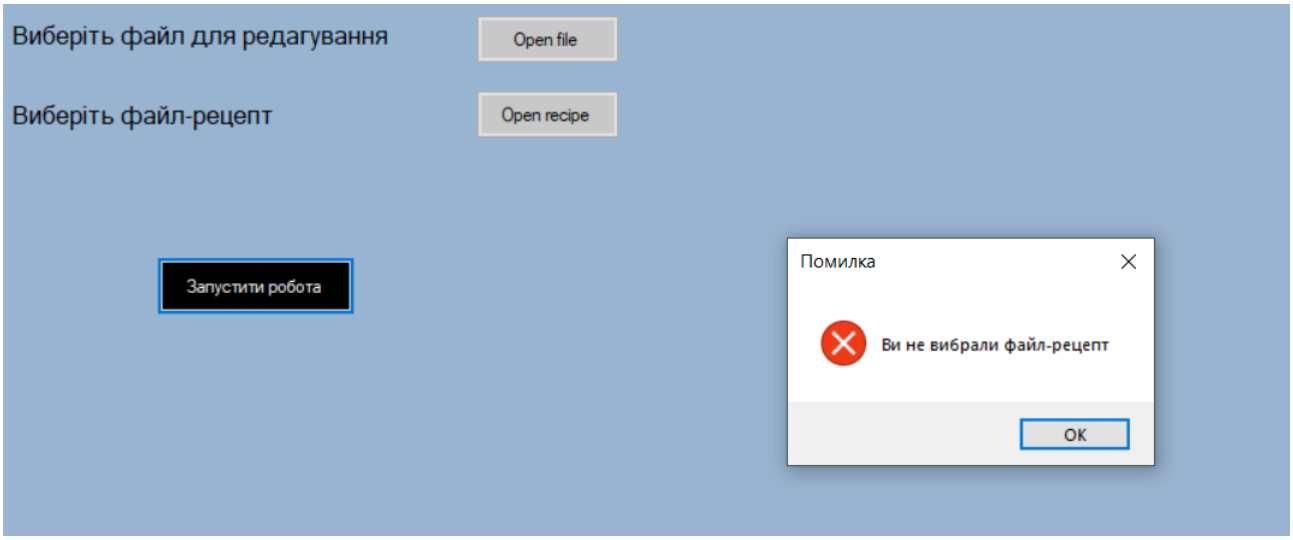

Рисунок 5.4 – Помилка, користувач не вибрав файли.

в) Введення неправильних даних:

Наприклад якщо число, яке вказує після якого рядка потрібно проводити видалення, буде зовсім не числом (наприклад 2A5). Під час обробки таких даних програма покаже вікно помилки та припинить роботу. На рисунку 5.5 можемо побачити приклад неправильно введених даних. На рисунку 5.6 – результат обробки таких даних

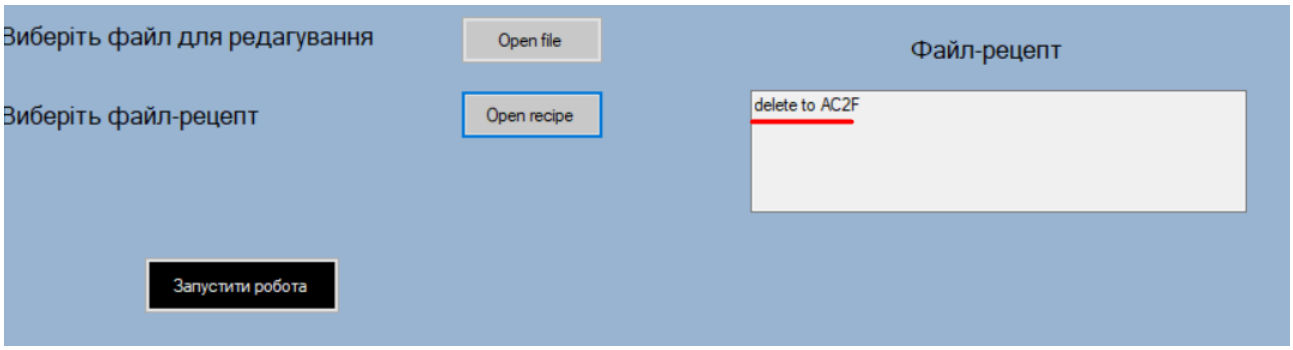

Рисунок 5.5 – Неправильно введене число

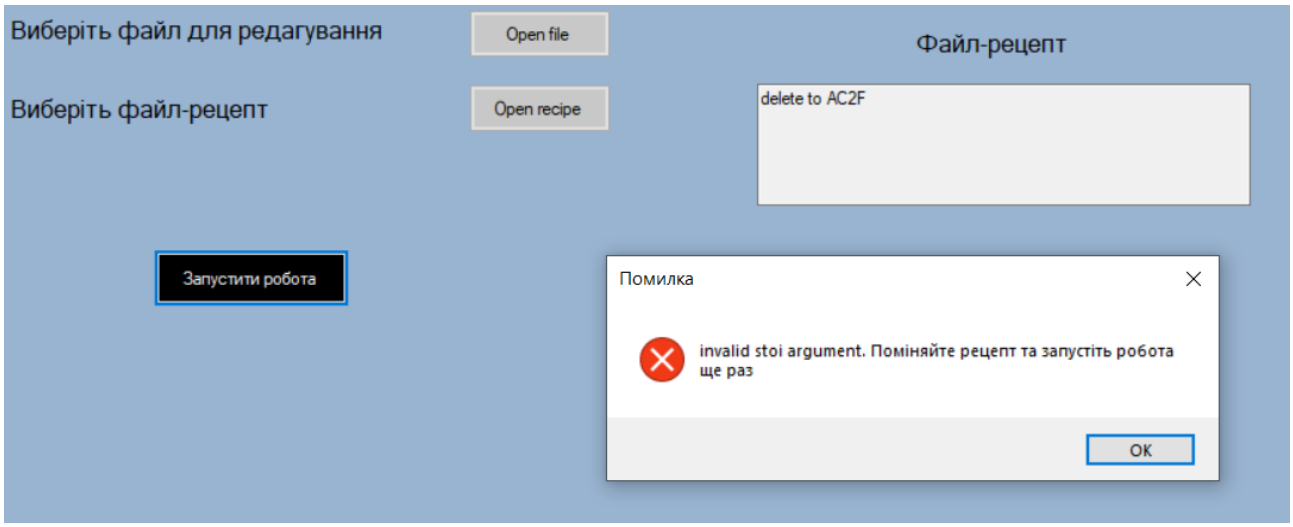

Рисунок 5.6 – Результат обробки неправильних даних

г) Тестування всіх методів програми:

