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

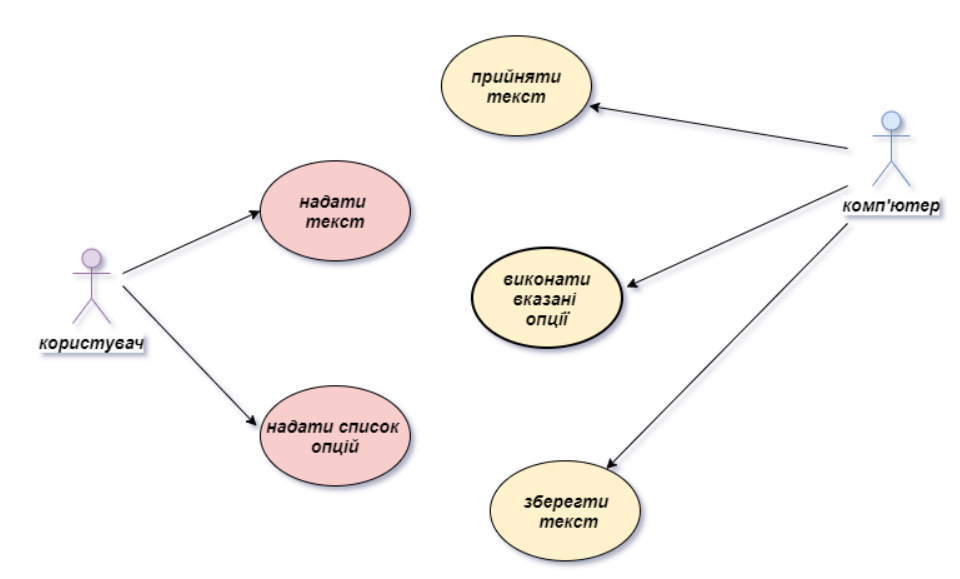
 На рисунку 2.1 зображено діаграму прецедентів.

Рисунок 2.1 – Діаграма прецедентів

Сценарій поведінки системи:

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

Для опису загального алгоритму 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:
7. ПОКИ не дістанемся кінця файла, будемо зчитувати по рядку і записувати в Text;
8. Зчитати файл-рецепт:
9. ПОКИ не дістанемся кінця файла, будемо зчитувати по рядку і записувати в que;
10. Цикл проходу всіх елементів que(): ЯКЩО дорівнює “undo”, ТО видаляєм попередній елемент;
11. ПОКИ que не порожнє:
12. Дістаєм перший елемент:
13. ЯКЩО він дорівнює “delete” викликаєм б;
14. ЯКЩО він дорівнює “change” викликаєм в;
15. ЯКЩО він дорівнює “insert” викликаєм г;
16. ЯКЩО він дорівнює “replace” викликаєм д;
17. Викликати вікно про успішне завершення програми;
18. КІНЕЦЬ;

б) Алгоритм методу delete:

1. З рядка que[i] отримати значення з якого рядка (ММ) 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 (ММ) 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):
6. Викликати алгоритм delete (пункт б) ) для рядків від (MM + size(TEXT)-1) до (NN -1);
7. Цикл проходу по всіх елементах TEXT: Text[i+NN] = TEXT[i];
8. ІНАКШЕ:
9. Цикл проходу від MM до NN: Text[i] = TEXT[i];
10. Цикл проходу від NN до size(TEXT): вставлення TEXT[k] після k;
11. КІНЕЦЬ;

г) Алгоритм методу insert:

1. З рядка que[i] отримати значення після якого рядка (ММ) вставляти текст, та TEXT на який потрібно вставити;
2. ЯКЩО не вдало прочитати MМ, ТО вивести вікно про неправильно введені дані;
3. ЯКЩО MM< 1, ТО MM = 1;
4. ЯКЩО ММ>size(Text), ТО ММ=size(Text);
5. Цикл проходу від і=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 (ММ) і до якого (NN) заміняти текст, TEXT1 який потрібно замінити та TEXT2, на який потрібно замінити;
2. ЯКЩО не вдало прочитати MМ або NN ТО вивести вікно про неправильно введені дані;
3. ЯКЩО MM< 1, ТО MM = 1;
4. ЯКЩО ММ>size(Text), ТО ММ=size(Text);
5. ЯКЩО size(TEXT1) == 1 то:
6. Цикл проходу від і=MM-1 до NN:
7. ПОКИ в рядках від MM до NN буде TEXT1:
8. Виконати заміну в Text[i] на TEXT[0];
9. ЯКЩО size(TEXT2 > 1):
10. Видаляєм рядки від і до size(Text)
11. Цикл проходу від j=1 до size(TEXT2): після j вставляєм TEXT2[j]
12. ІНАКШЕ:
13. Перевірити чи існує в Text підрядок TEXT1,ЯКЩО не існує, то перейти до пункт 7, ІНАКШЕ:
14. ЯКЩО size(TEXT2) > size(TEXT1):
15. Цикл проходу від k=0 до k< size(TEXT1):
16. ЯКЩО size(Text[k]) < size(TEXT2), ТО ітеруємся від j = k до size(Text[i]): Text[j+k][j] = TEXT2[k][j];
17. ІНАКШЕ, ітеруємся від j =0 до size(TEXT2): Text[j+k][j] = TEXT2[k][j];
18. Цикл проходу від y = 0 до size(TEXT2): вставляєм після (k+y) рядок TEXT2[k+y];
19. ІНАКШЕ видаляєм рядки від (k) до (k + (size(TEXT1) – size(TEXT2)));
20. 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.

На рисунку 3.1 представлено діаграму класів.

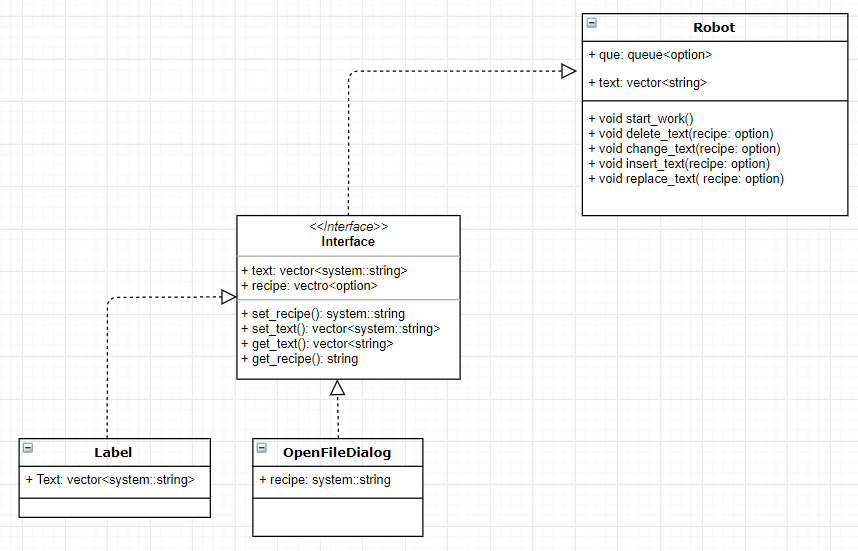


Рисунок 3.1- Діаграма класів.