Для цього тесту візьмем файл-рецепт в якому вказані всі можливі команди, його ми можемо побачити на рисунку 5.7, та запустимо для тексту, вказаного на рисунку 5.1. Результат можемо побачити на рисунку 5.8. І справді рядок з 6 до 999 (тобто до кінця) видалені. Рядки з -1(тобто з початку) до 1 рядка замінені на “TEST TEST TEST”. Після 5 рядка вставлено текст “INSERT”. А команда delete(повинні були видалитися всі рядки) не виконалась, бо її скасувала команда undo.

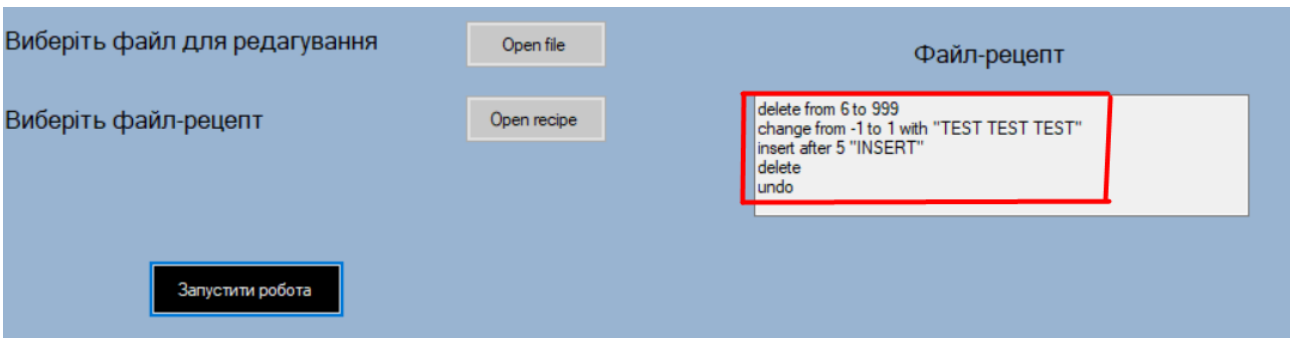

Рисунок 5.7 – Набір всіх можливих команд

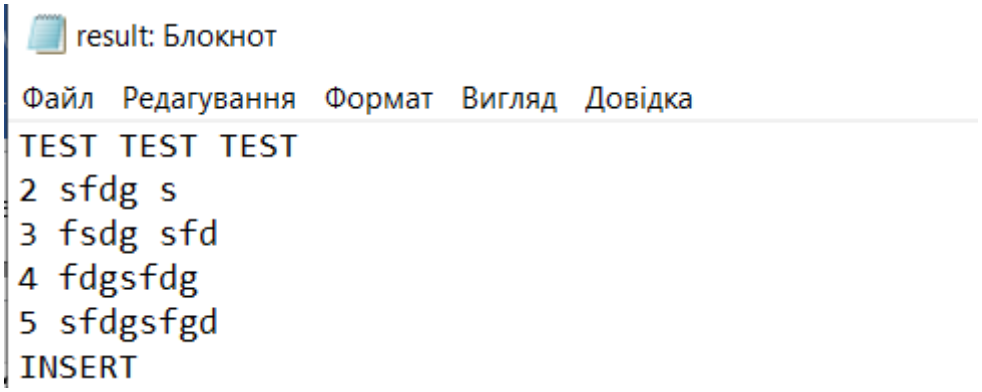

Рисунок 5.8 – Результат виконання програми

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

а) Кроки для успішної роботи програми:

1) Натиснути кнопку Open file. На рисунку 6.1 вона підсвічена;

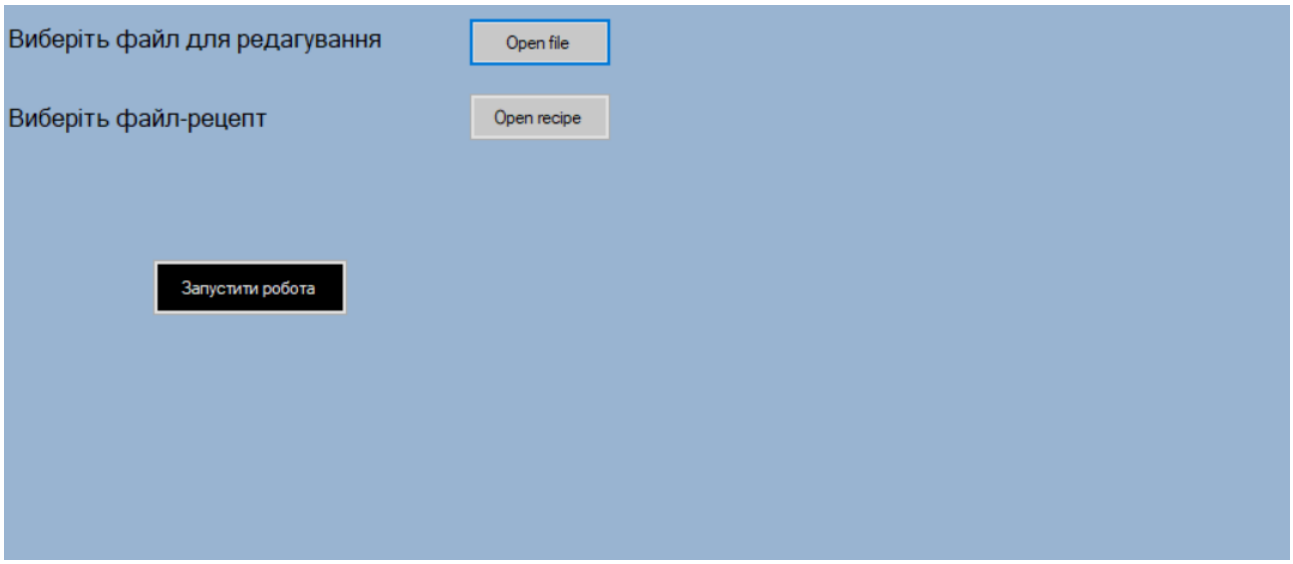

Рисунок 6.1 – Кнопка Open file

2) Вибрати файл, та натиснути Відкрити, рисунок 6.2;

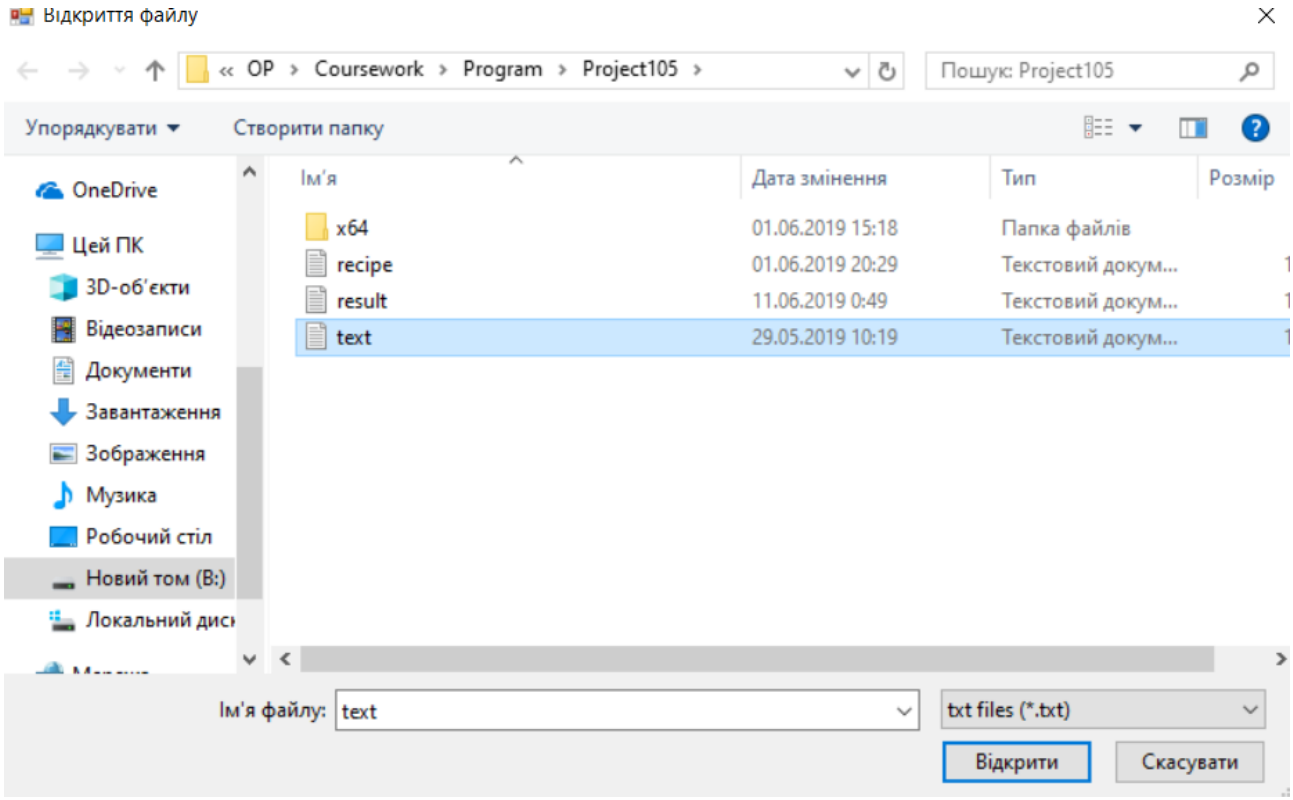

Рисунок 6.2 – Вибір файлу

3) Повторити дії 1) і 2) для кнопки Open recipe, тільки вибрати файл-рецепт;

4) Натиснути кнопку “Запустити робота” і очікувати появлення вікна з повідомленням про успішне виконання редагування, рисунок 6.3;