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

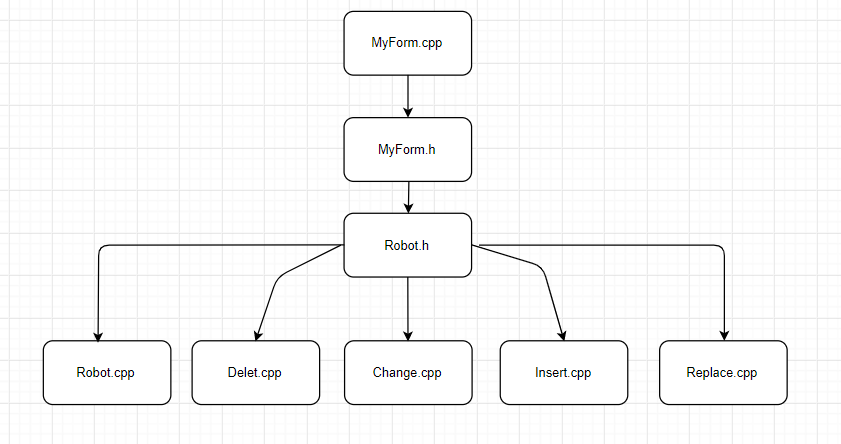


Рисунок 4.1 – Структура програми

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

1. MyForm.h – реалізує інтерфейс користувача
2. Robot.h – реалізує клас Robot, і його методи:
3. Видалення вказаних рядків
4. Заміна вказаних рядків на TEXT (під TEXT розуміється текст введений користувачем)
5. Вставка рядка після вказаного номера
6. 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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | Відредагова-ний 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 | Відредагова-ний 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 | Відредагова-ний 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 | Відредагова-ний 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 | Стандартна бібліотека (клас) |
| 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 | іnsert | 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, 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.

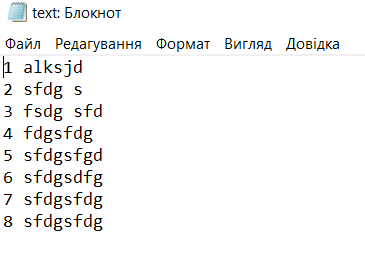


Рисунок 5.1 – Вхідний текст

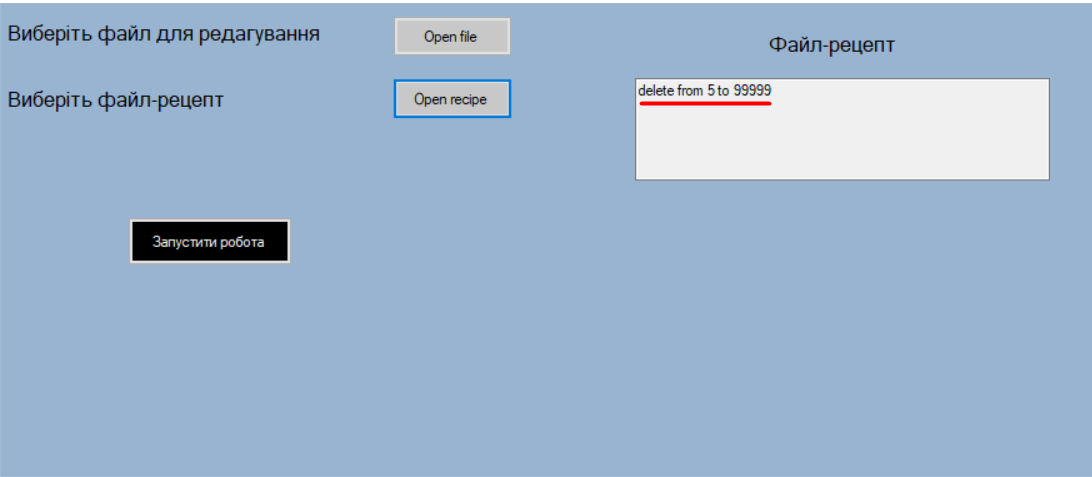


Рисунок 5.2 – Вхідна команда

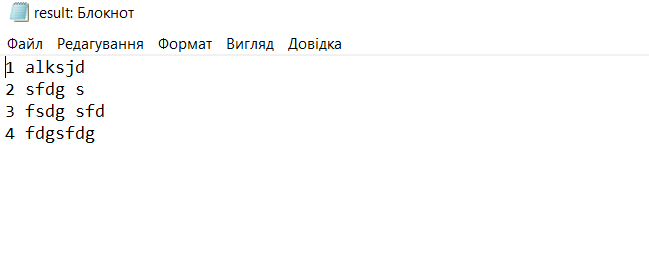


Рисунок 5.3 – Результат виконання команди

б) Тестування програми при невказаних шляхах до файлів:

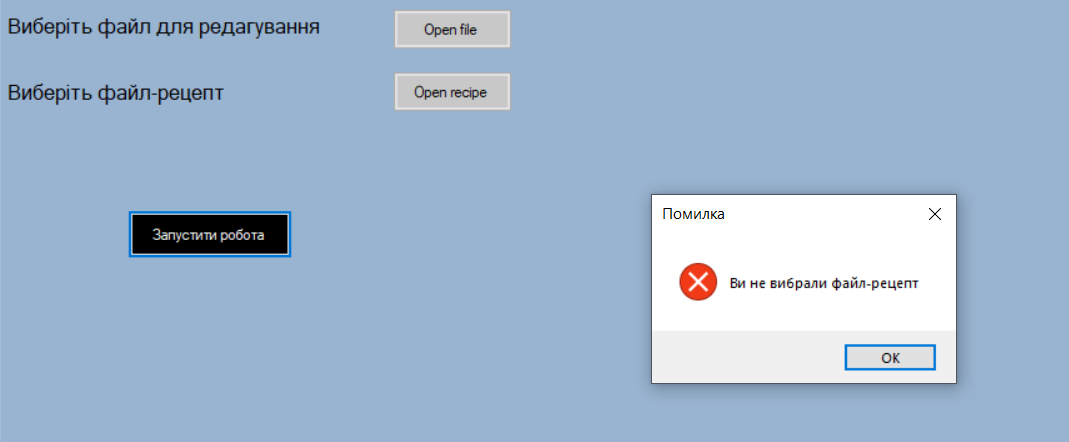
 Якщо під час роботи з програмою користувач не вибере шляхи до файлів, які вказані як вхідні данні, то появиться вікно помилку, і користувачу надасться змога вибрати шлях ще раз, і так до тих пір поки користувач не вибере шлях або не завершить програму. Це можна побачити на рисунку 5.4.

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 2А5). 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

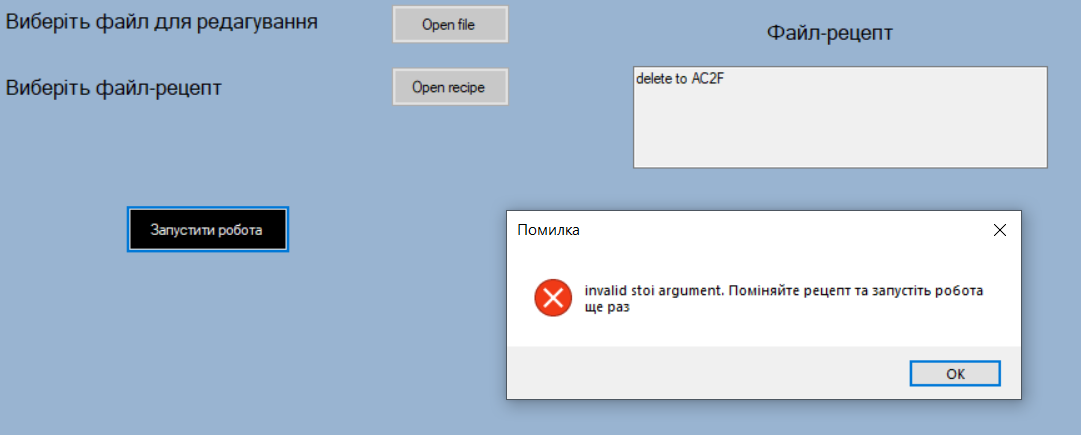
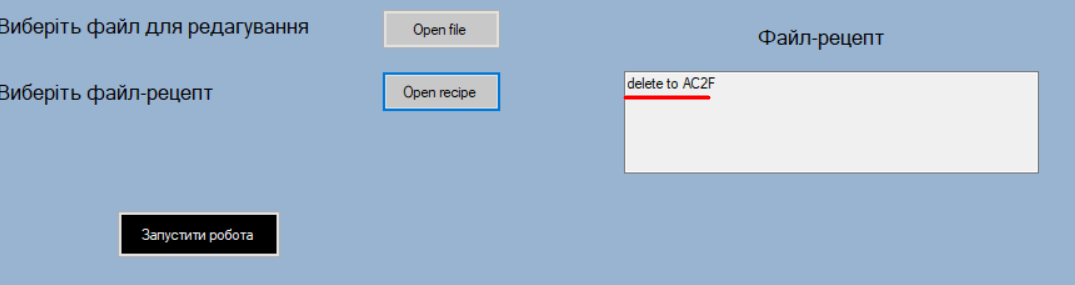
Рисунок 5.5 – Неправильно введене число