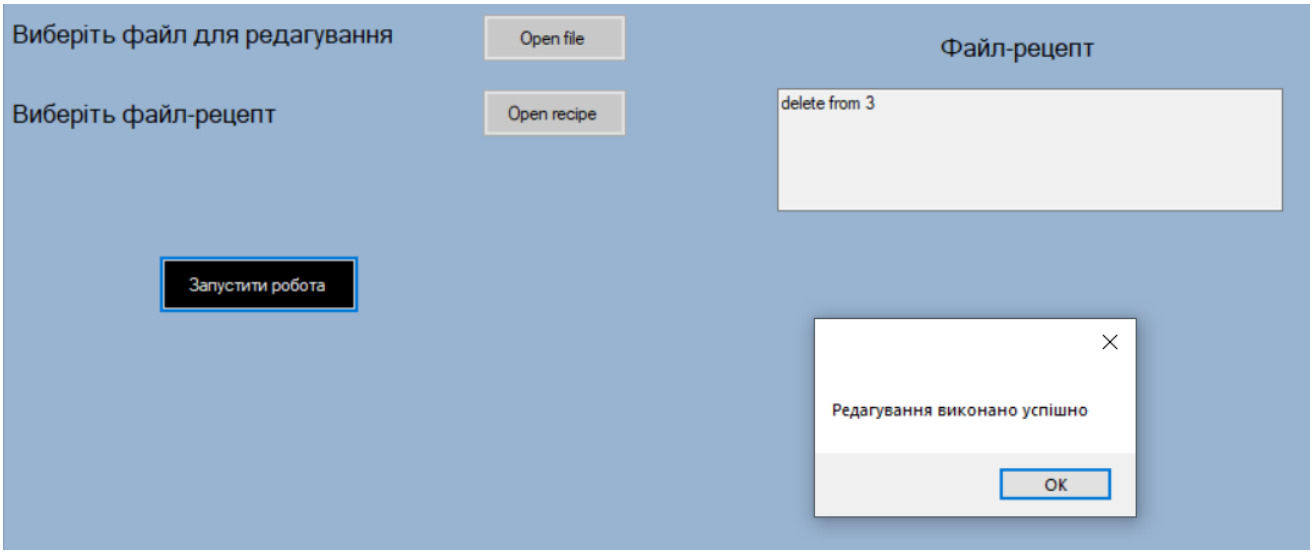

Рисунок 6.3 – Успішне завершення програми

б) Формат вхідних та вихідних даних:

1) На вхід подається два файли. В першому знаходить текст який потрібно відредагувати. В другому команди які потрібно виконати. Команди подаються в наступному вигляді:
   I. delete from MM to NN – видалити рядки з номерами MM..NN. Секції from MM і to NN можуть опускатися, в цьому випадку мається на увазі, що видаляти потрібно з початку або до кінця файлу.
   II. change from MM to NN with “TEXT”. Замінити рядки з MM до NN на текст TEXT. Текст може бути багаторядковим. Переведення рядка записується як \n, символ зворотного слеша як \\ , лапки - \”. Секції from MM і to NN опційні
   III. insert after MM “TEXT”. Вставити TEXT після рядка MM.
   IV. replace from MM to NN “TEXT” with “TEXT2”. Замінити в рядках від MM до NN текст TEXT на TEXT2. Секції from MM і to NN опційні.
   V. undo. Скасувати дію попередньої команди.
2) На виході отримаєм відредагований текст, який знаходиться в першому файлі.

в) Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1

Таблиця 6.1 – Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10 (з останніми обновленнями)	Windows 7/ Windows 8/Windows 10 (з останніми обновленнями)
Процесор	Intel® Pentium® III 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Pentium® D або AMD Athlon™ 64 X2
Оперативна пам’ять	256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/ Windows 8/Windows 10)	2 GB RAM
Відеоадаптер	Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще
Прилади введення	Клавіатура, комп’ютерна миша	
Додаткове програмне забезпечення	Microsoft .Net Framework 4.5.2 або вище	

ВИСНОВОК

Під час розроблення програми “Робот-редактор” мною були вивчені нові принципи побудови програми. Після виконання даної роботи я зрозумів що ООП принцип я досить зручний у користуванні під час побудови програми, адже дозволяє легко розуміти навіть чужий код, розробляти принципи роботи на основі діаграм, не затрудняючись вносити зміни у уже існуючи код. Під час проведення аналізу предметної області та опису архітектури програмної системи, я зрозумів наскільки зручно та ефективно користуватися діаграмами, які представленні за допомогою UML.

У ході виконання даної роботи було розроблено чотири алгоритми для роботи з текстом. У розділі тестування програмного забезпечення ми побачили що всі алгоритми працюють вірно, причому за досить короткий проміжок часу. Також на прикладі команди undo було показано що не обов’язково виконувати всі функції, а можна схитрити і позбутися непотрібних команд. Всі результати та програмна реалізація знаходиться у відкритому доступі на GitHub за посиланням [5].

В результаті виконання даної роботи ми отримали програму, яка може допомогти швидко відредагувати текст. Це може бути корисно як школяру, студенту так і офісним працівникам, адже вона значно зменшує час редагування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Седжвик Роберт Фундаментальные алгоритмы на C++. Анализ/ Структуры данных/Сортировка/Поиск: Пер. с англ./Роберт Седжвик. – К: Издательство «ДиаСофт», 2001 – 688 с.
2. Лафоре Роберт Объектно-ориентированное программирование в C++, 4-е изд. : Пер. с англ./ Роберт Лафоре. – Питер : Издательство «Питер СПб», 2018 – 928 с.
3. Довідка по C/C++ - vector, queue [«Електронний ресурс»] Режим доступу до статті: https://ru.cppreference.com/w/Заглавная_страница
4. Вікіпедія - Перевантаження операторівв в C++ [«Електронний ресурс»] Режим доступу до статті: https://uk.wikipedia.org/wiki/Оператори_в_C_та_C++
5. Git – розподілена система керування версіями файлів [«Електронний ресурс»] Режим доступу до статті: https://github.com/Siusarna/Coursework

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко М.М

«___»_____201_ р.

Виконавець:

Студент Петруняк Дмитро Васильович

«___» _____201_ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Робот-редактор»

з дисципліни:

«Основи програмування -2. Основи об'єктно-орієнтоване програмування»

Київ 2019

1. *Мета*: Метою курсової роботи є розробка програми, яка буде редагувати файл за заданим «рецептом» і зберігати результат в інший файл.
2. *Дата початку роботи*: «_____»_____201_ р.
3. *Дата закінчення роботи*: «___»_____ 201_ р.
4. *Вимоги до програмного забезпечення*.
    1) Функціональні вимоги:
        - Можливість видаляти рядки від MM до NN.
        - Можливість замінити рядки з MM до NN на текст TEXT.
        - Вставляти TEXT після рядка MM.
        - Замінити в рядках від MM до NN текст TEXT на TEXT2
        - Скасувати дію попередньої команди
        - Виведення результату роботи в файл;
        - Можливість роботи програми без MM і/або NN. В такому випадку програма буде опрацьовувати всі рядки з початку і/або до кінця
        - Робота з графічним інтерфейсом
    2) Нефункціональні вимоги:
        - Програма повинна мати доступ до файлів з вхідними данними
        - На комп’ютері повинна бути встановленна ОС родини Microsoft Windows.
        - Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:
        - ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.
        - ГОСТ 19.106 - 78 - Вимоги до програмної документації.

- ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

5. Стадії та етапи розробки:

1) Об'єктно-орієнтований аналіз предметної області задачі (до__.__.201_ р.)

2) Об'єктно-орієнтоване проектування архітектури програмної системи (до __.__.201_р.)

3) Розробка програмного забезпечення (до __.__.201_р.)

4) Тестування розробленої програми (до __.__.201_р.)

5) Розробка пояснювальної записки (до __.__.201_ р.).

6) Захист курсової роботи (до __.__.201_ р.).

6. Порядок контролю та приймання. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду програмного забезпечення
«Редагування текстових файлів»

(Найменування програми (документа))

CD-RW

(Вид носія даних)

14 арк, 124 Кб

(Обсяг програми (документа), арк., Кб)

студента групи ІП-81 І курсу
Петруняка Д.В

ПРОГРАМНИЙ КОД

MyForm.cpp

```
#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(cli::array<String^>^ args){

	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Project105::MyForm form;
	Application::Run(%form);

}
```

MyForm.h

```
#include"Robot.h"
#include <msclr\marshal_cppstd.h>
#pragma once

namespace Project105 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

    protected: <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
```

```
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());
			this->label1 = (gcnew System::Windows::Forms::Label());
			this->label2 = (gcnew System::Windows::Forms::Label());
			this->label3 = (gcnew System::Windows::Forms::Label());
			this->button2 = (gcnew System::Windows::Forms::Button());
			this->label4 = (gcnew System::Windows::Forms::Label());
			this->textBox1 = (gcnew System::Windows::Forms::TextBox());
			this->label5 = (gcnew System::Windows::Forms::Label());
			this->button3 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->BackColor = System::Drawing::SystemColors::ScrollBar;
			this->button1->Location = System::Drawing::Point(419, 12);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(128, 41);
			this->button1->TabIndex = 0;
			this->button1->Text = L"Open file";
			this->button1->UseVisualStyleBackColor = false;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
			//
			// openFileDialog1
			//
			this->openFileDialog1->FileName = L"openFileDialog1";
			this->openFileDialog1->FileOk += gcnew
System::ComponentModel::CancelEventHandler(this, &MyForm::openFileDialog1_FileOk);
			//
			// label1
this->label1->AutoSize = true;
			this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
				static_cast<System::Byte>(204)));
			this->label1->ForeColor = System::Drawing::SystemColors::ControlText;
			this->label1->Location = System::Drawing::Point(2, 17);
			this->label1->Name = L"label1";
			this->label1->Size = System::Drawing::Size(310, 25);
			this->label1->TabIndex = 1;
			this->label1->Text = L"Виберіть файл для редагування";
			//
			// label2
			//
			this->label2->AutoSize = true;
			this->label2->Location = System::Drawing::Point(12, 406);
			this->label2->Name = L"label2";
			this->label2->Size = System::Drawing::Size(46, 17);
			this->label2->TabIndex = 2;
			this->label2->Text = L"label2";
			this->label2->UseWaitCursor = true;
			this->label2->Visible = false;
			//
			// label3
			//
```

```
                this->label3->AutoSize = true;
                this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label3->ForeColor = System::Drawing::SystemColors::ControlText;
                this->label3->Location = System::Drawing::Point(2, 83);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(226, 25);
                this->label3->TabIndex = 3;
                this->label3->Text = L"Виберіть файл-рецепт";
                //
                // button2
                //
                this->button2->BackColor = System::Drawing::SystemColors::ScrollBar;
                this->button2->Location = System::Drawing::Point(419, 74);
                this->button2->Name = L"button2";
                this->button2->Size = System::Drawing::Size(128, 41);
                this->button2->TabIndex = 4;
                this->button2->Text = L"Open recipe";
                this->button2->UseVisualStyleBackColor = false;
                this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
                //
this->label4->AutoSize = true;
                this->label4->Location = System::Drawing::Point(70, 406);
                this->label4->Name = L"label4";
                this->label4->Size = System::Drawing::Size(46, 17);
                this->label4->TabIndex = 5;
                this->label4->Text = L"";
                this->label4->Visible = false;
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(677, 74);
                this->textBox1->Multiline = true;
                this->textBox1->Name = L"textBox1";
                this->textBox1->ReadOnly = true;
                this->textBox1->Size = System::Drawing::Size(441, 100);
                this->textBox1->TabIndex = 6;
                this->textBox1->Visible = false;
                //
                // label5
                //
                this->label5->AutoSize = true;
                this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label5->ForeColor = System::Drawing::SystemColors::ControlText;
                this->label5->Location = System::Drawing::Point(815, 28);
                this->label5->Name = L"label5";
                this->label5->Size = System::Drawing::Size(139, 25);
                this->label5->TabIndex = 7;
                this->label5->Text = L"Файл-рецепт";
                this->label5->Visible = false;
                //
                // button3
                //
                this->button3->BackColor = System::Drawing::SystemColors::Desktop;
```

```
			this->button3->ForeColor = System::Drawing::SystemColors::ButtonFace;
			this->button3->Location = System::Drawing::Point(136, 211);
			this->button3->Margin = System::Windows::Forms::Padding(0);
			this->button3->Name = L"button3";
			this->button3->Size = System::Drawing::Size(176, 48);
			this->button3->TabIndex = 8;
			this->button3->Text = L"Запустити робота";
			this->button3->UseVisualStyleBackColor = false;
			this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
			//
this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->BackColor = System::Drawing::SystemColors::ActiveCaption;
			this->ClientSize = System::Drawing::Size(1168, 466);
			this->Controls->Add(this->button3);
			this->Controls->Add(this->label5);
			this->Controls->Add(this->textBox1);
			this->Controls->Add(this->label4);
			this->Controls->Add(this->button2);
			this->Controls->Add(this->label3);
			this->Controls->Add(this->label2);
			this->Controls->Add(this->label1);
			this->Controls->Add(this->button1);
			this->Name = L"MyForm";
			this->Text = L"MyForm";
			this->ResumeLayout(false);
			this->PerformLayout();

		}
#pragma endregion
	private: System::Void openFileDialog1_FileOk(System::Object^  sender,
System::ComponentModel::CancelEventArgs^  e) {
	}
	private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
		Stream^ myStream;
		OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog;

		openFileDialog1->InitialDirectory = "b:\\Test";
		openFileDialog1->Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
		openFileDialog1->FilterIndex = 1;
		openFileDialog1->RestoreDirectory = true;

		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
		{
			if ((myStream = openFileDialog1->OpenFile()) != nullptr)
			{
				// Insert code to read the stream here.

				// Add file name without path to the form title

				Text = String::Concat("Simple Text File Viewer - ", openFileDialog1-
>SafeFileName);

				// Write complete file path to file path text box

				label2->Text = openFileDialog1->FileName;
```

```
                        // Load text file contents into viewer text box:

                        myStream->Close();
                    }
                }
            }
            private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
                Stream^ myStream;
                OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog;

                openFileDialog1->InitialDirectory = "b:\KPI\OP\Coursework\Test";
                openFileDialog1->Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
                openFileDialog1->FilterIndex = 1;
                openFileDialog1->RestoreDirectory = true;

                if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
                {
                    if ((myStream = openFileDialog1->OpenFile()) != nullptr)
                    {
                        // Insert code to read the stream here.

                        // Add file name without path to the form title

                        Text = String::Concat("Simple Text File Viewer - ", openFileDialog1-
>SafeFileName);

                        // Write complete file path to file path text box

                        label4->Text = openFileDialog1->FileName;

                        // Load text file contents into viewer text box:
                        textBox1->Text = (gcnew StreamReader(myStream))->ReadToEnd();
                        textBox1->Visible = true;
                        label5->Visible = true;

                        myStream->Close();
                    }
                }
            }
            private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
                string recipe = msclr::interop::marshal_as<std::string>(label4->Text);
                string text = msclr::interop::marshal_as<std::string>(label2->Text);
                if (label4->Text->Length == 0) {
                    System::String^ err_name = gcnew System::String("Помилка");
                    System::String^ err_descr = gcnew System::String("Ви не вибрали файл-рецепт");
                    System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                }
                else if (label2->Text->Length == 0) {
                    System::String^ err_name = gcnew System::String("Помилка");
                    System::String^ err_descr = gcnew System::String("Ви не вибрали файл для
редагування");
                    System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                }
                else {
                    Robot text_editor(recipe, text);
                    text_editor.start_work();
```

```
                        MessageBox::Show("Редагування виконано успішно");
                        label4->Text = "";
                }
        }
};
}
```