Рисунок 5.6 – Результат обробки неправильних даних

г) Тестування всіх методів програми:

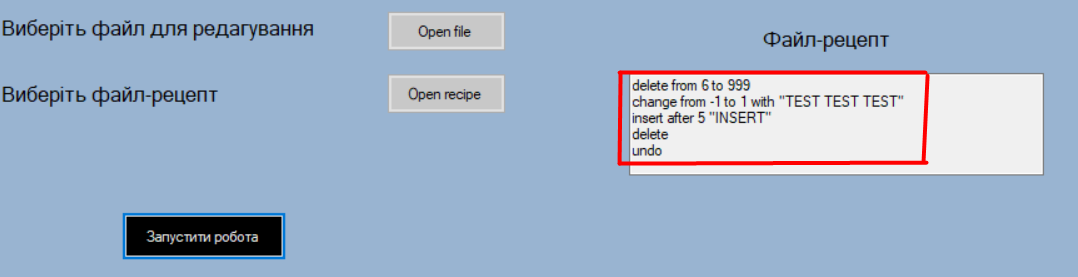
 Для цього тесту візьмем файл-рецепт в якому вказані всі можливі команди, його ми можемо побачити на рисунку 5.7, та запустимо для тексту, вказаного на рисунку 5.1. Результат можемо побачити на рисунку 5.8. І справді рядок з 6 до 999 (тобто до кінця) видалені. Рядки з -1(тобто з початку) до 1 рядка замінені 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

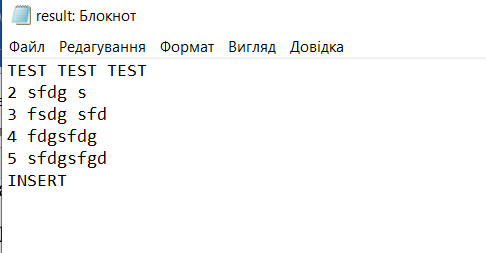


Рисунок 5.8 – Результат виконання програми

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

а) Кроки для успішної роботи програми:

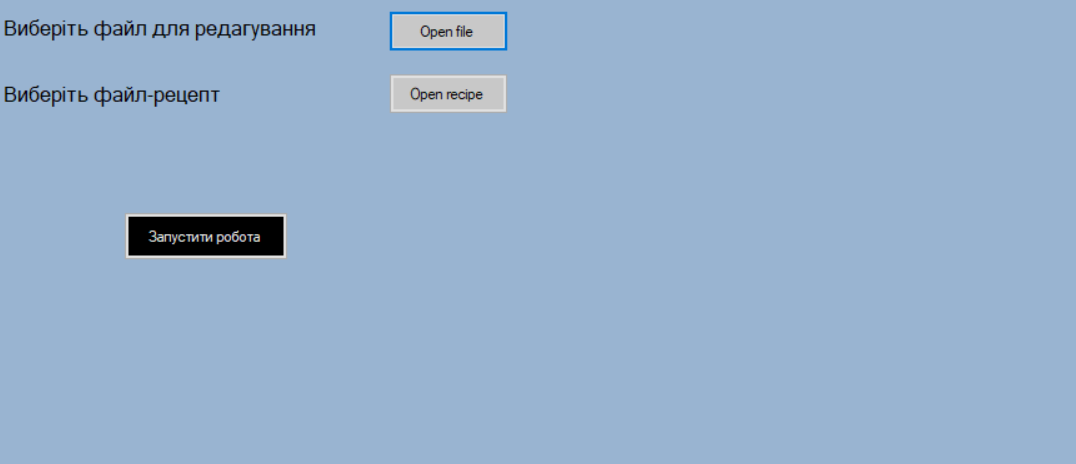
1) Натиснути кнопку Open file. На рисунку 6.1 вона підсвічена;