Robot.h

```
#pragma once
#include <fstream>
#include <sstream>
#include <iostream>
#include <stack>
#include <queue>
#include <vector>
#include <string>

//using namespace System::Windows::Forms;
using namespace std;

struct option
{
        string word;
        string row;
};

class Robot
{
public:
        Robot(string path_recipe, string path_text);

        void start_work();
        void delete_text(option recipe);
        void change_text(option recipe);
        void insert_text(option recipe);
        void replace_text(option recipe);

private:
        queue<option> que;
        vector<string> text;
        string path_recipe;
        string path_text;

        void parce_for_delete(string recipe, int &from, int &to);
        void parce_for_change(string recipe,  int &from, int &to, vector<string> &text_for_change);
        void parce_for_insert(string recipe, int &after, vector<string> &text_for_insert);
        void parce_for_replace(string recipe, int &from, int &to, vector<string> &text_from, vector<string>
&text_to);

        vector<string> Open_Text(string path);
        queue<option> Open_Recipe(string path);

};
```

Robot.cpp

```
#include "Robot.h"
```

```
bool parce_row(string &row) {
	stringstream s(row);
	string temp;
	string word = "delete change insert replace undo";
	do {
		getline(s, temp, ' ');
		if (word.find(temp) != string::npos) {
			row = temp;
			return true;
		}
	} while (getline(s, temp, ' '));
	return false;
}

queue<option> Robot::Open_Recipe(string path) {
	ifstream fin;
	fin.open(path);
	queue<option> q;
	string str;
	string t;
	option temp;
	while (!fin.eof()) {
		getline(fin, str);
		t = str;
		if (parce_row(str)) {
			temp.word = str;
			temp.row = t;
			q.push(temp);
		}
	}
	return q;
}

vector<string> Robot::Open_Text(string path) {
	ifstream fin;
	fin.open(path);
	vector<string> q;
	string str;
	while (!fin.eof()) {
		getline(fin, str);
		q.push_back(str);
	}
	return q;
}


queue<option> reversQueue(queue<option> q) {
	stack<option> stack;
	while (!q.empty()) {
		stack.push(q.front());
		q.pop();
	}
	while (!stack.empty()) {
		q.push(stack.top());
		stack.pop();
	}
```

```
	return q;
}

void Robot::start_work() {
	queue<option> temp = reversQueue(this->que);
	string str;
	queue<option> temp1;
	unsigned int size = temp.size();
	for(unsigned int i=0;i<size;i++) {
		if (temp.front().word == "undo") {
			temp.pop();
			temp.pop();
			size--;
		}
		else {
			temp1.push(temp.front());
			temp.pop();
		}
	}
	this->que = reversQueue(temp1);
	while (!this->que.empty()){
		str = this->que.front().word;
		if (str == "delete") delete_text(this->que.front());
		if (str == "change") change_text(this->que.front());
		if (str == "insert") insert_text(this->que.front());
		if (str == "replace") replace_text(this->que.front());
		this->que.pop();
	}
	ofstream out("result.txt");
	for (int i = 0; i < this->text.size(); i++) {
		out << this->text[i] << "\n";
	}
}

Robot::Robot(string path_recipe, string path_text)
{
	this->que = Open_Recipe(path_recipe);
	this->text = Open_Text(path_text);
	this->path_recipe = path_recipe;
	this->path_text = path_text;
}
```

delete.cpp

```
#include "Robot.h"

void Robot::delete_text(option recipe) {
	int from = 0;
	int to = 0;
	parce_for_delete(recipe.row, from, to);
	if (from < 1) from = 1;
	if (to > this->text.size()) to = this->text.size();
	this->text.erase(this->text.begin() + from - 1, this->text.begin() + to);
}

void Robot::parce_for_delete(string recipe, int& from, int& to) {
	if (recipe.find("from") == string::npos) from = 0;
	if (recipe.find("to") == string::npos) to = this->text.size();
```

```
	if (from != 0 && to != 0) return;
	string temp;
	stringstream s(recipe);
	getline(s, temp, ' ');
	do {
		if (temp == "from") {
			getline(s, temp, ' ');
			try
			{
				from = stoi(temp);
			}
			catch (const std::exception& ex)
			{
				System::String^ err_name = gcnew System::String("Помилка");
				System::String^ err_descr = gcnew System::String(ex.what());
				err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
				System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
				delete err_name;
				delete err_descr;
				System::Windows::Forms::Application::Exit();
			}

		}
		if (temp == "to") {
			getline(s, temp, ' ');
			try
			{
				to = stoi(temp);
			}
			catch (const std::exception& ex)
			{
				System::String^ err_name = gcnew System::String("Помилка");
				System::String^ err_descr = gcnew System::String(ex.what());
				err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
				System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
				delete err_name;
				delete err_descr;
				System::Windows::Forms::Application::Exit();
			}
		}
	} while (getline(s, temp, ' '));
}
```

change.cpp

```
#include "Robot.h"


void Robot::change_text(option recipe) {
	int from;
	int to;
	vector<string> text_for_change;
	parce_for_change(recipe.row, from, to, text_for_change);
	if (from > 0) from--;
	if (from < 1) from = 0;
```

```
	if (to > this->text.size()) to = this->text.size();
	if ((to - from) > text_for_change.size()) {
		this->text.erase(this->text.begin() + from + text_for_change.size() - 1, this->text.begin() + to
- 1);
		for (unsigned int i = 0; i < text_for_change.size(); i++) {
			this->text[i + from] = text_for_change[i];
		}
	}
	else {
		int k = 0;
		for (unsigned int i = from; i < to; i++) {
			this->text[i] = text_for_change[k];
			k++;
		}
		for (unsigned int i = to; i < text_for_change.size(); i++) {
			this->text.insert(this->text.begin() + i, text_for_change[k]);
			k++;
		}
	}
}

void Robot::parce_for_change(string recipe, int& from, int& to, vector<string>& text_for_change) {
	if (recipe.find("from") == string::npos) from = 0;
	if (recipe.find("to") == string::npos) to = this->text.size();
	string temp, temp1;
	stringstream s(recipe);
	getline(s, temp, ' ');
	do {
		if (temp == "from") {
			getline(s, temp, ' ');
			try
			{
				from = stoi(temp);
			}
			catch (const std::exception& ex)
			{
				System::String^ err_name = gcnew System::String("Помилка");
				System::String^ err_descr = gcnew System::String(ex.what());
				err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
				System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
				delete err_name;
				delete err_descr;
				System::Windows::Forms::Application::Exit();
			}
		}
		if (temp == "to") {
			getline(s, temp, ' ');
			try
			{
				to = stoi(temp);
			}
			catch (const std::exception& ex)
			{
				System::String^ err_name = gcnew System::String("Помилка");
				System::String^ err_descr = gcnew System::String(ex.what());
```

```
                                        err_descr += gcnew System::String(". Поміняйте рецепт та запустіть робота ще раз");
                                        System::Windows::Forms::MessageBox::Show(err_descr, err_name, System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                                        delete err_name;
                                        delete err_descr;
                                        System::Windows::Forms::Application::Exit();
                                }
                        }
                        if (temp == "with") {
                                getline(s, temp, "");
                                getline(s, temp);
                                temp1 = "";
                                for (int i = 0; i < temp.size(); i++) {
                                        if (temp[i] == '\\' && temp[i + 1] == 'n' || temp[i] == "") {
                                                text_for_change.push_back(temp1);
                                                i++;
                                                temp1.clear();
                                        }
                                        else if (temp[i] == '\\' && temp[i + 1] == "") {
                                                temp1 += "";
                                                i++;
                                        }
                                        else if (temp[i] == '\\' && temp[i + 1] == '\\') {
                                                temp += '\\';
                                        }
                                        else if (temp[i] == '\\') {
                                                temp1 += '\\';
                                        }
                                        else temp1 += temp[i];
                                }
                        }
        } while (getline(s, temp, ' '));
}
```

insert.cpp

```
#include "Robot.h"

void Robot::insert_text(option recipe) {
        int after;
        vector<string> text_for_insert;
        parce_for_insert(recipe.row, after, text_for_insert);
        if (after < 1) after = 1;
        if (after > this->text.size()) after = this->text.size();
        int k = 0;
        for (unsigned int i = 0; i < text_for_insert.size(); i++) {
                this->text.insert(this->text.begin() + after + k, text_for_insert[k]);
                k++;
        }
}

void Robot::parce_for_insert(string recipe, int& after, vector<string>& text_for_insert) {
        string temp, temp1;
        stringstream s(recipe);
        getline(s, temp, ' ');
        do {
                if (temp == "after") {
                        getline(s, temp, ' ');
```

```
                        try
                        {
                                after = stoi(temp);
                        }
                        catch (const std::exception& ex)
                        {
                                System::String^ err_name = gcnew System::String("Помилка");
                                System::String^ err_descr = gcnew System::String(ex.what());
                                err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
                                System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                                delete err_name;
                                delete err_descr;
                                System::Windows::Forms::Application::Exit();
                        }
                        getline(s, temp, "");
                        getline(s, temp);
                        temp1 = "";
                        for (int i = 0; i < temp.size(); i++) {
                                if (temp[i] == '\\' && temp[i + 1] == 'n' || temp[i] == "") {
                                        text_for_insert.push_back(temp1);
                                        i++;
                                        temp1.clear();
                                }
                                else if (temp[i] == '\\' && temp[i + 1] == "") {
                                        temp1 += "";
                                        i++;
                                }
                                else if (temp[i] == '\\' && temp[i + 1] == '\\') {
                                        temp += '\\';
                                }
                                else if (temp[i] == '\\') {
                                        temp1 += '\\';
                                }
                                else temp1 += temp[i];
                        }
                }
        } while (getline(s, temp, ' '));
}
```