Рисунок 6.1 – Кнопка Open file

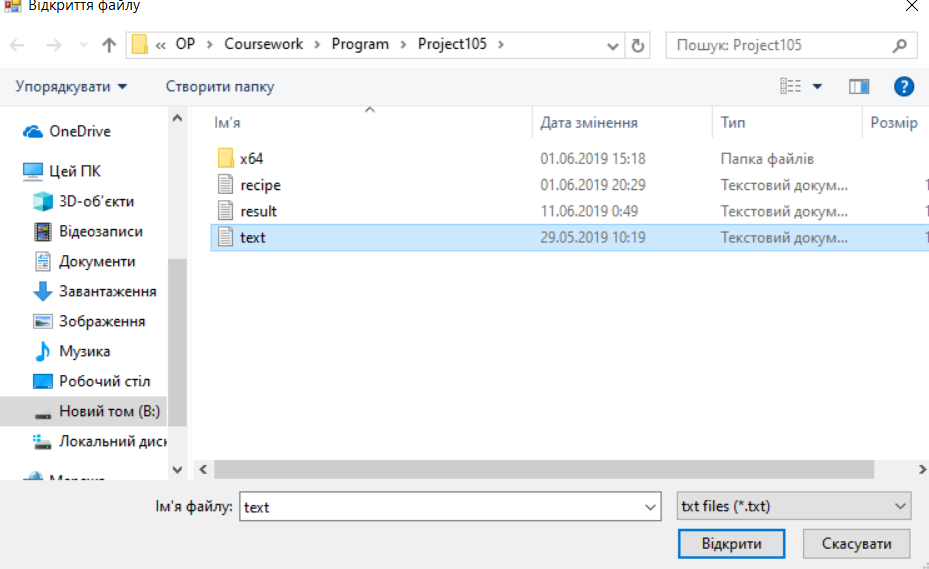
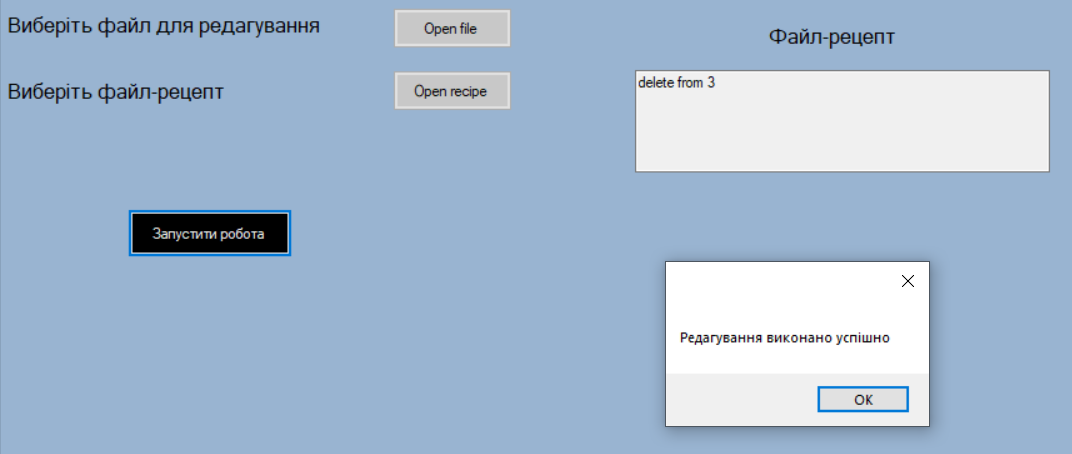
2) Вибрати файл, та натиснути Відкрити, рисунок 6.2;

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:
2. delete from MM to NN – видалити рядки з номерами MM..NN. Секції from MM i to NN можуть опускатися, в цьому випадку мається на увазі, що видаляти потрібно з початку або до кінця файлу.
3. 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 i to NN опційні
4. insert after MM “TEXT”. Вставити TEXT після рядка ММ.
5. replace from MM to NN “TEXT” with “TEXT2”. Замінити в рядках від MM до NN текст TEXT на TEXT2. Cекції from MM i to NN опційні.
6. undo. Скасувати дію попередньої команди.
7. 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® ІІІ  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 | 800х600 | 1024х768 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.