replace.cpp

```
#include "Robot.h"

void Robot::parce_for_replace(string recipe, int& from, int& to, vector<string>& text_from,
vector<string>& text_to) {
        if (recipe.find("from") == string::npos) from = 0;
        if (recipe.find("to") == string::npos) to = this->text.size() - 1;
        string temp, temp1;
        stringstream s(recipe);
        getline(s, temp, ' ');
        do {
                if (temp == "from") {
                        getline(s, temp, ' ');
                        try
                        {
```

```
                        from = stoi(temp);
                    }
                    catch (const std::exception& ex)
                    {
                        System::String^ err_name = gcnew System::String("Помилка");
                        System::String^ err_descr = gcnew System::String(ex.what());
                        err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
                        System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                        delete err_name;
                        delete err_descr;
                        System::Windows::Forms::Application::Exit();
                    }

                }
                if (temp == "to") {
                    getline(s, temp, ' ');
                    try
                    {
                        to = stoi(temp);
                        getline(s, temp, "");
                        getline(s, temp, "");
                        temp1 = "";
                        for (int i = 0; i < temp.size(); i++) {
                            if (temp[i] == '\\' && temp[i + 1] == 'n') {
                                text_from.push_back(temp1);
                                i++;
                                temp1.clear();
                            }
                            else if (temp[i] == '\\' && temp[i + 1] == "") {
                                temp1 += "";
                                i++;
                            }
                            else if (temp[i] == '\\' && temp[i + 1] == '\\') {
                                temp += '\\';
                            }
                            else if (temp[i] == '\\') {
                                temp1 += '\\';
                            }
                            else temp1 += temp[i];
                        }
                        text_from.push_back(temp1);
                    }
                    catch (const std::exception& ex)
                    {
                        System::String^ err_name = gcnew System::String("Помилка");
                        System::String^ err_descr = gcnew System::String(ex.what());
                        err_descr += gcnew System::String(". Поміняйте рецепт та запустіть
робота ще раз");
                        System::Windows::Forms::MessageBox::Show(err_descr, err_name,
System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Error);
                        delete err_name;
                        delete err_descr;
                        System::Windows::Forms::Application::Exit();
                    }
                }
                if (temp == "with") {
```

```
                        getline(s, temp, "");
                        getline(s, temp);
                        temp1 = "";
                        for (int i = 0; i < temp.size(); i++) {
                                if (temp[i] == '\\' && temp[i + 1] == 'n') {
                                        text_to.push_back(temp1);
                                        i++;
                                        temp1.clear();
                                }
                                else if (temp[i] == '\\' && temp[i + 1] == "") {
                                        temp1 += "";
                                        i++;
                                }
                                else if (temp[i] == '\\' && temp[i + 1] == '\\') {
                                        temp += '\\';
                                }
                                else if (temp[i] == '\\') {
                                        temp1 += '\\';
                                }
                                else temp1 += temp[i];
                        }
                        text_to.push_back(temp1);
                }
        } while (getline(s, temp, ' '));
}

void Robot::replace_text(option recipe) {
        int from, to;
        size_t pos;
        vector<string> text_from, text_to;
        parce_for_replace(recipe.row, from, to, text_from, text_to);
        if (from < 1) from = 1;
        if (to > this->text.size()) to = this->text.size();
        if (text_from.size() == 1) {
                for (int i = from - 1; i < to; i++) {
                        pos = 0;
                        while ((pos = this->text[i].find(text_from[0], pos)) != string::npos) {
                                this->text[i].replace(pos, text_from[0].length(), text_to[0]);
                                pos += text_to[0].length();
                                if (text_to.size() > 1) {
                                        string save;
                                        for (int l = pos; l < this->text[i].size(); l++) {
                                                save += this->text[i][l];
                                        }
                                        int size = this->text[i].size();
                                        this->text[i].erase(this->text[i].begin() + pos, this->text[i].begin() +
size);
                                        int k = 1;
                                        for (int j = 1; j < text_to.size(); j++) {
                                                this->text.insert(this->text.begin() + i + k, text_to[k]);
                                                to++;
                                                k++;
                                        }
                                        this->text[i + text_to.size() - 1] += save;
                                }
                        }
                }
        }
```

```
	else {
		for (int i = from - 1; i < to - 1; i++) {
			bool flag = true;
			for (int k = 0; k < text_from.size(); k++) {
				for (int j = 0; j < this->text[i].size(); j++) {
					if (this->text[i + k][j] != text_from[k][j]) {
						flag = false;
						break;
					}
				}
			}
			if (flag) {
				int k = 0;
				if (text_to.size() > text_from.size()) {
					for (k = 0; k < text_from.size(); k++) {
						if (this->text[i + k].size() < text_to[k].size()) {
							int j = 0;
							while (j < this->text[i].size()) {
								this->text[i + k][j] = text_to[k][j];
								j++;
							}
							for (int q = j; q < text_to[k].size(); q++) {
								this->text[i + k] += text[k][q];
							}
						}
						else {
							for (int j = 0; j < text_to[k].size(); j++) {
								this->text[i + k][j] = text_to[k][j];
							}
						}
					}
					for (int y = 0; y < text_to.size() - k; y++) {
						this->text.insert(this->text.begin() + i + k, text_to[k + y]);
					}
				}
				if (text_from.size() > text_to.size()) {
					this->text.erase(this->text.begin() + i + k, this->text.begin() + i + k +
(text_from.size() - text_to.size()));
				}
			}
		}
	}
}
```